Politecnico di Milano
Dipartimento di Elettronica e Informazione
Dottorato di Ricerca in Ingegneria dell'Informazione

Scaling Up the Formal Analysis of Real-Time Systems

Ph.D. Dissertation of:
Carlo Alberto Furia

Advisor:
Prof. Dino Mandrioli
Coadvisor:
Dr. Matteo G. Rossi
Tutor:
Prof. Angelo Morzenti
Supervisor of the Doctoral Program:
Prof. Patrizio Colaneri

2007 - XIX

quare habe tibi quidquid hoc libelli
qualecumque

Abstract

We develop techniques and methods to scale the application of formal methods up to large and heterogenous systems. Our solution focuses on real-time systems, and develops along the two aspects of *compositionality* and *integration*.

In the first part, we provide a compositional framework for specifying and verifying properties of modular systems, written using metric temporal logics with a descriptive approach. The framework consists in technical as well as methodological features. On the technical side, we provide an array of compositional inference rules to deduce facts about the composition of modules from facts about each component in isolation. Each rule has different features that make it more suitable for certain classes of systems. We also provide a general methodology to guide the formal specification of modular systems, and the application of the inference rules to verify them.

In the second part, we consider the problem of integration for systems with both discrete-time and continuous-time components. We provide suitable conditions under which metric temporal logic formulas have a consistent truth value whether they are interpreted in discrete or in continuous time, and we identify a language subset that meets these conditions. We also characterize the conditions and the language subset with detail. Our approach to integration permits the verification of global properties of heterogenous systems.

Sommario

Questa tesi sviluppa tecniche e metodi per la scalabilità dell'applicazione di metodi formali a sistemi grandi ed eterogenei. Le soluzioni proposte si riferiscono a sistemi real-time, e sviluppano i due aspetti della *composizionalità* e dell'*integrazione*.

Nella prima parte, si descrive un framework composizionale per la specifica e verifica di proprietà di sistemi modulari, formalizzati con logiche temporali metriche attraverso un approccio descrittivo. Il framework è costituito da aspetti sia tecnici che metodologici. Per quanto riguarda quelli tecnici, si propongono una serie di regole di inferenza composizionali, per la deduzione di fatti sulla composizione di moduli a partire da fatti su ciascun componente isolato. Ogni regola ha caratteristiche proprie, che la rendono più adeguata all'uso con certe classi di sistemi. Si fornisce anche una metodologia generale per guidare l'attività di specifica formale di sistemi modulari, e l'applicazione delle regole di inferenza nella loro verifica.

Nella seconda parte, si considera il problema dell'integrazione per sistemi con sia componenti a tempo discreto che componenti a tempo continuo. Si definiscono condizioni sotto le quali le formule in logica temporale metrica conservano un valore di verità consistente sia che siano interpretate a tempo continuo sia che lo siano a tempo discreto, e si identifica un linguaggio che soddisfa queste condizioni. Inoltre, si caratterizzano dettagliatamente tali condizioni e tale linguaggio. L'approccio all'integrazione proposto permette la verifica di proprietà globali di sistemi eterogenei.

Acknowledgments

One of the favorite *adages* of my advisor's is about the importance of studying very abstract subjects — projective geometry [Var], in particular — to nurture one's *forma mentis*. I think this example accurately characterizes some of his traits: the importance given to principles, the independence of thought, and some old-fashioned attitude. I sincerely thank Dino Mandrioli for his indispensable guidance through my Ph.D. program, but also for his stimulating and invaluable suggestions and examples, and his attitude toward science and engineering, in a broad sense. I just hope he doesn't notice that this thesis is typeset in LaTeX.

Matteo Rossi has been an "elder brother in research" since my Master's years. I thank him especially for his constant suggestions and his scrupulous scrutiny of my work. They both have strongly contributed to address it in the right direction and to greatly improve it.

Besides his formal role as my Ph.D. *tutor*, Angelo Morzenti has effectively played the role of a second advisor. I thank him for his guidance, suggestions, and finical examples on how to express ideas clearly and cogently, a priceless skill in science.

I have spent some of my Ph.D. period visiting John Knight at the Computer Science Department of the University of Virginia. I thank John not only for his kind hospitality and his remarkable British wit, but also for the opportunity to develop a stimulating research program that, I'm sure, will become far-reaching in the future. From UVa, I also thank Elisabeth Strunk.

Although not part of this thesis, my Ph.D. program consisted also in a minor research. I sincerely thank Carlo "Mere" Mereghetti for advising me on the fascinating topic of quantum computing [Fur06a]. I hope that we will have new opportunities to work together ("unitarily" is probably the *mot juste* in this case) again on this challenging subject, in the very near future.

I also would like to thank Connie Heitmeyer for serving as official external reviewer of this thesis. Her remarks and detailed observations have greatly contributed to improving this piece of work.

Moving to the *younger* co-workers, I'd like to thank Paola Spoletini for her unique support and remarkable presence: she can really Spin the world (pun intended [Spo05]).

Among the other members of the research group I've worked in at Politecnico di Milano, I'd like to thank Matteo Pradella — for stimulating

conversations and his peculiar sense of humor — Pierluigi San Pietro, and Carlo Ghezzi (also for succeeding in the seemingly impossible task of moving my defense date twenty days sooner).

I'd also like to thank my various office-mates that have accompanied my three years at the Computer Science Department, and have always been excellent sources of confrontation and fun. Notice that, although they are many, the office space we shared was little. In the first round, besides Paola and Matteo whom I've already mentioned, let me thank Enzo Colombo, Danilo Ardagna, and Cinzia Cappiello (although strictly never an office-mate, "spiritually" always so). In the second round, in strict alphabetical order: Davide Balzarotti, Luca Cavallaro, Paolo Costa (and his P-browsing skills), Davide Frey, Sam Guinea, Paolo Mazzoni, Matteo Migliavacca, Alessandro Monguzzi, Luca Mottola, Filippo Pacifici, Marco Plebani.

Finally, there are many other colleagues inside and outside of the Department that I'd like to thank; I apologize in advance for any omission. Let me just mention the DB people: Alessandro Campi, Daniele Braga (eponymous?), Alessandro Raffio, Alessandra Savelli, and Davide Martinenghi; and Mara Tanelli and Stefano Zanero.

Still following the the "work-related acknowledgments only" policy [Fur03a, Fur03b], I'm not mentioning here my parents, (other) friends, and relatives whose fundamental support and love they have provided me with is incommensurate with everybody else's. Instead, I intend to thank them in more direct ways.

Contents

List of Figures

List of Tables

1. Introduction

The idea of analyzing computer programs and, more generally, computing systems with formal techniques dates back to the origin of computer science itself. Indeed, pioneers such as Turing [Tur49] and von Neumann [GvN63] have been the first to develop formal mathematical models of programs and to verify their properties with analytical techniques. This seemed a perfectly natural task to pursue: since its inception [Tur36, Soa96], computer science is rooted in mathematical logic and formal reasoning.

As the discipline evolved, however, the emphasis on formal analysis lessened, and the degree of its applicability grew much more slowly than the other aspects of the computing technology. This happened, on the one hand, because the most practical areas grew enough to stand independently on their own, losing the initial entanglement between theory and practice. On the other hand, the actual application of formal techniques requires a conspicuous amount of effort, and the effort involved is inherently hard to automate [Coo71]. Therefore, although it has been longly advocated by figures such as Dijkstra [Dij72] or Hoare [Hoa84], the practical application of formal techniques has often been neglected, outside the research community.

During the last decade or so, things have changed significantly [CW96]. First, the development of novel techniques for the formal analysis of programs and systems — the most notable being *model-checking* [CGP00] — together with the availability of increasingly large amounts of computing power as predicted by Moore's law [Moo65], have made formal verification more practically affordable, at least in some domains. Second, we finally begin to witness areas in which formal methods are slowly making their ways through industrial practice [CGR95], such as in hardware verification and — less frequently — in software development [BCLR04, Das06].

At the same time, formal methods are meeting new challenges [CW96, JOW06, Hoa03, HS06]. First of all, their scope of application is widening: formal analysis is no more limited to computer programs, but it addresses more generally computing *systems*. The notion of system encompasses a huge variety of diverse artifacts, where the computer is only a component

interacting with several other entities, embedded in a physical environment [HS06, SLMR05, Lee05]. Systems are typically large and complex; their components are usually heterogeneous in their nature and in the mathematical formalisms they require to be modeled and analyzed. Finally, *time* is a dimension which is often prominent in systems interacting with a physical environment [FMMR07]. In particular, a quantitative notion of time is required to deal with *real-time* systems, that is systems whose requirements involve timeliness properties [Liu00, HM96]. As a result, formal methods must deal with quantitative real-time properties, a demanding requisite in itself.

This thesis takes some of these challenges. From a broad perspective, the overall goal is to develop techniques and methods to improve the *scalability* of formal methods to large systems. The size of a model is a hurdle that is well-known to traditional software engineering [GJM02, Som06, Pre04]. Its solutions stem from the principles of *modularization* and *abstraction* [Par72]: a complex system is suitably divided into interacting parts (called "modules"), and the characteristics fundamental to each module (with respect to its interactions with the others) are retained, while the details of the internal behavior of the module are abstracted away. This thesis aims at extending these practices to the realm of formal analysis techniques and methods, in order to make it possible to apply them to large systems.

In the context of formal languages and verification, the principles of modularization and abstraction are often referred to with the term "compositionality". According to this terminology, the first part of this thesis deals with *compositional techniques* and *compositional methods*. In other words, it tackles the problem of scaling up the processes of formalization and verification of systems.

Since, as we hinted at above, large systems are often heterogeneous, different parts require different formal techniques and practices. Accordingly, the natural generalization of a compositional approach is in the direction of integrating modules of dissimilar attributes, and modeled with varied formalisms. In this respect, the second part of this thesis tackles the problem of *integration*: extending compositional techniques and methods to deal with heterogeneous systems.

Finally, this thesis focuses on *real-time* systems in developing the themes of compositionality and integration. This class of systems is of major importance, pervasive, and such that the scalability and heterogeneity challenges are most prominent. Heterogeneity manifests itself in real-time systems es-

pecially in the form of different *time models* required to describe different parts of a system. Therefore, compositionality and integration are defined in this thesis according to different time models, and in particular for both discrete time models and dense (and continuous) time models.

Our approach to real-time systems. Let us describe how the above goals are approached in this thesis. First of all, we exploit the *descriptive* approach to the formalization of systems. Therefore, both the system model and its properties are expressed with the same formal language. Since we focus on real-time aspects, our reference language is temporal logic [GHR94, Eme90, Pnu77] endowed with metric constructs [AH93, FMMR07, BMN00, FPR06].

The descriptive approach is effectively coupled with a *deductive* approach to verification, where proving that a specification satisfies a set of requirements amounts to the application of logic-deductive techniques. This is not a restriction *a priori*, as the framework that will be developed permits also different verification techniques. Finally, the emphasis of the work is not only on technical aspects, but also on methodological instances. In this respect, besides the viewpoint developed in this thesis, this work should be collocated within the broader horizon of software and system development engineering, some of whose fundamental issues we tackled elsewhere [SFR+06, FRS+06].

The reference formal language. Let us discuss our choice of reference metric temporal logic. First, we look for a very expressive language, that is capable of expressing extensive classes of properties. This is required to pursue in full the descriptive approach, where complex systems specifications must be expressible with the language. Second, in order to meet the methodological aspects required by scalability, the logic must possess suitable constructs to manage modular descriptions in a natural way, such as modules, genericity, visibility management, etc. Notice that these aspects are orthogonal to expressiveness: such constructs do not add to the expressiveness of the language, but only to its amenability to practical use. Third, the description of heterogeneous systems requires the language to be flexible enough to permit different semantics. In particular, we discussed our interest in developing common techniques for different time models. Finally, an adequate support of tools and a standard graphical notation are two very important requirements that are usually desirable of any formal

language, regardless of its intended usages.

The TRIO metric temporal logic [CCPC+99] fulfills satisfactorily these requirements, and it is therefore the reference formal language of this thesis. We remark, however, that most of the technical results drawn in this work may be extended to other formal languages of similar expressiveness, and the methods can be applicable in different contexts. In particular, the second part of the thesis focuses on a TRIO subset that is shown to be as expressive as MTL (according to its definition in [AH93]); therefore, the results of the second part apply also to MTL straightforwardly.

Plan of the thesis. This introduction is followed by Chapter 2, where the TRIO temporal logic language is presented in detail. The chapter serves as a reference for all subsequent uses of the language, and also demonstrates the use of the language on an example.

The core of the thesis comes after Chapter 2. It is composed of two parts, corresponding to the two main investigated aspects: Part I is about compositionality, and Part II is about integration.

In the first part, the notion of compositionality is defined and framed in Chapter 3. According to the general guidelines laid there, Chapter 4 presents several related works about compositional frameworks, and compares their approaches with ours. Chapter 5 introduces our compositional framework for the language TRIO. The framework is based on several compositional inference rules and on a general methodological approach. Both are introduced and discussed in Chapter 5. Afterward, Chapter 6 illustrates the compositional framework through two examples: one is a real-time variation of the classical dining philosophers problem [Dij71], the other is a peer-to-peer communication protocol.

The second part of the thesis is bridged to the first part by Chapter 7, where the integration problem is presented as a generalization of compositionality. More specifically, we deal with the integration between discrete time and continuous time, as it is also discussed in Chapter 7. Then, Chapter 8 presents a framework in which formulas written in a subset of the TRIO language and interpreted in discrete time or continuous time can be integrated within the same system. The example of a simple controlled system is introduced there to illustrate the ideas in practice. Chapter 9 discusses some aspects of the integration framework, in order to frame precisely its usability. In particular, the expressiveness of the subset of TRIO used in the integration framework is discussed, and a comparison with other

notions of discretization is detailed.

Finally, Chapter 10 concludes the thesis, by summarizing the main results and by discussing lines for future work.

Publications. Part of the compositional framework presented in Part I of the thesis has been published in preliminary form in [FRMM05a], and then in [FRMM07]. Other compositional rules are instead presented in the technical report [Fur06b]. A detailed analysis of the related works in compositionality, a subset of which is presented in Chapter 4, is the object of the technical report [Fur05].

The integration framework discussed in Part II has been published in [FR06]. Details omitted in [FR06] have been published in the technical report [FR05], while the other report [Fur06c] discusses the features of the integration framework.

2. The TRIO Metric Temporal Logic

TRIO (Tempo Reale ImplicitO[1]) [CCPC+99,GMM90,MSP94] is a general-purpose specification language suitable to describe real-time systems. At its core, it is a first-order linear temporal logic that supports a metric on time. This basic language is enriched with advanced modular features that are useful in writing specifications of complex systems. Indeed, it has been used in a number of industrial projects for for modeling and analyzing time-critical systems [CCPC+99,BCC+98,GLMZ96,BCC+95,MM97].

In presenting the language, we distinguish TRIO *in-the-small* — that is the bare first-order metric temporal language — from TRIO *in-the-large* — its modular features.

Let us add a terminological remark here. In the descriptive approach to modeling and analysis — the one we are pursuing in this thesis — a system is formally described by formally stating its salient properties as formulas of the language. According to a common software engineering paradigm [GJM02], we name *specification* the model of the system, that is the set of formulas describing its behavior. On the other hand, the putative properties of the system constitute its *requirements*.

2.1. TRIO in-the-Small

TRIO is a model-parametric temporal logic [MMG92]. This means that the semantics of its formulas can be defined for various temporal domains (also named "time models"). We denote the generic time domain as $\mathbb{T}$. In practice, it is customary to choose totally ordered metric sets as temporal domains, and more specifically numeric sets.

In this thesis, we are particularly interested in supporting both discrete and dense time models. Therefore, we will commonly adopt the integers $\mathbb{Z}$ as standard discrete time model, and the reals $\mathbb{R}$ as standard dense

[1] Italian for "Implicit Real-Time".

(and continuous) time model. However, other choices are possible. In particular, the mono-infinite time domain of the natural numbers $\mathbb{N}$ and the nonnegative reals $\mathbb{R}_{\geq 0}$ are very common choices in the literature on temporal logic. In the remainder, we will point out distinctions due to different choices in the time domains, whenever relevant.

2.1.1. Items

A TRIO specification is built out of a set of basic items. Each item is a primitive element, such as a predicate (or a function), representing the elementary model of some feature of the system. Items are defined by a domain they map their value to and, for functions, the domain of their arguments. For instance, a predicate is identified by a Boolean domain, whereas a function of the reals is defined through the domain $\mathbb{R}$ of its argument and that of its values (also $\mathbb{R}$).

Items can be time-dependent (TD) or time-independent (TI); the latter take values that are constant with respect to the time of evaluation, whereas the former have in general different values at different time instants. We define these notions more precisely when dealing with TRIO semantics below.

2.1.2. Syntax

TRIO is first of all a full-fledged first-order language, so it possesses all usual propositional and predicative connectives and quantifiers. It also fully includes arithmetic (and thus, in particular, equality). We do not give here a formal account of this part of the language, which is standard.

Concerning temporal aspects, TRIO defines a single modal metric operator named Dist. Thus $\text{Dist}(F, t)$ is a well-formed TRIO formula, where F is a time-dependent formula (namely, a predicate whose truth value is defined for every time instant), and t is an element of the time domain denoting a time distance. t can be positive, negative, or zero, denoting respectively a distance in the future, past, or the present instant. Intuitively, $\text{Dist}(F, t)$ denotes the fact that the formula F holds at t time units from the current time instant. Notice that in TRIO the current time instant is *implicit*, so it is not denoted by any syntactic element.

$\text{Dist}(F, t)$ is in turn a time-dependent formula, so it is possible to recursively combine the basic operator with propositional and predicative constructs, as well as arithmetic, to express complicated time relationships. In

particular, it is customary to define a number of *derived* temporal operators to denote commonly encountered temporal relationships. Table 2.1 lists a number of them.

Operator	Definition
$\text{Futr}(F,t)$	$t \geq 0 \wedge \text{Dist}(F,t)$
$\text{Past}(F,t)$	$t \geq 0 \wedge \text{Dist}(F,-t)$
$\text{Alw}(F)$	$\forall d : \text{Dist}(F,d)$
$\text{Som}(F)$	$\exists d : \text{Dist}(F,d)$
$\text{AlwF}(F)$	$\forall d > 0 : \text{Futr}(F,d)$
$\text{AlwP}(F)$	$\forall d > 0 : \text{Past}(F,d)$
$\text{SomF}(F)$	$\exists d > 0 : \text{Futr}(F,d)$
$\text{SomP}(F)$	$\exists d > 0 : \text{Past}(F,d)$
$\text{Lasts}(F,t)$	$\forall d \in (0,t) : \text{Futr}(F,d)$
$\text{Lasted}(F,t)$	$\forall d \in (0,t) : \text{Past}(F,d)$
$\text{WithinF}(F,t)$	$\exists d \in (0,t) : \text{Futr}(F,d)$
$\text{WithinP}(F,t)$	$\exists d \in (0,t) : \text{Past}(F,d)$
$\text{Within}(F,t)$	$\exists d \in (0,t) : \text{Past}(F,d) \vee \text{Futr}(F,d) \vee F$
$\text{Until}(F,G)$	$\exists d > 0 : \text{Lasts}(F,d) \wedge \text{Futr}(G,d)$
$\text{Since}(F,G)$	$\exists d > 0 : \text{Lasted}(F,d) \wedge \text{Past}(G,d)$
$\text{NowOn}(F)$	$\begin{cases} \exists d > 0 : \text{Lasts}(F,d) & \text{if } \mathbb{T} \text{ is dense} \\ \text{Futr}(F,1) & \text{if } \mathbb{T} \text{ is discrete} \end{cases}$
$\text{UpToNow}(F)$	$\begin{cases} \exists d > 0 : \text{Lasted}(F,d) & \text{if } \mathbb{T} \text{ is dense} \\ \text{Past}(F,1) & \text{if } \mathbb{T} \text{ is discrete} \end{cases}$
$\text{Becomes}(F)$	$\begin{cases} \text{UpToNow}(\neg F) \wedge \text{NowOn}(F) & \text{if } \mathbb{T} \text{ is dense} \\ \text{UpToNow}(\neg F) \wedge F & \text{if } \mathbb{T} \text{ is discrete} \end{cases}$

Table 2.1.: Some TRIO derived temporal operators

Inclusion/exclusion of endpoints. Table 2.1 also shows that the standard definition for TRIO temporal operators involving intervals takes open intervals, that is excludes both endpoints of the intervals. However, it is common to use variations of the temporal operators that specifically include one (or both) of the endpoints of the intervals of the definitions. These variations are denoted by adding subscripts i (for included) or e (for excluded). Table 2.2 lists some of these common variations, together with

their explicit definition.

2.1.3. Semantics

In order to define the semantics of TRIO formulas we have to introduce the notion of history (or behavior), which are the logic interpretations of the temporal logic formulas. Let $I_1, \ldots, I_n$ be the basic items of a TRIO specification. To each item I_i we associate a domain D_i, that is basically a type. The domains D can be arbitrarily complex, so as to represent functions and predicates. For simplicity, we refrain from detailing the representation of common items such as predicates and functions, and we refer the reader to standard definitions of mathematical logic [Men97]. We assume that a satisfaction relation $\models$ is suitably defined, so that for a basic item I and a value $d \in D$ in its corresponding definition domain, $d \models I$ denotes the fact that I is satisfied in the interpretation given by d.

Let us now define the notion of *behavior* (or *history*): it is a mapping (that is, a function) $b : \mathbb{T} \rightarrow \langle D_1 \times \cdots \times D_n \rangle$ from the time domain $\mathbb{T}$ to the composite set $\langle D_1 \times \cdots \times D_n \rangle$.

For any behavior b, time independent items have a constant value over the whole time axis. That is, for any time independent item I_i, and for any two time values $t_1, t_2 \in \mathbb{T}$, it is $b(t_1)|_{D_i} = b(t_2)|_{D_i}$, where $b(t)|_{D_i}$ denotes the projection of the value $b(t)$ to its ith component.

For a behavior b, a time value $t \in \mathbb{T}$, and a TRIO formula F, we write $b(t) \models_{\mathbb{T}} F$ to denote the fact that F holds (i.e., it is true) when t is taken as current time instant. The definition of the $\models_{\mathbb{T}}$ when F is a basic item is defined from the other satisfaction relation $\models$ introduced above. Namely, for a basic (time-dependent or time-independent) item I, associated to the domain D, the $\models$ relation is subsumed by the $\models_{\mathbb{T}}$ relation.

$$b(t) \models_{\mathbb{T}} I \quad \text{iff} \quad b(t)|_D \models I$$

Again, we avoid defining explicitly the relation $\models_{\mathbb{T}}$ for the basic propositional and predicative constructs, since this is routine [Men97]. Instead, let us define the semantics of the modal operators as follows. For any TRIO time-dependent formula F, behavior b, time instant $t \in \mathbb{T}$, and time value $t' \in \mathbb{T}$, we have:

$$b(t) \models_{\mathbb{T}} \mathrm{Dist}\big(F, t'\big) \quad \text{iff} \quad b(t + t') \models_{\mathbb{T}} F$$

We adopt the convention of assuming an implicit universal closure of all free variables, as well as of the (implicit) time variables, when interpreting

TRIO formulas. That is, for any TRIO formula $F(v_1, \ldots, v_m)$ with free variables $v_1, \ldots, v_m$, we define:

$$b \models_{\mathbb{T}} F \quad \text{iff} \quad \text{for all } t \in \mathbb{T} : b(t) \models_{\mathbb{T}} \forall v_1, \ldots, v_m : F(v_1, \ldots, v_m)$$

Finally, a set of TRIO formulas $F_1, \ldots, F_n$ constitutes a specification defining a set of behaviors $B = \{b_j\}$ satisfying *all* formulas of the specification: $b \in B$ iff $b \models_{\mathbb{T}} F_i$ for all $i = 1, \ldots, n$.

Conventions and simplifications. In the remainder, we will introduce implicit simplifications in presenting proofs involving TRIO formulas, whenever this does not sacrifice the clarity and precision of the exposition. More precisely, when proving some formula G from a specification given by the formulas $F_1, \ldots, F_n$, we implicitly assume to have taken: (1) a generic behavior b for the item and time domain under consideration, such that $b \models_{\mathbb{T}} F_i$ for all $i = 1, \ldots, n$; and (2) a generic implicit current time instant t, without mentioning them explicitly. When possible, we even assume the current time instant to be 0; notice that this is without loss of generality with a bi-infinite time domain such as $\mathbb{Z}$ and $\mathbb{R}$. For instance, we will use these simplifying conventions in the proof of the dining philosophers example, in Section 6.2.

We also assume the usual conventions on the precedence of logic operators; namely, in decreasing order of precedence, we have: $\neg$, $\wedge$ and $\vee$, $\forall$ and $\exists$, $\Rightarrow$, $\Leftrightarrow$.

2.1.4. States and Events

When describing the dynamics of complex systems, it is often useful to introduce items with a constrained behavior, that are suitable to model naturally classes of phenomena. These constrained items are also called *ontological constructs*. In particular, when dealing with *dense* time models, it is usually convenient to introduce two such constructs: states and event [GM01].

An *event* is a time-dependent predicate which is true only at isolated time instants; it is useful in modeling phenomena whose actual duration is negligible with respect to the overall dynamics of the system, so that they can be modeled as having a zero-time duration [GMM99]. For instance, the clicking of a switch is a phenomenon which can usually modeled as having zero-time duration. Formally, a time-dependent predicate E is an event

whenever it obeys the following formula [GM01] (recall that we assume a dense time domain):

$$\text{UpToNow}(\neg E) \wedge \text{NowOn}(\neg E)$$

A *state* is instead a time-dependent predicate which holds over non-empty time intervals. It is suitable to model phenomena which are never "instantaneous", but always have a non-null duration; for instance, a lamp being on or off is a phenomenon suitable modeled with a state, since it is reasonable to assume that it cannot switch from on to off and back to on again in zero time (under the assumption that the time unit is sufficiently fine-grained). Formally, a time-dependent predicate S is a state whenever it obeys the following formula [GM01] (recall that we assume a dense time domain):

$$\begin{aligned}(\text{UpToNow}(S) \wedge S \wedge \text{NowOn}(S)) \quad \vee \\ (\text{UpToNow}(\neg S) \wedge \neg S \wedge \text{NowOn}(\neg S)) \quad \vee \\ (\text{UpToNow}(\neg S) \wedge \text{NowOn}(S)) \quad \vee \\ (\text{UpToNow}(S) \wedge \text{NowOn}(\neg S))\end{aligned}$$

Notice that the value of a state on the edge of a transition is unspecified: it can be either the one on the left or the one on the right of the transition instant. We can further specialize a state to be *right-continuous* (resp. *left-continuous*) by requiring that the value at transition points is always equal to the one on the right (resp. on the left). Formally, S is a right-continuous state if it is a state and it obeys:

$$S \Leftrightarrow \text{NowOn}(S)$$

and it is a left-continuous state if it is a state and it obeys:

$$S \Leftrightarrow \text{UpToNow}(S)$$

2.1.5. Zenoness

When defining the semantics of TRIO formulas above, we made no assumptions about regularity of the behaviors a formula (or a whole specification) is interpreted on. In particular, we did not rule out behaviors where time distances between consecutive events become arbitrarily small

(i.e., infinitesimal). Note that this is possible only for dense time models, therefore the following discussion applies to such time models only.

Such behaviors are called *Zeno* behaviors [AL94]: they do not usually correspond to physically meaningful behavior, and they are a source of incompleteness when reasoning about derived properties of a system [GM01]. For instance, intuitive notions such as "the next changing point" of an item may be undefined for Zeno behaviors. As a result, most metric temporal logic languages adopt *a priori* restrictions on the behaviors on which formulas of the language are interpreted, ruling out Zenoness at the semantic level (see e.g. [AH92b]).

In TRIO, instead, one usually follows a different approach, detailed in [GM01]. First, we do not introduce any restriction *a priori* on the regularity of the behaviors. This allows one to state general results on formulas of the language in the most general way. Then, if Zenoness is really an issue in the specification of a system, one introduces suitable restrictions on the basic item of the specification so that they do not encompass Zeno behaviors. In general, it is sufficient to assume the following formula for a generic time-dependent item I to rule out its Zeno behaviors:

$$(\text{UpToNow}(I) \vee \text{UpToNow}(\neg I)) \quad \wedge \quad (\text{NowOn}(I) \vee \text{NowOn}(\neg I))$$

Notice that the definition of states and events subsumes this axioms, so, in particular, states and events cannot exhibit Zeno behaviors by construction. Similarly, we can introduce suitable restrictions to rule out Zeno behaviors for items mapping to denumerable and uncountable domains. For brevity, we do not present them here [GM01].

2.2. TRIO in-the-Large

To specify large and complex systems, and to support encapsulation, reuse and information hiding at the specification level, TRIO has the usual object-oriented constructs such as classes, inheritance and genericity (see e.g. [Mey97]).

In this thesis, we do not develop aspects depending on inheritance mechanisms, so we do not introduce details about how this feature is implemented in TRIO; we refer the interested reader to [MSP94,FMM+04]. Let us just note that TRIO's notion of inheritance is a very liberal one, so it gives the user a lot of freedom in reusing specification artifacts.

2.2.1. Classes and Modules

In TRIO, the basic encapsulation unit is the *class*. Classes can be simple or structured. Simple classes are collections of parameters, basic items, and formulas. Parameters introduces *genericity* in the class definition. Formulas, instead, constitute the formal description of the class, that is formalize the behavior of the class' items. Structured classes, in addition, contain instances of other classes, called *modules*.

An instance of a simple class is, in logical terms, a model of the class' formulas (i.e., the set of all behaviors satisfying the conjunction of all formulas), according to the semantics we have defined above. An instance of a structured class recursively contains instances of its composing modules.

Connections. Items of different modules can be *connected*: whenever two items are connected, they are implicitly defined as logically equivalent. If a more complicated semantics to represent connections in a system is needed, one may represent all the relevant information in an *ad hoc* class.

2.2.2. Types of Formulas

Each formula in a TRIO class can be "tagged" with one of these three attributes: *axiom*, *assumption* and *theorem*. When necessary, we will refer to these formulas as (TRIO) "axiom, assumption, theorem formulas", respectively. From a methodological point of view, axioms postulate the basic behavior of the system (its specification), assumptions express constraints we must validate[2] by means of other parts of the system (external to the current class) and theorems describe properties that are derived from other formulas (including requirements). Therefore, the verification of a specification would basically consist in proving theorems from axioms and assumptions, after validating the latter. In Section 5.4 we illustrate a compositional methodology that exploits these language features.

Occasionally, in the remainder we will also use the term "lemma" to denote an intermediate result for a theorem. Although this is not strictly a TRIO keyword, its role is basically the same as that of a "lesser" theorem.

[2] From a purely formal point of view, however, the validation of a formalized assumption consists exactly in a formal *proof*, from other formulas. Thus, here we use the verb "validate" only for methodological clarity.

2.2.3. Visibility and Interfaces

Each primitive item can be declared to be visible to other classes, or non-visible. Each formulas also has a notion of visibility, which is derived from the visibility of the items the formula predicates on. Namely, all formulas predicating on visible items *only* are considered as visible, while all other formulas are considered as non-visible outside the class.

Classes and their interfaces are synthetically represented with a graphical notation, where classes are pictured as boxes, visible items are denoted by lines crossing the class box, whereas non externally visible items stay entirely within the box boundaries. States are graphically denoted by thick lines, whereas events are denoted with a bullet placed where the line of the corresponding item touches (or crosses, if visible) the class' box.

For example, Figure 2.1 represents a simple class, while Figure 2.2 represents a structured class and the connections among items of its modules. Both will be described in the following Section 2.3.

2.2.4. Higher-Order TRIO and ArchiTRIO

In this thesis we comply with TRIO "standard" notation and semantics. More recently, however, a higher-order extension of the TRIO language — called *HOT* — has been introduced [FMM+04]. Higher-order constructs make it possible — among other things — to define more simply the semantics of modular and object-oriented features. In particular, higher-order types allows one to identify classes with *types*. This not only simplifies the formalization of modularity and inheritance rules, but also makes TRIO classes *first class* entities [GJ97]. Thus, the use of higher-order features is the most natural way to express complicated relationships between instances of classes.

Besides providing a more general formal framework, and exhancing the expressiveness of the language to a higher level, HOT has proved useful in defining formally the semantics of other extensions of the TRIO language. In particular, HOT provides a way to define the semantics of the *ArchiTRIO* language [PRM95]. ArchiTRIO exploits the UML graphical notation [UML05, UML04, EPLF03], to which it gives a defined formal semantics. Therefore, it combines a popular and intuitive notation — familiar to most software engineers, and not only [MFR04] — with the rigor and unambiguity of a logic notation. ArchiTRIO has been designed to deal especially with the formal specification of software and system architectures,

although its actual scope of applicability is much wider. We remark that most of the results of this thesis can be readily adapted to ArchiTRIO notation and concepts.

2.3. The Dining Philosophers Example

In order to illustrate TRIO's features, and to familiarize with its syntax and semantics, we introduce an illustrative example based on the specification of the *dining philosophers problem* [Dij71]. This classic problem considers a set of philosophers sitting around a table, competing for the acquisition of some shared resources, namely one fork for each pair of philosophers. When one succeeds in acquiring the forks on both sides, he/she can eat for some time, after which the forks are released for others to use. The non-eating activity is usually referred to as "thinking". In the traditional formulation of the problem, the goal is to prove the absence of deadlocks, which may arise in acquiring the shared forks. In our specification, instead, the goal is to prove a real-time, global, non-starvation property for all the philosophers, i.e., the fact that, under suitable assumptions, every philosopher eats within a given time.

This will be pursued later, through the compositional methodology to be introduced in Section 5.4. In this section, we just give the basic specification of the dining philosophers problem. In order to familiarize the reader with TRIO syntax, we also provide the full TRIO code of the dining philosopher in Appendix A.

2.3.1. Basic Items and Classes

The basic class in our formalization is a **philosopher** class. Throughout the example, we assume time to be continuous.

The basic items of the class are the event item start for system initialization, and the events take(s) and release(s), with $s \in \{\mathrm{l}, \mathrm{r}\}$, indicating, for each philosopher, the action of taking or releasing the left or right fork. Other items are the state eating, which is true exactly when the philosopher is eating, and the states holding(s) and available(s), meaning that the philosopher is holding a given fork or that the fork is available (i.e., not held by the adjacent philosopher). For notational convenience, we also introduce the states thinking and hungry, representing the philosopher not eating, and being ready to eat again (after a sufficiently long thinking period), respectively.

The **philosopher** class is parametric with respect to three constants t_e, T_e, T_t. They denote, respectively, the minimum eating time, the maximum eating time and the thinking time after an eating session, before becoming hungry again. Obviously, we assume $T_e > t_e > 0$.

2.3.2. Philosopher Specification

In order to demonstrate the use of TRIO as a specification formalism, we now provide a set of formulas that formally describe the basic behavior of a philosopher. In particular, we provide a set of informally stated basic properties of the behavior of each philosopher, and we encode them as TRIO axioms of the **philosopher** class. Through the example, the reader can familiarize with the syntax and semantics of TRIO operators.

We postulate that each philosopher always takes and releases both forks simultaneously (axiom **holding_synch**); consequently, if only one fork is available, the philosopher waits till the other fork becomes available as well.

Axiom 1 (**philosopher.holding_synch**). $\mathsf{holding}(l) \Leftrightarrow \mathsf{holding}(r)$

A philosopher is "hungry" when he/she has not eaten for a period of at least T_t: this duration is formalized through the Lasted operators, as in the following axiom.

Axiom 2 (**philosopher.hungry**). $\text{Lasted}(\neg\mathsf{eating}, T_t) \Leftrightarrow \mathsf{hungry}$

If the philosopher is hungry and if the forks are available, two situations are possible: either he/she takes both forks, or nondeterministically one of his/her neighbors takes a fork at that very time, so that the fork is not available anymore and the philosopher "loses his/her turn". This is formalized by axiom **acquire**. In particular, in order to express the availability of the forks in the immediate past (from the current instant) we use the UpToNow operator. Similarly, the NowOn operator formalizes the non availability in the immediate future.

Axiom 3 (**philosopher.acquire**).
$\mathsf{hungry} \wedge \text{UpToNow}(\mathsf{available}(l) \wedge \mathsf{available}(r))$
$\Rightarrow\ (\mathsf{take}(l) \wedge \mathsf{take}(r)) \vee \text{NowOn}(\neg\mathsf{available}(l) \vee \neg\mathsf{available}(r))$

Axioms **eat_till_release** and **eating_duration** state that, when a philosopher succeeds in acquiring both forks (i.e., $\mathsf{holding}(l) \wedge \mathsf{holding}(r)$ becomes true), he/she eats for a time duration of more than t_e and less than T_e time units, after which he/she releases both forks.

Axiom 4 (**philosopher.eat_till_release**).
$\text{Becomes}(\mathsf{holding}(\text{l}) \wedge \mathsf{holding}(\text{r}))$
$\Rightarrow \ (\exists t > \text{t}_\text{e} : \text{Lasts}(\mathsf{eating}, t) \wedge \text{Futr}(\mathsf{release}(\text{l}) \wedge \mathsf{release}(\text{r}), t))$

Axiom 5 (**philosopher.eating_duration**). [3]
$\text{Lasted}(\mathsf{eating}, t) \Rightarrow t < \text{T}_\text{e}$

Whenever a philosopher holds both forks we consider him/her eating. This corresponds to predicate eating being true, and is formalized by axiom **eating_def**.

Axiom 6 (**philosopher.eating_def**). $\mathsf{holding}(\text{l}) \wedge \mathsf{holding}(\text{r}) \Leftrightarrow \mathsf{eating}$

The state eating is false whenever the philosopher is "thinking".

Axiom 7 (**philosopher.thinking_def**). $\mathsf{thinking} \Leftrightarrow \neg\mathsf{eating}$

A thinking session (i.e., one in which the philosopher does not hold both forks) which has just begun lasts at least T_t time units (axiom **thinking_duration**).

Axiom 8 (**philosopher.thinking_duration**).
$\text{Becomes}(\mathsf{thinking}) \Rightarrow \text{Lasts}(\mathsf{thinking}, \text{T}_\text{t})$

Axioms **taking** and **putting** describe the consequences of a take and release action, respectively. More precisely, a fork s can be taken when (up to now) it is available and not already held; in this case, the state $\mathsf{holding}(s)$ stays true until a release for the same fork is issued. On the other hand, a fork s can be released when (up to now) it is held; in this case, the state $\mathsf{holding}(s)$ stays false until a take for the same fork is issued.

Axiom 9 (**philosopher.taking**).
$\mathsf{take}(s) \wedge \text{UpToNow}(\neg\mathsf{holding}(s)) \wedge \text{UpToNow}(\mathsf{available}(s))$
$\Rightarrow \ \text{Until}(\mathsf{holding}(s), \mathsf{release}(s))$

Axiom 10 (**philosopher.putting**).
$\mathsf{release}(s) \wedge \text{UpToNow}(\mathsf{holding}(s)) \ \Rightarrow \ \text{Until}(\neg\mathsf{holding}(s), \mathsf{take}(s))$

[3] Recall that free variables (i.e., t) are implicitly universally quantified at the outermost level.

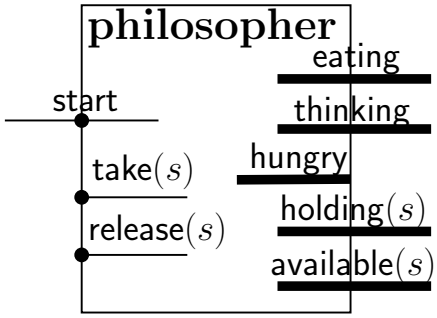

Figure 2.1.: Interface of the **philosopher** class

2.3.3. Interface and Graphical Representation

Figure 2.1 illustrates the items and interface of the **philosopher** class. Thus, for instance, holding(s) and eating are both visible items; therefore axiom **eating_def** is also visible. eating has to be declared as visible because, even if it is not used directly in the connections between philosophers, the global non-starvation property that we are going to state, in Section 6.2.4, predicates on it. The same holds for the start event, whereas the thinking state is visible since a theorem predicating on it will be used in proving the assumptions of other classes. Notice that TRIO's graphical notation does not represent any timing information about the class, which is instead represented entirely by formulas.

We compose $N \geq 2$ instances of the **philosopher** class into the new composite class **dining_N**. The N modules of the **philosopher** class are instantiated in an array **Philosophers** indexed by the range $[0..N-1]$. The modules are connected so that the **available** item of each philosopher corresponds to the negation of the **holding** item of the philosopher on his/her left/right, as pictured in Figure 2.2.

In Section 6.2, a global property of this set of philosophers will be proved, namely that each philosopher eats regularly for a certain amount of time. Note that all the instances in the array have the same values for the parameters $\mathrm{t_e}, \mathrm{T_e}, \mathrm{T_t}$. This ensures that the timed behavior of the various philosophers is the same, thus permitting to prove the global property. We leave the analysis of the impact of relaxing such constraint to future work.

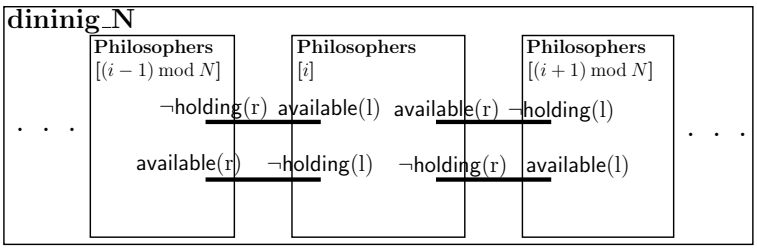

Figure 2.2.: Interface of the **dining_N** class

2.4. TRIO Tools and Implementations

Although tool support and implementation is not the focus of this thesis, it is undeniable that an adequate tool support is a key feature that a formal language must possess in order to be usable — and used — in practice, on large and complex systems. Therefore, in this section we give a brief overview of the available tool support for the TRIO language.

We used TRIO tools to actually carry out various examples (of different sizes and complexities) of specification and verification; however, we present them only in human-readable form in this thesis. This shows that the methods and techniques we develop in this thesis are suitable to be used with TRIO tools. Then, tool development is an orthogonal, but reconcilable, concern with respect to those that are the content of this thesis.

The purposes of tools with respect to formal methods usage are various. Here, we distinguish three classes of usages, although this classification is not meant to be exhaustive or rigid. On the other hand, in [SFR+06] we provide a framework where the precise roles of verification and validation can be more clearly defined.

Verification tools. The purpose of verification is to prove that a specification entails some set of requirements. In other words, given a specification model S and a statement of the requirements R, the goal of verification is to prove the theorem $S \vdash R$. There is a large variety of verification techniques, and correspondly a large variety of verification tools. In particular, we have both automatic and semi-automatic tools; the latter require human guidance, whereas the former are fully algorithmic.

Validation and testing tools. Their purpose is to provide some confidence in the correctness of the specification by exploring some of its be-

haviors (rather than exhaustively deriving properties common to all behaviors entailed by the specification, as in verification tools). This can be achieved, for instance, by extracting test cases from a formal specification, by executing the specification itself, etc.

Integrated Development Environment (IDE) tools. These are integrated editors and graphical environments that support the user in activities such as: writing formulas respecting the syntax of the formal language, extracting a textual formal description from a graphical one, making elementary consistency checks among parts of a specification, etc.

With respect to the goals of this thesis we are particularly interested in formal verification tools; so, the next section presents some of those available for the TRIO language. Next, we also briefly discuss other TRIO tools.

2.4.1. Verification Tools

In the design of a specification language there is a typical trade-off between expressiveness and complexity of verification. On the one hand, one desires the most expressive language, in order to be able to express extensive classes of complex properties. On the other hand, the complexity of verification typically increases with expressiveness: the more expressive the language, the more complex the verification procedures for that language.

In particular, TRIO is a very expressive language, so much that it is also undecidable. This means that the general verification problem is not fully automatable for TRIO formulas: it is impossible to design an algorithm capable of correctly determining the truth of a TRIO formula from a set of TRIO axioms, which is both fully automatic and works for any TRIO formula [GM93,HMU00,Sip05]. As a consequences, there are two ways out of this conundrum: either we restrict ourselves to a proper subset of the TRIO language, one which is fully decidable, or we give up full automation.

The TVS Tool

The latter choice has been pursued with the TRIO Verification System (TVS, also called TRIO/PVS). It consists of an encoding of all the TRIO language in the PVS higher-order theorem prover [ORS92], coupled with a set of proof strategies (i.e., scripts in PVS jargon) that work with the PVS

prover and aid to simplify and — to some extent — automate proofs of TRIO formulas encoded in the PVS language.

TVS is particular suited to deal with the peculiar difficulties of dense time, for which it exploits PVS capabilities of manipulating properties of simple arithmetic over the reals. As we already discussed the tool cannot be fully automatic, and indeed it requires substantial human intervention. On the other hand, it provides a relatively complete set of inference rules [San92] which is used in practice to guarantee and verify every single step in a system verification, still automating the most elementary and repetitive steps.

The TVS system is described in [GM01,Fif98] for what concerns in-the-small TRIO, while the encoding of the modular extensions is documented in [Fur03b, FR04]. We extensively used it in machine-checking proofs of the various examples we provide in this thesis.

TRIO2ProMeLa and Other Fully Automatic Tools

Restricting to a proper subset of the TRIO language allows to achieve decidability, and thus to provide fully automated tools.

TRIO2ProMeLa is based on a TRIO subset where all items vary over finite domain, the only infinite domain being time, which is taken to be the naturals $\mathbb{N}$; moreover, there is a limited support for TRIO modular features. Under these restrictions, it is possible to encode a TRIO specification as a set of ProMeLa processes [Hol03]. ProMeLa is the input language of the SPIN linear-time model checker [Hol03]. Therefore, one can rely on the SPIN tool to verify TRIO properties automatically.

The TRIO2ProMeLa tool is documented in [MPSS03, PSSM03]. We notice that the language subset used by TRIO2ProMeLa is close in expressiveness to $^{\mathbb{R}}_{\mathbb{Z}}$TRIO, a subset of the TRIO language that we will define and use in Chapter 8 of this thesis, when defining a framework for discrete- and continuous-time integration. Indeed, the example of Section 8.4 has been formally verified with TRIO2ProMeLa.

Zat. The Zat tool is based on translating a purely propositional subset of the TRIO language to a satisfiability (SAT) solver. The translation exploits bounded model-checking techniques [BCC+99, BCCZ99] to verify a TRIO specification over finite discrete time, or over periodic behaviors

over infinite discrete time (i.e., the naturals).[4]

2.4.2. Other Tools

TRIDENT (TRIO Development EnviroNmenT) is a IDE for the TRIO language that supports writing modular TRIO specification. It is developed as an Eclipse [Ecl] plug-in, and in its future developments it will serve as an integrated interface to all other TRIO tools, as well as the UML features of ArchiTRIO [CPRS06].

Other TRIO tools exist for tasks such as test-case generation [MMM95, SMM00] and and history checking [FM94]. We refer to [TRI] for more information about these tools.

[4] There is no official Zat documentation yet, but the interested reader can refer to [TRI06].

Operator	Definition
$\mathrm{AlwF_e}(F)$	$\forall d > 0 : \mathrm{Futr}(F, d)$
$\mathrm{AlwF_i}(F)$	$\forall d \geq 0 : \mathrm{Futr}(F, d)$
$\mathrm{AlwP_e}(F)$	$\forall d > 0 : \mathrm{Past}(F, d)$
$\mathrm{AlwP_i}(F)$	$\forall d \geq 0 : \mathrm{Past}(F, d)$
$\mathrm{SomF_e}(F)$	$\exists d > 0 : \mathrm{Futr}(F, d)$
$\mathrm{SomF_i}(F)$	$\exists d \geq 0 : \mathrm{Futr}(F, d)$
$\mathrm{SomP_e}(F)$	$\exists d > 0 : \mathrm{Past}(F, d)$
$\mathrm{SomP_i}(F)$	$\exists d \geq 0 : \mathrm{Past}(F, d)$
$\mathrm{Lasts_{ee}}(F, t)$	$\forall d \in (0, t) : \mathrm{Futr}(F, d)$
$\mathrm{Lasts_{ei}}(F, t)$	$\forall d \in (0, t] : \mathrm{Futr}(F, d)$
$\mathrm{Lasts_{ie}}(F, t)$	$\forall d \in [0, t) : \mathrm{Futr}(F, d)$
$\mathrm{Lasts_{ii}}(F, t)$	$\forall d \in [0, t] : \mathrm{Futr}(F, d)$
$\mathrm{Lasted_{ee}}(F, t)$	$\forall d \in (0, t) : \mathrm{Past}(F, d)$
$\mathrm{Lasted_{ei}}(F, t)$	$\forall d \in (0, t] : \mathrm{Past}(F, d)$
$\mathrm{Lasted_{ie}}(F, t)$	$\forall d \in [0, t) : \mathrm{Past}(F, d)$
$\mathrm{Lasted_{ii}}(F, t)$	$\forall d \in [0, t] : \mathrm{Past}(F, d)$
$\mathrm{WithinF_{ee}}(F, t)$	$\exists d \in (0, t) : \mathrm{Futr}(F, d)$
$\mathrm{WithinF_{ei}}(F, t)$	$\exists d \in (0, t] : \mathrm{Futr}(F, d)$
$\mathrm{WithinF_{ie}}(F, t)$	$\exists d \in [0, t) : \mathrm{Futr}(F, d)$
$\mathrm{WithinF_{ii}}(F, t)$	$\exists d \in [0, t] : \mathrm{Futr}(F, d)$
$\mathrm{WithinP_{ee}}(F, t)$	$\exists d \in (0, t) : \mathrm{Past}(F, d)$
$\mathrm{WithinP_{ei}}(F, t)$	$\exists d \in (0, t] : \mathrm{Past}(F, d)$
$\mathrm{WithinP_{ie}}(F, t)$	$\exists d \in [0, t) : \mathrm{Past}(F, d)$
$\mathrm{WithinP_{ii}}(F, t)$	$\exists d \in [0, t] : \mathrm{Past}(F, d)$
$\mathrm{Within_{ee}}(F, t)$	$\exists d \in (0, t) : \mathrm{Past}(F, d) \vee \mathrm{Futr}(F, d) \vee F$
$\mathrm{Within_{ii}}(F, t)$	$\exists d \in [0, t] : \mathrm{Past}(F, d) \vee \mathrm{Futr}(F, d) \vee F$

Table 2.2.: Some included/excluded variations of TRIO temporal operators

Part I.

Compositionality

3. Compositionality and Compositional Inference Rules

As we mentioned in the Introduction, one drawback often attributed to formal methods is that they do not "scale up", i.e., when the system grows in complexity, formal methods are too cumbersome and unwieldy to be used effectively. A natural solution to this problem is to apply well-known software engineering principles such as modularity and separation of concerns not only to design, but also to the verification of formal models.

Compositional techniques can help in this regard: at their core, they consist in the application of a compositional inference rule which allows one to infer facts about a system built out of the composition of some modules from other — usually more elementary — facts about the individual modules in isolation. There is a certain amount of work available in the literature about compositional inference rules; we review the most significant works in Section 4. The goal of this Part is to develop a practical approach to compositionality, according to the guidelines that we are laying down henceforth.

3.1. Defining Compositionality

Let us start by giving a definition to the object of our study: *compositionality*.

The term *compositionality* and the idea of composition encompass a large variety of methods and techniques that aim at describing how to compose smaller systems to form larger systems, and how to manage the resulting complexity of the overall system. Clearly, this broad objective can be considered from several, disparate, very different perspectives.

Let us first attempt a very general (and hence necessarily, to some extent, vague) definition of compositionality. In lay terms, the Merriam-Webster Dictionary of English [MW98] defines the verb *compose* as "to form by putting together". This basic definition encloses the idea at the root of

compositionality: forming a system by putting together smaller subsystems.

A more technical definition reflecting this idea is that usually adopted in the philosophy of language, as reported by Janssen [Jan98]:

> "The meaning of a compound expression is a function of the meanings of its parts and of the rule by which the parts are combined"

This compositionality principle is also called *Frege's principle*, from the name of the author who first formulated the principle [Fre23].

This very same idea can be adapted to program verification: de Roever et al. [dRdBH$^+$01, pg. 48] define compositionality as:

> "That a program meets its specification should be verified on the basis of specifications of its constituent components only, without additional need for information about the interior construction of those components"

Therefore, in compositional verification we aim at verifying the global specification of a system by *composing* the specifications which are local to the various components of the system.

Adopting a rather liberal — but still technical — definition, compositionality consists in bringing the well-known engineering (and software engineering in particular) practices of modularization and abstraction from the specification to the verification domain. Hence, in this thesis we adopt the following general definition of compositionality:

> "The techniques and methods that permit the modularization of the verification process of a large system"

Notice that the above definition prescribes two main areas of investigation: *techniques* and *methods*.

- The *technical* aspects of compositionality consist in the study of the rules of composition for the chosen formal language; in particular, we are interested in temporal logics, and real-time extensions of them.

  Even more concretely, studying the technical aspects of compositionality means formulating a number of *compositional inference rules*. These are logic inference rule whose hypotheses state some facts about modules of the system in isolation, and whose conclusions state some facts about the composite system working as a whole.

We remark that the technical aspect is the aspect that has been more deeply studied in the literature, in particular in the form of compositional inference rules, as we will discuss in Chapter 4.

- The *methodological* aspects of compositionality consist in the study of how compositional techniques can be successfully applied *in practice* to (models of) modular systems.

  Therefore, a compositional method should suggest how to organize the local specifications of each module so that they can be composed through a compositional inference rule. Moreover, it should also describe a general "attitude" in formalizing and reasoning about composite systems, one which helps in managing the complexity of formalizing systems composed of several modules, and of the resulting correctness proofs.

  We will develop methodological aspects of compositionality in the following Chapter 5, and especially in Section 5.4.

Finally, let us remark that in this thesis we focus on the widespread *rely/guarantee* paradigm of compositionality. This is a natural way of specifying the module of a system whose behavior depends, in general, on the behavior of its environment. In a nutshell, a rely/guarantee specification expresses the guaranteed behavior of a module, relying on an assumption about the behavior of the module's environment. In particular, the module does not guarantee any behavior if its environment does not respect the assumption about it. In Chapter 4, when reviewing related works on compositionality, we will briefly mention compositional works that do not use the rely/guarantee paradigm. However, the rely/guarantee paradigm is a general and proper one for specifying the behavior of modules to be composed; therefore, it is without practical loss of generality that we focus on this paradigm in this thesis.

3.2. Dimensions of the Compositional Problem

In order to describe the approach to compositionality developed in this thesis, we outline what are the major dimensions according to which a compositional framework can be developed.

First of all there are some technical dimensions that are related to issues such as the choice of the language in which to develop the framework,

the nature of time (discrete or dense, linear or branching), the semantics given to composition, etc. Let us postpone the analysis of these aspects to Chapter 4, where they will be presented with reference to related works in the literature. In fact, although considering these aspects cannot obviously be avoided, we claim that they are mostly determined by the choice of the reference formal language to be used in the analysis. Under this respect, the framework we present in this thesis aims at generality; thus it introduces as few semantic assumptions as possible and uses an expressive and straightforward formalism.

On the contrary, let us now focus on the most important features that transcend the technical choices and characterize a compositional framework. For each feature, let us briefly point out what are the most common choices in the current literature, and how our framework differs. Doing so, we lay out the general guidelines for our approach to compositionality.

- The first aspect is the "compose vs. decompose". In the literature, we find both *decompositional* and *compositional* approaches. Decompositional approaches are basically refinement techniques where a single module is refined into the composition of several smaller modules; compositional techniques ensure that the composition of the refined modules retains all (or some) properties of the original single module. Compositional approaches, instead, start from a set of individual modules and provide ways of inferring properties about the composition of these individual modules into a larger system.

  In this thesis, we are mostly interested in the *compositional* approach. First, a decompositional approach would by definition involve refinement techniques, and thus aspects that are simply out of the scope of this work, as they would require a different framing and a different focus. Second, compositionality allows one to *reuse* the specifications of single modules in different systems that instantiate the modules. Therefore, we argue that compositionality is mostly useful in such contexts, where the possible additional burden introduced by the construction of a compositional specification [Lam98] is then amortized by reusing the same specification in multiple proofs for multiple different systems.

- Most of the literature deals with the *technical* aspects of compositionality, whereas *methodological* aspects are often neglected.

In this thesis we try to develop not just some compositional inference rules, but also a general methodological approach to the issue of compositional verification. We already discussed in the Introduction how the traditional practices of modularization, abstraction, and information hiding can be successfully applied also to the formal specification and verification of modular systems. This is integrated with a suitable notation such as TRIO, which gives to the user suitable constructs to implement these practices.

- Most works in the literature provide one unique compositional inference rule, with the tacit assumption that all facts about the composition of modules should be inferred using the rule. This is often not very practical: the verification process is often a complex task that requires much flexibility on how it should be carried out. Therefore, constraining the process to some prescribed "recipe" may lead to needless complications.

  On the contrary, we provide *several different rules*, we give a possible taxonomy according to which different rules can be categorized, and we discuss how the dimensions of the taxonomy mirror aspects of real systems. Our goal is not only to provide a variety of rules among which the user can choose, but also — and most importantly — to illustrate the attitude to reverse the usual approach to applying compositionality. Rather than having a rule (or a set of rules) and describing the system in a way which is amenable to the use of the rule, we suggest to first formalize the system according to the general modularization methods we discussed in the previous point. Then, after the system is formalized, one tries to find a suitable rule that fits the system in question; possibly, the user should also *develop* a new rule, based on the "lessons" learned from the existing rules and the formalization of the system, to suitably fit the system itself and its verification goals. Therefore, our catalog of compositional rules is also meant to provide an illustration on how compositional rules can be built and modified according to one's needs. In Section 6.3 we will provide an example of how this can be done in practice.

- Since, as we discussed in the previous point, one single rule is often assumed to be used to handle all possible cases, its *completeness* is usually regarded as an important feature [NT00, Fur05]. However, as we see in Chapter 5, compositional rules are complete in a *trivial*

way, with respect to compositionality. This means that there exist global properties which — although provable with the rule in question thanks to completeness — cannot be usefully "split" among properties local to different modules, but require a trivial partitioning of the system where one module behaves as the whole system.

Therefore, we claim that completeness is not an indispensable feature of a compositional rule. More specifically, completeness must not be traded off with simplicity of a compositional rule or with its practical applicability. Compositional reasoning cannot ease all problems of modular verification: it is important that it solves effectively some of them, those likely to happen in practice. Therefore, in this paper we develop both complete and incomplete rules with equal interest, and we try to understand what features may render a rule (in)complete.

- Most, if not all, of the compositional rules in the literature are *circular*, i.e., informally, the property of some module references circularly (in general, indirectly) to itself. Circular reasoning is clearly unsound, in general, so developing circular rules is a much more challenging task than developing non-circular ones.

  Nonetheless, we claim that circularity is not always needed in practice, and indeed some compositional reasoning can be carried out with non-compositional rules. Formulating non-circular inference rules is technically not very challenging, as they are simple consequences of the basic logic rules of conjunction and implication [Lam98]. Nonetheless, we develop some of them, in Section 5.1, together with circular ones. In fact, we believe that the technical challenges involved in developing the latter rules should not make us favor them over non-circular rules. Realizing what is the role of non-circular rules in practice should help framing the compositionality problem from the right perspective, and with the right objectives, that is practical usefulness rather than purely technical appeal.

We conclude this introduction to compositionality by stressing that we advocate a "lightweight" approach to compositionality. The approach is driven by the specifics of the system we are considering. Method comes first, as it gives general guidelines on how the system should be specified and organized. Then, according to the peculiar verification needs that arise, one should be able to pick a suitable technical solution to it, possibly developing new ones according to the problem at hand.

3.3. Defining Compositional Inference Rules

This section defines the notion of inference rule in general, and of compositional inference rule in particular. Then, it introduces some dimensions along which a variety of compositional inference rules can be characterized; in doing so, it tries to link the dimensions of the taxonomy to features of "real" systems they mirror.

3.3.1. Definition of (Compositional) Inference Rule

Inference Rules

Inference rules. Let us start by defining what is an inference rule, generally speaking. An *inference rule* [Men97] is defined by a relation between pairs of sets of formulas: $\Pi = \{\pi_1, \ldots, \pi_m\}$ and $\Xi = \{\xi_1, \ldots, \xi_n\}$ and usually denoted by:

$$\frac{\Pi = \{\pi_1, \ldots, \pi_m\}}{\Xi = \{\xi_1, \ldots, \xi_n\}}$$

Elements of Π are called the *premises* (or *antecedents*, or *hypotheses*) of the inference rule, while elements of Ξ are called the *conclusions* (or *consequents*, or *theses*). Π and Ξ are usually thought of as ordered sets, as our notation suggests.

Intuitively, whenever a set of premises $\widetilde{\Pi}$ and a set of conclusions $\widetilde{\Xi}$ belong to the relation defined by the inference rule, it means that the truth of $\widetilde{\Pi}$ entails the truth of $\widetilde{\Xi}$.

Soundness and completeness. An inference rule is *sound* (or *correct*) if: for any pair $\widetilde{\Pi}, \widetilde{\Xi}$ that belongs to the relation defined by the rule, if the premise $\widetilde{\Pi}$ is true, then the conclusion $\widetilde{\Xi}$ is also true.

We usually only care about sound inference rules, since the intuitive purpose behind defining and using an inference rule is exactly that of deducing true facts from other known true facts. Therefore, if we establish a proposition of the following form:

Proposition 3.3.1 (Soundness of an inference rule)**.** *If:*

1. π_1
2. π_2

   $\vdots$

m. π_m

then:

1. ξ_1

2. ξ_2

$\vdots$

n. ξ_n

we have in practice proved the soundness of the corresponding inference rule.

On the other hand, an inference rule is *complete* if: for any conclusion $\widetilde{\Xi}$ which is true, there is some premise $\widetilde{\Pi}$ such that the pair $\widetilde{\Pi}, \widetilde{\Xi}$ belongs to the relation defined by the rule, and $\widetilde{\Pi}$ is true. In other words, a complete rule allows the proof of any true fact.

If our reference logic language is an intrinsically incomplete one — as it is the case with TRIO, which includes arithmetic — then we cannot have a fully complete inference rule, that is one which is complete for any formula in the language. In such cases, when we speak about completeness we actually mean *relative completeness* [Coo78], that is, in a nutshell, completeness with respect to an oracle for the truth of arithmetic formulas.

While soundness is an indispensable property for an inference rule, completeness (even relative one) is a feature which is not strictly required, but may be desirable in some contexts.

Compositional Inference Rules

Let us now focus on *compositional* inference rules. The full practical significance of these rules will become apparent after presenting a methodology that exploits them; this will be the object of Section 5.4. However, while introducing the technicalities necessary to discuss compositional inference rules, in this section we also try to illustrate the meaning of the inference rules, and their overall rationale.

Generally speaking, the goal of a compositional inference rule is to provide a way to prove some facts about the *composition* of some modules, from other known facts about each of the modules. Let us assume that we are dealing with systems made of some $N \geq 2$ modules.

Local rely/guarantee specifications. The starting point for the application of a compositional inference rule is given by a set of *local specifications*, one for each of the modules in the system. This means that we have some formalization of the elementary significant behavior of each module, in terms of a formula.

In general, the local specifications could be any kind of formula. In practice, a very natural way to specify a module is according to the *rely/guarantee paradigm*.[1] With the rely/guarantee paradigm one makes some assumptions on the behavior of the *environment* of the module under specification, and he/she links these assumptions to the guaranteed behavior of the module itself: the module behaves as expected only if its environment does the same.

In order to formalize this in the inference rules, we associate to each module $1 \leq i \leq N$ two formulas: E_i and M_i. E_i is called the *local assumption* of module i, while M_i is called the *local guarantee* of module i. Therefore, the local specification of module i consists of some formula linking E_i to M_i.

In the simplest case, this link consists in logical implication: $E_i \Rightarrow M_i$ is a local specification describing a module that respects formula M_i as long as its environment respects formula E_i. For instance, let us consider a simple adder module which takes an integer a and returns it increased by 2 units: $b = a + 2$. A rely/guarantee local specification for such a module could be the following: if a is an even number, then so is b.

More generally — and in particular when dealing with the description of timing behavior — the link between assumption and guarantee is defined through a binary operator which we generically denote as $\succ$. We call it *compositional operator*. So, implication is a simple example of compositional operator, but in general different inference rules use different compositional operators. We will provide several examples of compositional operators in Chapter 5.

Global specifications. As we said, the goal of a compositional inference rule is to infer some facts about the composition of modules. In the most general case, the N modules are composed into an open system, which can therefore also be characterized by a global rely/guarantee specification. The global rely/guarantee specification can be of the same form of the local specifications, or of a different form.

[1] Other paradigms for the description of modules will be presented in Section 4.3.

Anyway, we characterize it through two formulas called *global assumption* and *global guarantee*, denoted as E and M, respectively. As it is simple to understand, they play similar roles as the local formulas, but with reference to the whole system: E formalizes assumptions about the expected behavior of the system's environment, and M formalizes the guaranteed behavior of the composite system.

Parts of a compositional inference rule. Let us now illustrate the typical components of a compositional inference rules, through a simple example.

Recall the module "adding 2" described above; let us introduce a predicate $\mathsf{eve}(k)$ to denote the fact that k is even. Thus, the rely/guarantee specification of this module can take the form $\mathsf{eve}(a) \Rightarrow \mathsf{eve}(b)$. Let us now connect two instances $1, 2$ of this module such that the input of module 1 is the output of module 2, and *vice versa*. The rely/guarantee specification of module 2 would then be $\mathsf{eve}(b) \Rightarrow \mathsf{eve}(a)$. We now illustrate the components of compositional inference rules through this simple system.

Global specification. Let us start from the conclusion of the inference rule. In a nutshell, it consists of the *global specification*: $E \succ M$. In other words, the overall goal of the inference rule is to deduce the truth of the global specification.

For our simple example, let us assume that the global guarantee is $\mathsf{eve}(a) \wedge \mathsf{eve}(b)$, that is both values exchanged by the two modules are even numbers. Since the system is a closed one, we need not care about a global assumption; equivalently, we assume E to be identically true.

Local specifications. Let us now consider the formulas that appear among the antecedents of the rule. Obviously, we have the local specifications, as the ultimate goal of the rule is to infer the truth of the global specification from the truth of all the local specifications. Therefore all the local specifications $E_i \succ M_i$ for all $1 \leq i \leq N$ appear as antecedents of a compositional inference rule.

In our example, we have both $\mathsf{eve}(a) \Rightarrow \mathsf{eve}(b)$ and $\mathsf{eve}(b) \Rightarrow \mathsf{eve}(a)$ among the antecedents.

Assumption discharging formulas. Since we are considering local specifications in rely/guarantee form, we cannot say anything about the truth

of the local guarantees unless we are able to prove the truth of the local assumptions. In other words, in order to use local guarantees as descriptions of the modules' behavior, we must first show that the modules' environments behave as required by the guarantees. But the system is made by the composition of the various modules; therefore, for each module i, the other modules constitute i's environment.

This suggests to introduce an hypothesis in the inference rule that allows one to show that the truth of the environment assumptions of the various modules follows logically from that of the modules' guarantees. Since the verb "discharge" is typically used to mean "prove an assumption", we call these hypotheses (one for each module) *assumption discharging formulas*. They are in the form $M_1 \wedge M_2 \wedge \cdots M_N \Rightarrow E_i$ for each module $1 \leq i \leq N$; notice that, in the most general case, we allow one to use any module j, including itself, to discharge the assumption. Later, we discuss how this gives rise to different kinds of rules.

The global environment E may take part in determining the properties of the environment of each module i. Therefore, in the most general case it may be required to use the global assumption E to discharge some local assumption E_i: the discharging formula becomes $E \wedge M_1 \wedge M_2 \wedge \cdots M_N \Rightarrow E_i$ in this case. In other words, the local environment of module i "inherits" some properties of the global environment.

In our running example, notice that the assumption of module 1 is the guarantee of module 2, and *vice versa*. Therefore, the assumption discharging formula for module 1 (resp. 2) coincides with the local specification of module 2 (resp. 1).

Circularity breaking and initialization. As it should be clear even from our trivial example, rely/guarantee compositional inference rules may involve a *circularity* between assumptions and guarantees of some modules. In our running example, this is apparent by the fact that each module relies on the other module's guarantee in order to behave correctly (i.e., according to its own guarantee). Circular reasoning is in general unsound, therefore we need to introduce an additional hypothesis in the inference rule to make the deduction sound. We call this additional hypothesis *initialization condition*.

Even if the rule does not involve a circularity, some form of initialization condition is required as "starting point" to apply the chain of inferences implied by the local specifications. In other words, if we have no circularities

between specifications and dischargings, we nonetheless have to start from the truth of some local assumption or guarantee to exploit the rely/guarantee specifications. Therefore, the existence of some initialization condition does not depend on the circularity of the rule.

The actual form of the initialization condition can vary a lot from inference rule to inference rule, and it might even be implicit (i.e., subsumed by some assumptions on the semantics of the language). In our simple example, it might simply consists of an assertion about a being even *a priori*: $\mathsf{eve}(a)$. This clearly breaks the circularity as it holds regardless of any assumption. In fact, in this case the application of the inference rule amounts to the application of the well-known *modus ponens* logic inference rule [Men97]: from the truth of $\mathsf{eve}(a)$ and $\mathsf{eve}(a) \Rightarrow \mathsf{eve}(b)$ we deduce $\mathsf{eve}(b)$, so overall we have $\mathsf{eve}(a) \wedge \mathsf{eve}(b)$. Although compositional inference rules will be much more complicated than this, being able to deal with temporal properties, our trivial example shows that the very basics of compositional reasoning are grounded in basic logic inference.

Global implementation. Combining the local specifications with the dischargings, and through the initialization condition, one should be able to infer the truth of some (or all) of the local guarantee formulas M_i's. One final ingredient is usually required: an hypothesis should link the truth of the local guarantees to that of the global guarantee; the latter is in fact the ultimate goal. Therefore, in general we have a formula $M_1 \wedge M_2 \wedge \cdots M_N \Rightarrow M$ among the antecedents of a compositional inference rule. We call this hypothesis *global implementation*.

In our simple running example, the global implementation is trivially true, as $M = M_1 \wedge M_2 = \mathsf{eve}(b) \wedge \mathsf{eve}(a)$.

As with the local dischargings, it may be necessary to exploit the assumptions about the global environment to infer the truth of the global guarantee. Therefore, a more general form of the global implementation formula is: $E \wedge M_1 \wedge M_2 \wedge \cdots M_N \Rightarrow M$

Time and initialization. Although the notions of compositionality and compositional inference rules is grounded in basic logic languages, our interest is about compositionality for temporal logic formalisms. Therefore, our compositional inference rules must deal with temporal descriptions of the behavior of systems.

In such cases, it happens often that the rely/guarantee behavior of

a module involves a temporal relationship between the assumption and the guarantee of the module. For instance, consider a functional module that takes some input and returns an output after some time T. Our running example may be refined to have such a temporal behavior: the local specification would now be expressible through the TRIO formula $\mathsf{eve}(a) \Rightarrow \mathrm{Futr}(\mathsf{eve}(b), T)$.

In accordance, we may need to restrict the "temporal scope" of the antecedents, or of the consequent, or both. Formally, this can be achieved by making each formula of the inference rule the consequent of an implication, whose antecedent is a (temporal) formula that holds if and only if we are within the desired "temporal scope". We denote such antecedents with the letter I. In general, we may have a different I for each hypothesis or conclusion of the inference rule. We denote them by adding subscripts and superscripts according to the formula to which that I is to be applied.

Returning one more time to our simple example, let us assume that both modules have a predicate s which is true exactly when the system is started. Then, for instance, the local rely/guarantee specification for module 1 would now be in the form $\mathsf{s} \Rightarrow (\mathsf{eve}(a) \Rightarrow \mathrm{Futr}(\mathsf{eve}(b), T))$, and similarly for the other formulas. The initialization condition would be naturally expressed by specifying that at when the system is started, a is even: $\mathsf{s} \Rightarrow \mathsf{eve}(a)$.

More generally, we observe that the need for a notion of "initialization" (or "system start") arises especially when one uses bi-infinite time domains (such as $\mathbb{R}$ and $\mathbb{Z}$, which are the default choices for TRIO); otherwise, with mono-infinite domains, the infimum element of the time domain conventionally denotes an absolute time instant at which the system starts. Indeed, we will see in Chapter 4 that most compositional inference rules in the literature adopt a mono-infinite time domain, and therefore do not use initialization predicates (at least in the form we have just presented).

General form of a rely/guarantee compositional inference rule. According to our explanations, the form of a rely/guarantee compositional inference rule is summarized in Table 3.1.

The observant reader may argue that the general form is not really a specialization of a generic inference rule, as the initialization condition allows any formula among the antecedents, and the global specification allows any formula as consequent through an adequate choice of compositional operator and of E, M. This is technically true, but the goal of the above schema

$\bigwedge_{1\leq i\leq N}\ (I_i^{\mathrm{sp}} \Rightarrow (E_i \succ M_i))$	local specifications
$\bigwedge_{1\leq i\leq N}\ \Big(I_i^{\mathrm{dsc}} \Rightarrow \Big(E \wedge \bigwedge_j M_j \Rightarrow E_i\Big)\Big)$	(assumption) discharging formulas
$I^{\mathrm{imp}} \Rightarrow \Big(E \wedge \bigwedge_j M_j \Rightarrow M\Big)$	global implementation
$I^{\mathrm{init}} \Rightarrow \mathbf{init}$	initialization (condition)
$I^{\mathrm{glb}} \Rightarrow E \succ M$	global specification

Table 3.1.: The components of a generic compositional inference rule.

is not to strictly characterize all compositional rules, but simply to give an (partly intuitive) idea of what a form a compositional rule should have, and to serve as a yardstick for the following developments, taxonomy, and explanations.

Syntactically-restricted completeness. For compositional inference rules one is usually interested in a notion of completeness that is slightly narrower than the one introduced previously for general inference rules. Namely, it is common to consider only formulas expressible as a global specification, that is through a compositional operator. Then, a compositional inference rule is complete — according to this syntactically-restricted notion of completeness — if: for any conclusion $\widetilde{\xi}$ that is expressible as $\widetilde{\xi} \equiv I^{\mathrm{glb}} \Rightarrow E \succ M$ and is true, there is some set of premises $\widetilde{\Pi}$ such that the pair $\widetilde{\Pi}, \widetilde{\xi}$ belongs to the relation defined by the rule, and $\widetilde{\Pi}$ is true. The difference between this definition and the more general one given above is in the fact that we now consider only formulas expressible according to the syntactic restrictions given by the statement of the compositional inference rule.

Notice that the syntactic restrictions may not impact semantic expressiveness at all: in such cases, when any formula can be written as a global specification, the two definitions coincide. This happens often in practice, as we will see with the compositional operators that we will introduce in the following sections. In the most general case, however, such a narrower notion of completeness is adopted since it is totally irrelevant to discuss the completeness of a compositional inference rule for formulas that are not expressible as specifications of a modular systems: in such cases com-

positionality is useless, and one should resort to traditional "flat" inference rules and standard deduction.

3.3.2. Dimensions for a Compositional Taxonomy

This section considers some common choices according to which the general compositional inference rule of Table 3.1 is instantiated into actual rules, and classifies the dimensions along which these instantiations are made. We stress again the fact that we do not aim at exhaustiveness — which would probably be of little practical interest anyway — but simply at giving a useful taxonomy which may guide the development of some actual rules.

Let us first list the dimensions we consider; afterward, we comment them.

- First, we distinguish between *circular* and *non-circular* (or *circle-free*) rules. A rule is circular if there are circularities between the discharging formulas and the local specifications: in other words, when the assumption of some module is discharged (directly or indirectly) through the guarantee of the same module.

- A *self-discharging* rule is one where the guarantee M_i of some module i appears on the left-hand side of the implication that discharges the assumption E_i of the same module. Notice that this is a different notion than circularity, as we discuss below.

- A rule is *fully compositional* when the global assumption E does not appear in any of the discharging formulas.

- A rule is *globally initialized* iff all of the formulas I_i^{sp}, I_i^{dsc}, I^{imp} are identically true; in other words, the truth of the corresponding formulas is not dependent of the occurrence of the start predicate. Otherwise, we call the rule *locally initialized*.

- Finally, another dimension along which to classify a compositional rule is the choice of the *compositional operator*: every choice of the compositional operator gives rise to a different "flavor" of inference rule.

Let us now give an idea of how the various features of inference rules may be related those of the systems whose correctness they are used to prove.

Circular vs. non-circular rules. This is a major feature according to which compositional inference rules can be classified. Circular rules are needed in practice when modeling systems whose components circularly rely on one another to function correctly when put together. For instance, the simple example of the two "adding 2" modules (shown in the previous Section 3.3.1) exhibits an obvious circularity, manifest by the fact that the guarantee $\mathsf{eve}(b)$ of module 1 coincides exactly with the assumption of module 2, and *vice versa*.

Indeed, our experience has shown that this is often *not* the case: designing a system whose components circularly rely on one another is most of the times an overly complicated choice, so it is not likely to be taken often. Of course, exceptions exist: for instance the dining philosophers example of Section 6.2, and the BitTorrent example of Section 6.3, are examples of systems that exhibit circularities. However, they have been chosen especially with this criterion in mind: to show circular inference rule in action. Therefore, in the remainder we also discuss non-circular inference rules, and argue that these are very commonly used in practice.

In general, the soundness of non-circular rules relies on very minimal (and simple) assumptions about the various elements of the rule. Indeed, their correctness is often a simple consequence of the elementary properties of the implication and of the conjunction, as we will show in Section 5.1.

On the other hand, showing the soundness of circular rules is in general more involved, in particular when dealing with the specification of temporal properties. In fact, circular reasoning is in general not sound, so that one has to show that there is some circularity breaking mechanism that resolves the issue (this is usually a result of the interplay between the initialization conditions and the properties of the compositional operator). This probably explains why most, if not all, the compositional rules that have been studied in the literature are circular (see Chapter 4): demonstrating their soundness has been deemed a sufficiently challenging problem to be of interest.[2]

Self-discharging rules. Whereas circularity is mostly a semantic property, self-discharging rules have a clear *syntactic* characterization.

In order to motivate the use of self-discharging rules, let us sketch a very simple example of a timed system where self-discharging is the natural way to go. For the sake of simplicity let us assume a discrete time domain, and

[2] Indeed, Abadi and Lamport [AL95] call "strong" inference rules which are circular.

let us just consider one single thermostat module.[3] Now, let us assume that the thermostat is capable of influencing the temperature of a room. In particular, whenever the temperature is within a given "safe" range, the device is capable of guaranteeing that the temperature is also within the range in the next time instant. Thus, if we introduce a time-dependent predicate ok to indicate that the temperature is within the desired range, the rely/guarantee behavior of the module may be described with the formula $\mathsf{ok} \Rightarrow \mathrm{NowOn}(\mathsf{ok})$. Notice that in this case $E = M = \mathsf{ok}$ for the module. Therefore, the tautology $\mathsf{ok} \Rightarrow \mathsf{ok}$ would naturally be a self-discharging formula for the assumptions in the system.

As this little example suggests, self-discharging rules may be needed in describing the behavior of modules with some form of feedback. In such modules, the functionality itself of the module (represented by the guarantee) may contribute directly, together with the other modules in the system, to make the environment of the module behave as in the assumption. On the other hand, when there is no feedback, then the module relies directly solely on the other modules to have a "correct" environment, while its actions contribute to guaranteeing that the other modules' environments are "correct". Another, more realistic, example of a system where self-discharging is useful is the controlled reservoir example presented in [Fur03b, Chap. 7].

Let us remark that self-discharging and circular are two distinct properties of a rule. More precisely, a self-discharging rule is always also a circular rule: this is apparent from our definition of circular rules. On the other hand, a rule may be circular without being self-discharging: in this case the circularity between assumption and guarantee of some module is not direct (i.e., within the same implication), but it requires some levels of indirection. The canonical example is that of two modules $1, 2$ whose local specifications are $E_1 \Rightarrow M_1$ and $E_2 \Rightarrow M_2$, and whose discharging formulas are $M_2 \Rightarrow E_1$ and $M_1 \Rightarrow E_2$. Then, the discharging formulas show that the rule is not self-discharging. However, the discharging of assumption E_1 (resp. E_2) relies on the guarantee M_2 (resp. M_1), which in turn depends on the assumption E_2 (resp. E_1) to hold, which relies on the guarantee M_1 (resp. M_2); thus all in all E_1 (resp. E_2) circularly depends on M_1 (resp. M_2) to be discharged.

[3] Clearly, this would make us lose the point of compositional reasoning, but we do this for illustration purposes only.

Fully compositional rules. According to the stricter definition of compositionality, which we presented in Section 3.1, in a compositional rule the truth of some global fact should be inferred "on the basis of its constituent components only". This suggests that the discharging of the assumptions should not depend on anything which is "outside" the system, such as the global assumption E. This is why we call a rule "fully compositional" if this is the case.

Closed systems — such as our "adding 2" example of the previous section — are a particularly relevant class of systems for which fully compositional rules clearly suffice. In fact, closed systems do not communicate with any external environment, so that the global assumption E is identically equal to true. In general, controlled systems are usually modeled as closed systems, as the physical system under control and the controller constitute each other's environment. From this viewpoint, modeling a complete system in a rely/guarantee style often results in a closed system, as one endeavors to include a model of the environment itself, according to the control standard paradigm.

On the contrary, when the modules are not "autonomous" in determining one other's environment, but they rely on some additional property of the global environment, one needs non-fully compositional rules. Notice also that an open system does not necessarily require a non-fully compositional inference rule, as the global assumption may be required only in the global implementation or in the global specification.

In the literature, several compositional inference rules are non-fully compositional (see Chapter 4). This is often driven more by an effort toward generality (and completeness), rather than by specific requirements of the systems of interests. In our analysis, we will consider both fully and non-fully compositional rules.

Globally vs. locally initialized rules. As we discussed in the previous Section 3.1, initialization predicates I are needed only when dealing with specifications describing the *temporal behavior* of a system. This is the focus of the present work, thus the notion of globally and locally initialized rules is needed.

If a system has a notion of "start" or initialization, then it may be that it behaves differently before than after it has started. Let us illustrate this on the thermostat example: let us postulate that the module can be started, and let us model this fact through a predicate s becoming true. Then, it

may be that the rely/guarantee behavior $\mathsf{ok} \Rightarrow \mathrm{NowOn}(\mathsf{ok})$ holds only after the system has started, that is after s has been true at least once (for simplicity, let us assume that further starts are ignored). Then, the initialization predicate I^{sp} for the local specification should be $I^{\mathrm{sp}} = \mathrm{SomP}(\mathsf{s})$, which is true precisely whenever s has been true (at least once) in the past. Thus, the full local specification would become: $\mathrm{SomP}(\mathsf{s}) \Rightarrow (\mathsf{ok} \Rightarrow \mathrm{NowOn}(\mathsf{ok}))$. Notice that this would also probably require to introduce the same initialization predicate for the other hypotheses and for the conclusion of the inference rule we would like to use. This corresponds to completely disregarding the behavior of the system before it has started.

Locally initialized rules subsume globally initialized ones, as in globally initialized rules one does not have to deal with initialization conditions except that in the initialization formula. Nonetheless, globally initialized rules usually present the local specifications in a simpler form, for modules whose rely/guarantee behavior does not depend on any initialization. Finally, notice that, in globally initialized rules, the initialization formula is only required for circularity breaking, in order to guarantee the soundness of the rule, but it does not influence the behavior of the single modules or the way they interact.

4. Related Work on Compositionality

This chapter presents a comparative review of several works about compositionality.

Besides the dimensions we have outlined in Chapter 3, there are a number of technical aspects by which compositional frameworks differ. Mostly, they derive from the various choices of reference formal language to be used, and on the semantics assumptions that are introduced in the abstract model. Let us recall that, in this sense, the framework we present in this thesis aims at generality, and thus it introduces as few semantic assumptions as possible.

Let us list these technical choices and briefly describe them.

Time Model This aspect considers continuous or discrete time, real-time (i.e. metric) or untimed (i.e. without metric) time models, the linear or branching nature of time, etc.

Computational Model This aspect considers issues such as whether we adopt a semantics based on states, on transition systems, an agent-based view, etc.

Model of Concurrency Choosing the model of concurrency involves considering issues such as synchronous vs. asynchronous, interleaving semantics, lower-level models of concurrency such as shared-variables concurrency and message passing systems.

Specification Model This aspect refers to the abstraction mechanisms introduced in the model that constitutes the specification. For example we may use an assertional approach, or an operational one, or a dual-language one (which combines an operational formalism with an assertional one), etc.

Implementation Language This aspect is only relevant if we are verifying implemented programs. In such case, the formal framework should give a formal semantics to the programming language.

Syntax vs. Semantics This refers to the general approach of the compositional method we are considering. We say it is *syntactic* if it is defined in terms of the language used in the formal description of the model, using rules that refer to the syntactic structure of the formulas, without reference to semantic concepts such as safety/liveness characterization, closures, etc. On the contrary, we label a method *semantic* if its applicability and justification are based on these latter concepts, usually expressed in terms of set-theoretic concepts.

How to Make a Module Compositional This aspect chooses among the rely/guarantee paradigm (dating back to Jones [Jon83] and Misra and Chandi [MC81]), or the lazy methodology proposed by Shankar [Sha98], or other original approaches such as the ones by Hooman [Hoo98], Manna et al. [BMSU01], etc.

Semantics of Composition The choice made by most authors is to take composition as conjunction of specifications for denotational formalism, and parallel composition for operational ones. Other, less common, variants are possible.

Table 4.1 schematically summarizes the above aspects. In italic are shown the choices we focused on in this thesis, that is a framework which works equally well for continuous and discrete time, relies on no assumptions about the computational and concurrency model, it is primarily syntactic, considers linear time and abstract specifications.

In the remainder of this chapter, we survey the most relevant works about compositional methods, with particular interest for rely/guarantee abstract inference rules. We refer the reader to [Fur05] for additional references and details about compositional framework in the literature.

4.1. Early Works on Compositional Methods

The first attempts at the formal verification of programs date back to the origins of computer science itself, with the pioneering works of Goldstine and von Neumann [GvN63], and Turing [Tur49], in the second half of the '40s. According to de Roever et al. [dRdBH$^+$01], a constant trend in developing methods for proving program correctness has been from *a posteriori* nonstructured methods to structured compositional techniques. For a detailed historical account of compositional and noncompositional methods

Issue	Possible choices
Time Model	*continuous*, *discrete*, *real-time*, untimed, *linear*, branching, etc.
Computational Model	state-based, agent-based, transition system, etc.
Concurrency Model	synchronous, asynchronous, interleaving, message passing, shared variables, etc.
Specification	*denotational*, operational, dual-language, etc.
Implementation	language used for implementation
Syntax vs. Semantics	*language* vs. (semantic) model
Compositionality Model	*rely/guarantee*, lazy, hybrid, etc.
Composition Semantics	*conjunction*, parallel, etc.

Table 4.1.: Dimensions of the Compositional World

and techniques for program verification, we refer the reader to the aforementioned book by de Roever et al. [dRdBH+01], and to the surveys by de Roever and Hooman [dR85, HdR86, dR98].

This development is rather easy to sketch for *sequential* program verification, where the first method is due to Floyd in 1967 [Flo67] and is non-compositional, whereas Hoare reformulated the method in an axiomatic compositional style in 1969 [Hoa69].

On the other hand, the first practical noncompositional methods for the verification of *concurrent* programs are due to Ashcroft [Ash75], Owicki and Gries [OG76], and Lamport [Lam77], during the mid '70s. Concurrency raises several technical difficulties which render the development of methods, as well as their compositional extensions, harder than in the sequential case. Indeed, the first compositional methods for concurrent program verification appeared at the end of the '70s, in the contributions by Francez and Pnueli [FP78], Jones [Jon83] and Misra and Chandy [MC81].

In particular, the works by Jones [Jon83] and by Misra and Chandy [MC81] introduced the rely/guarantee paradigm to specify open reactive systems. It was Jones who introduced the terminology *rely/guarantee* in [Jon81, Jon83], where he proposed a method for the shared-variable model of concurrency. His work was based on the previous work by Francez and Pnueli [FP78].

On the other hand, Misra and Chandy proposed in [MC81] a similar paradigm for concurrency models based on synchronous communication, and called it *assumption/commitment*.

Later, Abadi and Lamport proposed to call both compositional paradigms *assume/guarantee* [AL93], since they embody the same idea. Several of the works reviewed in Section 4.2 show in detail how the two paradigms of rely/guarantee and assumption/commitment can be unified. Because of this reason, in all of this paper we will not distinguish in the terminology between rely/guarantee, assumption/commitment and assume/guarantee: we choose to use only the term "rely/guarantee", in accordance with the name given to the first formulation of the principle [Jon81].

Finally, in this short recount of first works on compositionality for concurrent systems, it is worth mentioning the work by Stark, who formulated the first theory of *refinement* for concurrent processes in 1988 [Sta88].

4.2. Abstract Frameworks for Rely/Guarantee Compositionality

This section surveys several works about abstract compositional frameworks based on rely/guarantee inference rules that handle circularity. We try to show how all of these frameworks are fundamentally based on an inference rule which is a particular instance of the general rule of Table 3.1, to which each framework adds specific additional hypotheses for soundness.

The works are grouped into three sets, according to the basic formalisms used to express the compositional framework. Within each group, the works are presented roughly in chronological order, but listing sequentially works by the same authors which can be considered successive developments over the same ideas.

4.2.1. Purely Semantic Frameworks

Let us start by considering purely semantic works. By "semantic frameworks" we mean compositional frameworks based on inference rules expressed without reference to a particular language, but rather in terms of sets of behaviors and interpretation structures. This deviates from the presentation of inference rules that we did in Section 3.3.1, in that semantic rules have hypotheses and conclusions expressed as set-theoretic properties of behaviors, using directly set-theoretic notions to describe behavior sets.

Then, the concrete application of a semantic rule to a specific language requires one to check that the semantics of the language satisfies the semantic hypotheses made by the inference rule. This can be more or less simple, according to the peculiarities of the inference rule and of the language, but in any case it is left to the user of the rule.

Semantic frameworks often distinguish between system specifications that are, or are not, safety properties. In short, a safety property is one which is finitely refutable, that is every behavior failing to satisfy it does so at a finite point in time. For a more precise definition see [AS85,MP90]. Compositional reasoning with semantic frameworks is usually simpler for safety properties; this is the main reason for the widespread use of the notion of safety.

Abadi and Lamport [AL93] are the first to present a purely semantic analysis of compositionality, independent of any particular specification language or logic. This permits a general analysis of compositionality. However, the approach also has some drawbacks, which arise mainly from two aspects. On the one hand, the analysis is probably too abstract, that is it is far from immediate applicability with a given formalism. This is also briefly acknowledged by the authors themselves in [AL95]. By "too abstract" we also imply that its theoretical results have mostly intellectual (rather than practical) interest. On the other hand, the analysis done by Abadi and Lamport still requires a number of focused semantic assumptions, which mimic the features of the semantics of *some* specification formal language. As a result, mapping this general framework onto a given specification language may introduce additional complications and intricacies, due to the fact that a given language usually comes with its own semantics, which may be difficult to adapt or circumvent to match the general hypotheses of the paper.

Let us briefly outline the main semantic hypotheses of the model.

- Behaviors of described systems are infinite discrete sequences of states from a certain (unspecified) finite set.
- Any state change is triggered by agents. Agents can be divided into sets, which reflect their "nature". Notice that agents are an abstraction which typically translates to separated input and output variables, in a lower level formalism.
- An interleaving semantics for composition of behaviors is assumed.

- Stuttering-equivalence is also required, that is two behaviors are considered equivalent if they are equal after replacing every maximal sequence composed entirely of the same state with one single instance of that state.
- Fairness conditions (also known as progress properties) are usually introduced in a specification.
- Initialization predicates are also considered, so that the system has a notion of "start".

Let us now see how the general inference rule of Table 3.1 is modified by introduction of additional hypotheses.

- Each module is characterized by a set of agents that perform the state transition the module is responsible of. The agent sets for the various modules are assumed to be disjoint.
- Each environment assumption must be a safety property, constraining only agents which do not belong to any module (i.e. they belong to the environment). The authors note that, since it is well known that any property in temporal logic can be expressed as the intersection of a safety and a liveness property, it is possible to move the liveness part of the environment assumption to the guarantee part of the rely/guarantee specification. However, this consideration is of intellectual interest only, as the authors themselves admit in [AL95]. In fact, again, we must bear in mind that compositionality only has its strengths in being a *practically* usable technique, which is not the case if it forces one to write specifications in an unnatural manner.
- Implication and conjunction in the general rule of Table 3.1 are mapped to the set-theoretic operators $\subseteq$ and $\cap$. This is the most natural choice, since we are dealing with a semantic model where properties are sets.
- A rely/guarantee specification is written using the operator $\Rightarrow$, where $P \Rightarrow Q$ is a shorthand for the set-theoretic expression $(\mathcal{B} - P) \cup Q$, where $\mathcal{B}$ denotes the set of all behaviors.
- Local specifications are expressed in terms of the *realizable part* of the rely/guarantee expression $E_i \Rightarrow M_i$, denoted as $\mathcal{R}(E_i \Rightarrow M_i)$. Without introducing too many technical details, let us just say that

this requires the behavior defined by $E_i \Rightarrow M_i$ to be implementable, that is to be the outcome of a deterministic strategy.

- Assumption dischargings consider the safety closure $\mathcal{C}(M_i)$[1] of the M_i. Moreover, it is assumed that the safety closure of each M_i does not constrain any agent outside the agent set for the module i.

- A global implementation hypothesis is absent, since the rule actually proves $E \sqsubseteq \bigsqcap_{i=1}^{n} M_i$ instead of $E \sqsubseteq M$.

To sum up, the soundness is based on a discrete state structure, on which to perform induction, on the safety of the environment assumptions and on the distinction between input and output variables. Here it is the inference rule given in [AL93], after removing explicit mentions to the agents.

Proposition 4.2.1 (Abadi and Lamport, 1993 [AL93]). *If:*

1. *for all* $i = 1, \ldots, n$: $\mathcal{R}(E_i \Rightarrow M_i)$
2. $E \cap \bigcap_{i=1}^{n} \mathcal{C}(M_i) \subseteq \bigcap_{i=1}^{n} E_i$
3. E *and* E_i, *for all* $i = 1, \ldots, n$, *are safety properties.*

then: $\mathcal{R}(E \Rightarrow \bigcap_{i=1}^{n} M_i)$.

The rule was given some (partial) tool support in [HZB+96], using the theorem prover HOL.

Xu, Cau, and Collette [XCC94] introduce a compositional rule that subsumes two families of compositional rules, that is those for models of concurrency based on shared variables, and those for message passing models. The earliest works on composition for these models are the well-known contributions by Misra and Chandy [MC81] for message passing concurrency and by Jones [Jon83, Jon81] for shared variables. For our purposes, we will not account for the two specific models of concurrency, but we will just discuss and present the unifying rule of [XCC94].

The general framework is in the same spirit as Abadi and Lamport's [AL93]. More precisely, the following choices are made.

[1] The *safety closure* of a property P is the strongest (i.e., smallest) safety property P' such that P implies P'.

- Rely/guarantee specifications are written using the *spiral* operator $\hookrightarrow$, which has basically the same semantics as the $\stackrel{+}{\triangleright}$ of Abadi and Lamport [AL95], which we are going to present in Section 4.2.2.
- The assumptions are taken to be safety properties.
- The guarantees are expressed by the conjunction of a safety part M^S and a non-safety part M^R.
- We do not detail the treatment of agents, which is however very similar to that of [AL93].

Therefore, the inference rule goes as follows.[2]

Proposition 4.2.2 (Xu, Cau, and Collette, 1994 [XCC94]). *If:*

1. *for all* $i = 1, \ldots, n$: $(E_i \hookrightarrow M_i^S) \wedge (E_i \Rightarrow M_i^R)$
2. $E \wedge \bigwedge_{i=1}^n M_i^S \Rightarrow \bigwedge_{i=1}^n E_i$
3. a) $\bigwedge_{i=1}^n M_i^S \Rightarrow M^S$
   b) $E \wedge \bigwedge_{i=1}^n M_i^R \Rightarrow M^R$
4. $E \Rightarrow \bigwedge_{i=1}^n E_i$ *at the initial time (i.e. 0)*

then: $(E \hookrightarrow M^S) \wedge (E \Rightarrow M^R)$.

The main contribution of this work is the unification of the two different models of concurrency. In particular, it is shown that the general semantic framework, and the usual requirements on safety and "progression" operators can account for independently developed models of parallelism. Finally, it is speculated that a unifying rule may be useful in integrating the two models of concurrency in a single specification.

Cau and Collette extended this same semantic framework along the same lines in [CC96]. They choose to represent composition with the abstract semantic $\otimes$ operator, that merges two (discrete-time) behaviors into one (representing the composition of the behaviors). The $\otimes$ operator is assumed to respect simple properties, namely that if $\sigma = \sigma_1 \otimes \sigma_2$:

- $|\sigma_1| = |\sigma_2|$, that is only behaviors of equal length can be composed

[2] Actually, [XCC94] presents the rule for the case of two modules only (i.e. $n = 2$). Instead, we present here its verbatim generalization to arbitrary n.

- the result of the composition is a behavior with the same length of those composed, that is $|\sigma| = |\sigma_1| = |\sigma_2|$;
- any prefix of length k of σ is the composition of the prefixes of length k of σ_1 and σ_2, for all k.

The operator $\otimes$ is generalized to sets of behaviors (i.e. properties) as obvious: given properties P_1, P_2, we define their composition $P_1 \otimes P_2$ as the set of behaviors $\{\sigma | \sigma = \sigma_1 \otimes \sigma_2 \wedge \sigma_1 \in P_1 \wedge \sigma_2 \in P_2\}$. This abstract merge operator encompasses various models of composition, and in particular conjunction as well as disjunction.

Rely/guarantee specifications are assumed to be characterized by a triple of properties, which we indicate as E, M^S and M^R. As in the other work [XCC94], E is assumed to be a safety property, M^S is the safety part of the guarantee, and M^R is its non-safety part. We denote by M the set of behaviors which are both in M^S and in M^R, that is $M = M^S \cap M^R$. The link between the assumption and guarantee of a given module is formalized using an *ad hoc* operator, denoted as $@\!\!\rightarrow$. Its semantics is again basically the same as that of Abadi and Lamport's $\stackrel{+}{\triangleright}$ [AL95], which we will discuss in Section 4.2.2.

In presenting this inference rule, we do not represent, as usual, some details in order to fit the rule to the setting of our general rule of Table 3.1, and to convey only the very general intuition behind the rule.

Proposition 4.2.3 (Cau, and Collette, 1996 [CC96])**.** *If:*

1. *a)* $E_1 @\!\!\rightarrow M_1$
   b) $E_2 @\!\!\rightarrow M_2$
2. $E \otimes M_1^S \otimes M_2^S \subseteq E_1 \otimes E_2$
3. *a)* $M_1^S \otimes M_2^S \subseteq M^S$
   b) $E \otimes M_1^R \otimes M_2^R \subseteq M^R$

then: $E @\!\!\rightarrow M$.

Maier [Mai01] introduces another simple semantic compositional framework, and shows how it can be instantiated to some concrete computational models. The framework is based on set theory, and in this sense it is similar to [AL93]. However, with respect to [AL93] it is simpler and based on fewer semantic assumptions, and therefore more abstract.

In a nutshell, the framework is based on downward-closed sets. More precisely, a system is defined by a set of behaviors from a *universe set* $\mathcal{B}$, which is partially ordered (with order relation denoted as $\preceq$), well-founded (i.e. it has a *bottom element* ϵ), and *downward-closed*, that is for all b, b' if $b \in \mathcal{B}$ and $b' \preceq b$ then $b' \in \mathcal{B}$. Every property (or module) is identified with some downward-closed subset $B \subseteq \mathcal{B}$, with certain properties.

Structures called *chains* are introduced to describe the evolution of behaviors. A sequence of behaviors $b_i \in \mathcal{B}$ for $i \in \mathbb{N}$ is called a *chain* whenever:

- it ascends from the bottom, i.e. $b_0 = \epsilon$ and for all $i \in \mathcal{B}$: $b_i \preceq b_{i+1}$;
- it converges to some limit in $\mathcal{B}$, that is the upper bound b of the set $\{b_i | i \in \mathbb{N}\}$ exists and is in $\mathcal{B}$.

With a little abuse of notation, we will denote a chain $(b_i)_{i \in \mathbb{N}}$ simply as b_i.

We need two more notions to introduce the compositional inference rule of the framework in detail. One is the idea of *extension*: given two sets $B_1, B_2 \subseteq \mathcal{B}$ and a chain b_i, we say that B_1, B_2 extend along b_i iff, for all $i \in \mathbb{N}$, if $b_i \in B_1 \cap B_2$ then $b_{i+1} \in B_1 \cup B_2$. The other notion, more familiar, is that of closure: a set $B \subseteq \mathcal{B}$ is closed with respect to a chain b_i if, whenever $b_i \in B$ for all $i \in \mathbb{N}$, the limit of b_i is also in B.

Finally, as it is natural in a set-theoretic framework, composition is represented by intersection $\cap$ and implication by the subset relation $\subseteq$.

We are now ready to introduce the abstract compositional rule of the framework. We note that it is proposed in a narrower form than that of the general rule of Table 3.1: it considers the simple case of two modules only, and it considers composition into a closed, rather than open, overall system. Here it is the inference rule.

Proposition 4.2.4 (Maier, 2001 [Mai01]). *If:*

1. *a)* $E_1 \cap M_2 \subseteq M_1$
   b) $E_2 \cap M_1 \subseteq M_2$
2. *for every* $b \in \mathcal{B}$*, there exists a chain* b_i *(for* $i \in \mathbb{N}$*) such that:*
   a) b_i *converges to* b
   b) M_1, M_2 *extend along* b_i
   c) M_1, M_2 *are closed with respect to* b_i

then: $E_1 \cap E_2 \subseteq M_1 \cap M_2$.

Notice that the requirement 2c on the closure of M_1, M_2, together with downward-closure, is a strict analogue of requiring the guarantees to be safety properties, as done in other compositional frameworks. Moreover, the extension condition 2b mirrors the idea of a compositional operator such as Abadi and Lamport's *while-plus* [AL95], which we present in the next section. This, combined with the well-foundedness of the set of behaviors $\mathcal{B}$, permits the application of the usual induction in proving the soundness of the rule.

The proposed framework is simple enough to be interesting as a general framework. Its major contribution is, in our opinion, to explicitly pin down some basic facts about compositional reasoning at the semantic level. These facts constitute the basis of most works on compositionality and therefore they can be understood better. In this sense, we believe that this work is similar in spirit to [AENT03].

4.2.2. Compositionality for TLA

With reference to the specification language TLA [Lam94], Abadi and Lamport propose in [AL95] a sound composition theorem. The following assumptions are introduced to render the deduction rule sound.

First, general restrictions are implicit in the use of TLA as a specification language, and namely:

- Although time variables can take values over a continuous domain (i.e. the reals $\mathbb{R}$), the semantic of the language is defined in terms of discrete sequences of states, where a state is an assignment of values to the set of variables. Therefore, there is a considerable, and intrinsic, "amount of discreteness" in the description of a system with TLA. This is also shown by the important role given to invariance under stuttering.

- In TLA, there is the notion of initialization of variables, and therefore of "beginning" of the life of a system.

- Usually, it is assumed that the set of all variables can be partitioned into two subsets e and m of input and output variables. The specification is such that the assumption formulas only modify variables in e, while the guarantee formulas only modify variables in m. Actually, how to treat the more general case (i.e., where some variables

are modified both by assumptions and by guarantees) is discussed briefly in [AL95], although the solution still has some limitations.

- Although TLA can handle both interleaving and non-interleaving composition, the two cases must usually be treated separately in the correctness proofs.

Second, the following additional assumptions are introduced specifically in the rule.

- The local specifications are written using the $\stackrel{+}{\triangleright}$ "while-plus" compositional operator. Quoting [AL95], a specification

  > "is expressed by the formula $E \stackrel{+}{\triangleright} M$, which means that, for any n, if the environment satisfies E through "time" n, then the system must satisfy M through "time" $n+1$".

  Notice that such an operator requires the notion of discrete "time" and permits some form of induction on such a discrete structure with a beginning.

- Assumption dischargings are expressed through the *safety closures* of the involved formulas. The authors introduce some lemmas that permit to get rid of the closures in the formulas, under additional technical conditions.

- The global implementation formula is coupled with a similar one, where the safety closures of the formulas are considered. In particular, E is replaced by $\mathcal{C}(E)_{+v}$, a formula which asserts that, if $\mathcal{C}(E)$ ever becomes false, then the values of all the specification variables v stay unchanged always in the future. As for the previous condition, rules for handling the closures and the $+$ operator are proposed.

To sum up, the soundness in the compositional rule is gained by considering specifications written in a canonical form, exploiting an induction (using the $\stackrel{+}{\triangleright}$ operator) over a discrete structure of states with a beginning, and considering safety characterizations of some of the assumptions and guarantees formulas of the specification, handling these safety characterizations with *ad hoc* rules and techniques. Here it is the composition rule.

Proposition 4.2.5 (Abadi and Lamport, 1995 [AL95]). *If:*

1. *for all* $i = 1, \ldots, n$: $E_i \stackrel{+}{\triangleright} M_i$
2. $\mathcal{C}(E) \wedge \bigwedge_{i=1}^{n} \mathcal{C}(M_i) \Rightarrow \bigwedge_{i=1}^{n} E_i$
3. *a)* $E \wedge \bigwedge_{i=1}^{n} M_i \Rightarrow M$
   b) $\mathcal{C}(E)_{+v} \wedge \bigwedge_{i=1}^{n} \mathcal{C}(M_i) \Rightarrow \mathcal{C}(M)$

then: $E \stackrel{+}{\triangleright} M$.

We notice how the rule proposed by Abadi and Lamport introduces several assumptions on the computational model, and on how a specification, and the corresponding proof, "should be written". These restrictions may reduce generality and immediacy of use.

4.2.3. Compositionality for Intuitionistic Logic

Abadi and Plotkin [AP93] perform an abstract study of compositionality, in two frameworks based on intuitionistic and linear logics. The work treats compositionality at the semantic level, and considers safety properties only. As we briefly discussed at the beginning of Section 4.2.1, safety properties usually require simpler (or fewer) semantic assumptions for an inference rule to be sound, and are therefore the first and main focus in many compositional frameworks.

For brevity, let us just consider the simpler approach based on intuitionistic logic. The framework and the results closely match those of [AL93], with restriction to safety properties for both the assumptions and the guarantees. With respect to [AL93], an additional theoretical result is the reduction of the compositional inference rule to the the case for just one module. Let us state this result more explicitly. The semantic model is the same as that of [AL93]. It can be shown that such a model is an intuitionistic one. With these definitions and semantic assumptions, the following very simple circular inference rule is proved sound ($\rightarrow$ is a suitably defined implication).

Proposition 4.2.6 (Abadi and Plotkin, 1993 [AP93])**.** *If:*

1. $E \rightarrow M$
2. $M \rightarrow E$
3. E *and* M *are safety properties*

then M holds.

Although speaking of compositionality with just one module is pointless, one can derive the general compositional rule of [AL93], restricted to the intuitionistic framework, from the rule of Proposition 4.2.6, using only propositional reasoning and induction. This result is interesting as it basically shows that the only catch in this kind of compositional reasoning with safety properties lies in disentangling the circularity in the discharging of assumptions. Abadi and Plotkin demonstrate that the general case of this problem can be reduced to one module systems. However, it has no obvious practical impact, since it is a specialization of previously seen results by other means.

All in all, the paper by Abadi and Plotkin highlights quite clearly these basic facts about compositional reasoning:

- technical problems arise in proving the soundness because of the presence of circularity;
- these problems can be solved by restricting our attention to safety properties, since proving soundness reduces to proving a sort of "initial validity", which can be regarded as an implicit way of breaking the circularity;
- finally, induction is used to "propagate" the initial validity to any instant in the future.

Abadi and Merz [AM95] introduce an abstract syntactic framework in which to express compositional rules. Actually, even if it is based on an intuitionistic logic, it underlines some of the assumptions about the computational model which were also shared by other works such as [AL93,AL95], so it is not totally independent of particular computational models.

The framework consists of a propositional intuitionistic logic, with the usual logic connectives, plus a new connective $\stackrel{+}{\rightarrow}$ which clearly represents TLA's $\stackrel{+}{\triangleright}$ and similar ones. Even if the framework should be primarily syntactic, it actually introduces some assumptions on the structures on which the logic is to be interpreted. More precisely, these assumptions are:

- a pre-order relation $\subseteq$ is given on the sets of behaviors;
- the relation is such that atomic propositions are true on downward-closed sets of behaviors; that is, for any atomic proposition π and behaviors σ, σ' such that $\sigma \subseteq \sigma'$, if $\sigma' \models \pi$ then also $\sigma \models \pi$.

- the relation $\subset$ obtained from $\subseteq$ by removing reflexivity is well-founded on the set of all behaviors; notice that this permits induction along computations, similarly to the aforementioned models.

Moreover, notice that these assumptions are satisfied whenever the interpretation structures are Kripke models (which are standard models for intuitionistic logic).

Under these assumptions, Abadi and Merz prove a number of basic properties of the operators of the logic. However, a sound compositional rule on the model of Table 3.1 cannot be proved without further semantic assumptions and specializations. More specifically, the authors choose to instantiate the abstract framework with two temporal logics, namely TLA and a fragment of CTL*. Sound compositional rules can be proved only *after* such instantiations are done. Hence, the idea of an abstract framework is appealing in its syntactic approach, but is not detailed enough to implement a sound circular rule. However, one advantage of this approach is that it is not limited to temporal logics. The authors briefly discuss this aspect in the conclusion. Under this aspect, the approach is indeed more general than most others.

4.2.4. Compositionality for LTL

Jonsson and Tsay [JT96] analyze compositionality with linear temporal logic, from a syntactic perspective. The outcome is a compositional inference rule which, although similar in principle to that of Abadi and Lamport [AL95], is presented using a syntactic formulation, which has the advantage of being simpler to use, in practice, than purely semantic approaches.

The basic idea is still to rely on safety properties to render sound the compositional circular reasoning, but to enforce the safety characterization by requiring that the formulas of the specification are written according to some syntactic restrictions. In fact, in LTL it is possible to syntactically characterize safety [MP90]. In principle, the same syntactic approach could be pursued with TLA as well, but maybe with more practical complications. On the contrary, the syntactic characterization in LTL is often more viable.

The syntactic characterization requires the specifier to write the formulas for assumptions and guarantees in a canonical form, to enforce the underlying semantic hypotheses. The requirements for the canonical form are

the following ones.[3]

- Assumptions E are expressed with a LTL formula of the form $\Box(\exists x : \boxminus H_E)$, where:
  - H_E is a past formula, that is a formula without future operators;
  - x is a tuple of flexible variables, considered as internal variables (i.e. non-visible, existential quantification representing hiding);
  - $\Box H_E$ must be stuttering extensible, another technical requirement we do not discuss in detail.

  Under these conditions, the formula E is a safety formula. Actually, the requirement of H_E being a past formula could be relaxed to the requirement to be a *historical* formula, but with some additional complications.
- Guarantees M are expressed with a LTL formula of the form $\exists y : \Box H_M \wedge L_M$, where:
  - H_M is a past formula;
  - y is a tuple of flexible, internal variables;
  - $\Box H_M$ must be stuttering extensible;
  - L_M is a liveness property;
  - (H_M, L_M) is machine-closed, that is $\mathcal{C}(\Box H_M \wedge L_M) \Leftrightarrow \Box H_M$.
- Rely/guarantee specifications are expressed using the operator $\triangleright$, where $E \triangleright M$ is defined as $\Box(\widetilde{\ominus}(\exists x : \boxminus H_E) \Rightarrow (\exists y : \boxminus H_M)) \wedge (E \Rightarrow M)$.

Under these conditions, we can calculate syntactically the safety closures of the specification formulas. In fact it turns out that:

- $\mathcal{C}(\exists x : \boxminus H_E) = \Box(\exists x : \boxminus H_E)$
- $\mathcal{C}(\exists y : \Box H_M \wedge L_M) = \Box(\exists y : \boxminus H_M)$

Finally, the resulting compositional rule follows.

[3] The temporal operators are LTL's usual ones [MP90], and namely $\Box$, $\boxminus$, $\widetilde{\ominus}$ for, respectively, always in the future, always in the past, in the previous instant (if it exists). Moreover, $\mathcal{C}(F)$ represents the safety closure of a formula F.

Proposition 4.2.7 (Jonsson and Tsay, 1996 [JT96]). *If:*

1. *for all* $i = 1, \ldots, n$: $E_i \rhd M_i$
2. $\Box((\exists x : \boxminus H_E) \wedge (\exists y_1, \ldots, y_n : \boxminus \bigwedge_{i=1}^{n} H_{M_i}) \Rightarrow (\exists x_1, \ldots, x_n : \boxminus \bigwedge_{i=1}^{n} H_{E_i}))$
3. a) $\Box(\widetilde{\ominus}(\exists x : \boxminus H_E) \wedge (\exists y_1, \ldots, y_n : \boxminus \bigwedge_{i=1}^{n} H_{M_i}) \Rightarrow (\exists x_1, \ldots, x_n : \boxminus H_M))$
   b) $E \wedge \bigwedge_{i=1}^{n} M_i \Rightarrow M$

then: $E \rhd M$.

Discussions on how to actually discharge the hypotheses of the inference rule are introduced, with reference to the usual proof methodologies for temporal logics. Moreover, the same example of a queue of [AL95] is carried out, resulting in a simpler compositional proof.

To sum up, Jonsson and Tsay try to identify general conditions under which the semantic assumptions of the earlier works on compositionality can be expressed syntactically in LTL, resulting in inference rules which are simpler to apply and require less *ad hoc* methodology. Actually, not all semantic assumptions are rendered syntactically (for example machine closure and stuttering extensibility conditions are still required explicitly). Moreover, the syntactic restrictions may render the specifications harder to write. However, it is true that some aspects are indeed simplified and the application of the rule in some cases promises to be practically simpler.

Jonsson and Tsay's work presented in this paper was extended in a later work by Tsay [Tsa00]. The main additional contributions are in the use of a weaker compositional operator, and in some treatment of liveness formulas. In particular, an asymmetric circular rule is given. Asymmetric means that only one module is allowed to have a liveness property. For brevity, we refer to [Fur05] for additional details about Tsay's work.

4.2.5. Completeness of Compositional Rules

Namjoshi and Trefler [NT00] present an interesting analysis concerning the *completeness* of the compositional inference rules found in the literature, with specific reference to that of Abadi and Lamport [AL95], and McMillan [McM99]. Since those rules are the basis for most other existing rules in

the literature, the analysis draws rather general results. Furthermore, they tackle the problem of circularity from yet another perspective, showing a reason why circular rules are not needed, at least in principle.

McMillan's framework of [McM99] is the basis of the methodological work by the same author [McM00], where it is shown how to apply compositional techniques to the practical verification of system-level hardware, through model-checking techniques. We present McMillan's approach to compositional rules below, while the methodology introduced in [McM00] is not discussed here as it is beyond the scope of the present review section, being focused on model-checking techniques.

First of all, we have to state formally what are the compositional rules considered in Namjoshi and Trefler's paper. To render the comparison simpler, let us stick to the case of just two modules. The first rule is the circular one by Abadi and Lamport [AL95], described in Section 4.2.2. The second one is a very simple non-circular rule, which we can state as follows.[4]

Proposition 4.2.8 (Non-Circular Rule, [NT00])**.** *If:*

1. a) $E_1 \Rightarrow M_1$
   b) $E_2 \Rightarrow M_2$
2. a) $E \Rightarrow E_1$
   b) $M_1 \Rightarrow E_2$
3. $M_1 \wedge M_2 \Rightarrow M$

then: $E \Rightarrow M$.

The formulas M_1 and E_2 are called *auxiliary assertions* in this framework, since they serve to break the proof into two suitable parts, in a way that permits a sound deduction.

The third rule we consider is the one by McMillan [McM99]. Even if in [NT00] a generalization of the rule to handle n modules is presented, the generalization has a rather convoluted formulation. Therefore, for the sake of simplicity, we present here the original version of [McM99] for two modules only, in a minimal setting (that is, without reference to the computational model). Note that the rely/guarantee specifications are now

[4] Actually, the exact formulation of the rules of [NT00] would require a notion of computational model and several semantic notions. For simplicity, we present here only a simplified abstraction of a slight modification of that rule.

written using the $\triangleright$ operator, called *constrains* in that paper. As usual, $E \triangleright M$ is true iff, if E is true up to step i of a computation, then M is true at step $i+1$. It can be defined in linear temporal logic, using the *until* operator as $\neg(E\,\mathcal{U}\,\neg M)$, using the dual *release* operator as $\neg E\,\mathcal{R}\,M$, and can be thought as meaning "$\neg E$ precedes M", that is E is falsified before M is. The circular rule is then as follows.

Proposition 4.2.9 (McMillan, 1999 [McM99])**.** *If:*

1. a) $E \Rightarrow (M_2 \triangleright M_1)$
   b) $E \Rightarrow (M_1 \triangleright M_2)$
2. $M_1 \wedge M_2 \Rightarrow M$

then: $E \Rightarrow \Box M$.

Namjoshi and Trefler first show that, while the non-circular rule of Proposition 4.2.8 is complete, the other rules of Abadi and Lamport [AL95] and McMillan (Proposition 4.2.9 and [McM99]) are incomplete, even for simple properties. The incompleteness in McMillan's rule is due to the absence of auxiliary assertions, that is local assumptions E_i's, different than the other module's guarantee M_i. In fact, in a rather different setting, Maier [Mai03] has shown that the use auxiliary assertions is a necessary condition for completeness (see below). Still, it is not sufficient, since Abadi and Lamport's rule [AL95] does use auxiliary assertions but is nonetheless incomplete, as Namjoshi and Trefler also show in their paper.

Next, the authors show how to strengthen McMillan's circular rule in order to get a complete rule, obviously still keeping soundness and circularity. In order to do that, we have to write the assumptions and guarantees in a (simple) canonical form, where the parts representing the auxiliary assertions are conjoined to the remainder of the specification. So, each assumption E_i is written as $E_i = H_{\hat{i}} \wedge G_{\hat{i}}$ and each guarantee M_i as $M_i = (H_{\hat{i}} \Rightarrow G_i) \wedge H_i$, for some formulas H_i, G_i ($\hat{i}$ represent the index "other than" i, that is $\hat{i} \in \{1,2\} \setminus \{i\}$). The rule is finally as follows.

Proposition 4.2.10 (Namjoshi and Trefler, 2000 [NT00])**.** *If:*

1. a) $E \Rightarrow ((H_2 \wedge G_2) \triangleright ((H_2 \Rightarrow G_1) \wedge H_1))$
   b) $E \Rightarrow ((H_1 \wedge G_1) \triangleright ((H_1 \Rightarrow G_2) \wedge H_2))$
2. $G_1 \wedge G_2 \Rightarrow M$

then: $E \Rightarrow \Box M$.

Notice that Proposition 4.2.10 reduces to Proposition 4.2.9 if we take $H_1 = H_2 = true$.

The last aspect of compositionality considered by Namjoshi and Trefler is the translation between circular and non-circular proofs. Since both the non-circular rule of Proposition 4.2.8 and the circular rule of Proposition 4.2.10 are complete, every compositional proof can be carried out with any of the two, at least in principle. Hence, it is shown how one can translate the application of a rule into the application of the other rule; in other words, it is shown how one should pick the local assumptions and guarantees (possibly including the auxiliary assertions) so that the applicability of one rule follows from the application of the other one to the same system. It is also briefly discussed how this translation is efficient in both directions, that is the translation process builds formulas that are of the same size order as the formulas from which it translates (i.e., the ones of the original application of the rule).

This formalizes an important conclusion: circularity is not needed, at least in principle. Actually, this is is yet another evidence to the fact that compositional techniques must be useful *in practice*, but cannot reduce the worst-case complexity of verification. Therefore, a more complicated rule (and namely a circular one) can bring some benefits in some concrete cases, but is not more efficient than another equally complete rule in the general case.

Maier [Mai03] considers compositional rely/guarantee inference rules in a very abstract setting, in order to draw general conclusions about soundness and completeness of such rules. Contrarily to the other papers considered above, it does not introduce any new inference rule, but it considers the inherent limitations of circular compositional rely/guarantee reasoning in a given abstract framework. Let us first describe such framework and then show the results drawn by Maier.

In this framework, systems (or modules) and modules' properties are modeled uniformly as elements of a generic meet-semilattice with one. A meet-semilattice with one $\mathbf{S} = \langle S, \wedge, 1, \leq \rangle$ is a partial order $\langle S, \leq \rangle$ with greatest element $1 \in S$ and such that for any two elements $x, y \in S$ there exists their greatest lower bound, denoted as $x \wedge y \in S$. Intuitively $x \wedge y$ indicates the composition of two modules, or the conjunction of two properties; in other words, it corresponds to the compose operator $\sqcap$ of our

abstract setting. The entailment relation $\sqsubseteq$ is instead represented by the refinement relation $x \leq y$ (x refines y); equivalently it represents the fact that module x satisfies property y. In other words, the model assumes only a refinement order relation $\leq$, an associative commutative and idempotent operation $\wedge$ which respects the order, and a discrete structure S over which to interpret (in particular, no computational model is required).

In this setting, we define an *inference rule* as any rule R in the form:

$$R : \frac{\phi_1 \cdots \phi_n}{\psi} \text{ if } \Gamma$$

where $\Phi = \{\phi_1, \cdots, \phi_n\}$ and ψ are called *premises* and *conclusion*, respectively, as in our definitions of Section 3.3.1, while Γ is a relation called *side condition*. The meaning of R is as expected: if the premises and the side condition are true, then the truth of the conclusion follows logically. Notice that the language in which one can express the formulas is rather restricted in the given setting, so that the side condition is needed to express more powerful constraints (relations are strictly more expressive than formulas in the given setting).

Given an inference rule R, Maier defines R as:

sound iff for all evaluations α, $\alpha \models \Phi$ and $\alpha \models \Gamma$ implies $\alpha \models \psi$, i.e., the rule draws true facts from true premises;

complete iff for all evaluations α, $\alpha \models \psi$ and $\alpha \models \Phi$ implies $\alpha \models \Gamma$, i.e., the rule is applicable whenever premises and conclusions hold (in other words, the side condition is not "too restrictive");

assume-guarantee iff for all premises $\phi \in \Phi$, the left-hand side of ψ and the right-hand side of ϕ do not share any variables, i.e., we can identify "assumptions" and "guarantees" without ambiguities;

circular iff it is assume-guarantee and in general $\Phi \not\models \psi$, i.e., the side condition is strictly needed to guarantee soundness;

compositional iff it is assume-guarantee and (informally) it never considers the system composition (i.e., the conjunction of all modules) in the premises and in the side-condition (in other words, we can consider each module independently of the others).

Notice that some of Maier's definitions are non-standard; in particular, this is the case for his definition of completeness.

Assuming the aforementioned definitions and structures, the author shows that if the semilattice $\mathbf{S}$ contains forks (which are certain special structures) of sufficient width (and, in particular, of infinite width), then any compositional circular assume-guarantee rule is either unsound or incomplete. Since soundness is obviously indispensable, we must either give up completeness or compositionality. In the case of automatic verification, compositionality is probably more important, since it permits to really divide the burden of verification among modules, whenever possible, thus lowering the required computational effort. The author argues that in the case of manual verification, completeness may be more important, since the human user can always, at least in principle, invent a suitable system decomposition to correctly carry out the verification task.

Since the proposed framework is a very abstract one, it is important to understand if and how the results drawn for it can be extended to more concrete frameworks. The author claims that most rely/guarantee inference rule presented in the literature can be transformed into rules over meet-semilattices with forks of infinite width, while preserving properties such as soundness and compositionality. Hence, the limitation of completeness must hold also for the original rule.

Brief examples are shown of rules for Moore or Mealy machines, such as those in [HQRT02, Mai01], or for temporal logics, such as those in [AL95, AM95, JT96]. In particular, the presence of forks of infinite width follows quite naturally from the fact that the aforementioned frameworks are interpreted over sets of strings of infinite length: such structures provide such forks in all non-trivial cases. However, although the problem is not fully developed, it is likely that more expressive formalisms cannot be interpreted over meet-semilattices in all circumstances. For instance, it is not clear if metric temporal logics over a continuous time domain can be fully translated using the simple formalism of [Mai03], since such translation seems possible only for interpretations over algebraic (and discrete) structures. A more accurate investigations of these facts belongs to future work.

We mentioned that Maier's definition of completeness is a non-standard one. Indeed, Maier introduces the standard notion of completeness and calls it *backward* completeness, since — among other things — it is useful to characterize rules which enable backward reasoning. Backward completeness is used, among others, in [NT00]. In Maier's framework, a rule $R : \Phi/\psi$ if Γ is backward complete iff for all evaluations α, if $\alpha \models \psi$ then

$\alpha' \models \Phi$ and $\alpha' \models \Gamma$, for some evaluation α' which agrees with α on all the variables of ψ. In other words, backward complete rules admit the use of *auxiliary variables*, i.e., variables appearing in the premises but not in the conclusion, and of assertions on them; such auxiliary variables increase the expressive power of the inference rules. For rules without auxiliary variables, backward completeness implies completeness; therefore, every sound and backward complete compositional circular assume-guarantee rule necessarily employs some auxiliary variables. Notice that the presence of auxiliary variables also increases the complexity of the proofs, since one also needs to "guess" the value of auxiliary assertions about the system.

All in all, this paper brings forward an interesting analysis of some basic issues concerning rely/guarantee inference rules. In order to be able to apply the conclusions of the paper to concrete formalisms, it is important to always carefully consider, case by case, if and how the definitions characterizing the framework can be express suitably in less abstract settings. More precisely, things should probably be reconsidered with more expressive frameworks (in particular with continuous time formalisms) or with different definitions for inference rules and for completeness.

Amla et al. [AENT03] build a generic (i.e., sufficiently abstract) sound and complete compositional inference rule, following an approach which is different from those of most previous works. In fact, nearly all methods render the circular reasoning of Table 3.1 sound by introducing some sort of *progression* operator, which justifies an induction on the set of behaviors prefixes. On the other hand, this paper does not rely on any *ad hoc* operator, but rather it introduces some precise semantic assumptions among the conditions on the rules. These additional constraints also permit a well-founded induction on the set of behaviors prefixes, thus rendering the inference rule sound.

Actually, the focus of the paper is on refinement, so the setting is different than that of the other works we considered. However, since the approach is novel, we review it anyway, briefly detouring from our basic setting. Actually, refinement (and decomposition) can be regarded as a special case of composition, under reasonable conditions; for example, Abadi and Lamport in [AL95] derive their decomposition theorem as a corollary of the composition theorem. However, the differences in the two settings are such that the terminology and notational conventions of the composition setting, which we have considered thus far, sound strange and do not

provide intuition in the decomposition setting. Therefore, we are going to introduce some new notations, in order to provide the intuition needed to understand the core results of this paper.

The first difference of the decomposition setting is that we do not have rely/guarantee specifications of modules, but rely/guarantee only refers to the style of reasoning employed by the inference rule. For simplicity, let us consider the simple case of two modules, as it is done in [AENT03]. The goal of decomposition reasoning is to show that the composition of the two low-level specifications M_1^L and M_2^L of the modules (e.g. their implementations), implements a certain property M, possibly under a certain global environment specification E. In general, the composition of the two low-level specifications is very complicated, therefore the direct proof of the above is unfeasible or too costly (usually, we consider automated verification, but the same holds for manual verification). On the contrary, rely/guarantee reasoning gives conditions under which we can consider only the composition of the high-level specifications M_1 and M_2 of the modules, which is in general simpler than its low-level counterpart, together with one low-level specification *at a time*, that is avoiding to consider directly the composition of the two low-level specifications. The reasoning is rely/guarantee since we usually require that M_1^L refines M_1, *assuming* M_2 holds, and *vice versa* for the other module. Notice that it is not sufficient to just require that M_i^L refines M_i, since this is in general not the case: usually, the refinement relation holds only when the other module is a proper environment to the former one.

The reduction to the composition framework can be understood by taking the M_i's of the composition setting as the M_i^L's of the decomposition setting, and the E_i's as the M_i's of the decomposition setting. However, notice that the E_i's now represent the assumption of one module about the behavior of the other, which is needed for the refinement relation to hold, and not a condition under which the module behaves properly.

We are now ready to introduce the abstract framework of [AENT03]; as usual we introduce only the relevant aspects, while avoiding some details for simplicity. We consider the set of all behaviors (i.e. computations) $\mathcal{B}$. The following assumptions characterize the set of behaviors:

- $\mathcal{B}$ is partitioned into the two non-empty subsets of finite and infinite behaviors;
- $\mathcal{B}$ is partially ordered by an order "prefix" relation $\preceq$;

- the set of finite behaviors is downward closed under $\preceq$ (i.e. every prefix of a finite behavior is also a finite behavior);
- $\prec$ is well-founded on the set of finite behaviors; notice that this implies the existence of an initial condition, that serves as base for inductions.

As usual, a specification is a subset of $\mathcal{B}$, that is a set of behaviors, with the additional condition that every finite prefix of a behavior in the specification is also a (finite) behavior in the specification. This requirement basically extends downward closure to specifications. For the sake of simplicity, we overlook other technical requirements that are introduced in [AENT03]. Notice that all these conditions are semantic, and consider algebraic (and discrete) structures; in fact, the classical framework of state sequences is a straightforward realization of these assumptions.

Finally, notice that implication $\Rightarrow$ between formulas represents *refinement*; in other words $A \Rightarrow B$ denotes the fact that A is a lower-level specification that implements the higher-level specification B.

We are now ready to state the decomposition rule of [AENT03]. In a nutshell, the rule aims at showing that $E \wedge M_1^L \wedge M_2^L$ refines M, without considering explicitly the conjunction $M_1^L \wedge M_2^L$ in the hypotheses of the rule. In order to do that, we require:

1. that the low-level specification of each module, when composed with the high-level specification of the other, refines the high-level specification of the former module;
2. that the composition of the high-level specifications refines M, under the assumption E;
3. that the composition of one (anyone will do) of the low-level specifications with the (safety) closure of M refines one of the high-level specifications, or M itself. Notice that this last condition is the semantic requirement that allows to break the circularity, giving a base case for the induction, since it requires that one low-level specification refines a high-level one, without assuming anything about the other module.

The formal statement of the rule, using our notation, follows. The authors prove in [AENT03] that the rule is both sound and complete for the given setting.

Proposition 4.2.11 (Amla, et al., 2003 [AENT03]). *If:*

1. a) $M_1^L \wedge M_2 \Rightarrow M_1$
   b) $M_2^L \wedge M_1 \Rightarrow M_2$
2. $E \wedge M_1 \wedge M_2 \Rightarrow M$
3. *for some* $i \in \{1, 2\}$: $E \wedge M_i^L \wedge \mathcal{C}(M) \Rightarrow M \vee M_1 \vee M_2$

then: $E \wedge M_1^L \wedge M_2^L \Rightarrow M$.

All in all, the proposal of Amla et al. is, to the best of our knowledge, the first example of compositional rely/guarantee reasoning which presents an abstract framework not relying on an *ad hoc* operator for expressing rely/guarantee specifications, but only exploiting semantic assumptions. Moreover, the proposed rule is a complete one.

4.3. Abstract Non-Rely/Guarantee Methods

This section considers some examples of compositional method that adopt paradigms other than the rely/guarantee one, and therefore cannot be reduced to the general scheme of Table 3.1. In particular, we are considering abstract methods, rather than those tailored for a very specific formal language. As usual, we refer to [Fur05] for more examples and details.

A compositional proof method based on an abstract semantic setting is investigated by de Boer and de Roever in [dBdR98], with reference to assertion networks. Assertion networks are transition diagrams that represent programs and they were first proposed by Floyd [Flo67] for sequential programs. The work presented in [dBdR98] considers the extension of such diagrams to the treatment of concurrent programs.

More precisely, [dBdR98] considers state-based reasoning about concurrent programs that have *synchronous communication*. The method relies on two ingredients:

- compositional inductive assertion networks, that are used to describe the sequential parts of the system;
- compositional proof rules, to deduce global properties of the system, resulting from the concurrent interaction of the local networks.

The basic idea is to introduce auxiliary variables in the assertional description of a network that represent *logical histories*. A logical history simulates the local communication history of some component.

Each component is specified by a Hoare-like triple such as $\{\phi\}P\{\psi\}$, where ϕ, ψ are predicates over the logical histories variables, representing properties of such histories respectively before and after the program P (represented by a synchronous transition diagram) is executed. Finally, the compositional method is based on an inference rule for the parallel composition $P_1 \parallel P_2$ of two programs P_1, P_2. Under certain conditions on the predicates ϕ_i, ψ_i and on the variables involved in the communication channels, the following inference rule is sound and complete for synchronous transition diagrams.

$$\frac{\{\phi_1\}P_1\{\psi_1\}, \{\phi_2\}P_2\{\psi_2\}}{\{\phi_1 \cap \phi_2\}P_1 \parallel P_2\{\psi_1 \cap \psi_2\}}$$

Various other modifications and extensions of the method, to handle different models of concurrency, as well as other methodologies, are thoroughly discussed in the book [dRdBH$^+$01].

The lazy paradigm is an approach to compositionality alternative to the rely/guarantee one, proposed by Shankar in [Sha98].

The motivation for the study of an alternative paradigm stems from the fact that, according to Shankar, the rely/guarantee approach is often difficult to apply in practice, since it requires to anticipate several details of the guarantees of a component so that they are strong enough to permit the discharging of the assumptions of the other modules. On the contrary, one would often prefer to produce a weak compositional specification, adding the necessary details only in later phases of development. Lazy compositionality proposes to solve this problem by simply *lazily* postponing the discharging of the assumptions to later phases of development, when all the necessary details naturally come into the picture.

Under this respect, the lazy approach is more liberal than the rely/guarantee approach, since it does not enforce a pre-defined way of performing verification, but simply allows for the use of conventional techniques and methods. Moreover, lazy composition is mainly a *methodological* approach to compositionality, whereas rely/guarantee has a strong technical connotation. In a nutshell, lazy compositionality combines a *refinement* methodology with a compositional methodology. The result is an approach which

is quite general and can be applied to a large variety of formal specification languages and models. At the same time, its generality can also be a weakness, since the verification burden is implicitly subsumed to other aspects of system analysis, rather than exposed and dealt with directly.

In a hypothetical classification of methods for the verification of large systems, the lazy approach has an "intermediate" degree of compositionality, in between truly compositional approaches (and namely the rely/guarantee approach), and non-compositional (a.k.a. "global") approaches (namely the Owicki-Gries method [OG76] and derived methods). Indeed, we claim that several "non-strictly-compositional" methods fall in this intermediate region, and in particular those described in this section (i.e. Section 4.3).

Shankar elicits the differences between lazy compositionality and rely/guarantee compositionality in five points, namely:

1. components are not treated as black-boxes;
2. composition is not necessarily conjunction;
3. environment assumptions are specified as abstract components (rather than properties);
4. no rely/guarantee proof obligations are generated;
5. composition can yield inconsistent specifications.

Actually, we note that these differences apply only in a typical setting where we specify both a computational model and a formal language to express properties of computations in the model. For example, the "black-box" view referred to in point 1 implies a distinction between the implementation of a module (that is its description in the computational model) and some of its properties (which are described in some logic language, in all the frameworks we have considered). Similarly, "composition as conjunction" of point 2 is, in this case, a characteristic of the computational model, and an "abstract component" (in point 3) is, again, the implementation (in the computational model) of an abstract specification.

However, we are mainly interested in abstract frameworks; for such frameworks only the points 4–5 are relevant. Therefore, let us describe these points in some more detail.

No rely/guarantee proof obligations are generated. In fact, in lazy compositionality the discharging of assumptions is simply lazily postponed to when the necessary specification details are present. More

precisely, the environment assumption is embedded in the specification by means of a component that represents explicitly the abstract environment. The goal of the following refinement steps is to show that the overall system subsumes the abstract environment of that component, thus showing that the explicit environment is superfluous and can be eliminated without affecting the functionality of the component (in fact, the rest of the system does play the role of the assumed environment).

Composition can yield inconsistent specifications. This feature is a consequence of the previous characteristic: since we lazily postpone the discharging of the assumptions, tentatively assuming that they are indeed dischargeable, it may happen that our suppositions show to be false according to any possible refinement of the system. In such cases, the specification is globally inconsistent and we must amend some previous specification choice.

We now finally describe how lazy compositionality is actually carried out, with reference to two modules (the generalization to an arbitrary number of modules is rather straightforward). Let us consider two modules with specifications P_1 and P_2. Note that we do not say anything about how these local specifications should be written; in particular they could also be rely/guarantee specifications, even if we will not use the rely/guarantee *methodology* to treat them. When composing these two modules, we explicitly add to the specification of each module the assumptions that the other module makes on its environment. So, if the two modules assume an environment with properties E_1 and E_2, respectively, then we strengthen P_1 to $P_1 \wedge E_2$ and P_2 to $P_2 \wedge E_1$. Then, we compose these two strengthened modules; in doing so, it is clear that the environment assumptions of each module are satisfied by its actual environment, by construction. More explicitly, in a setting where composition is conjunction, we end up with the system $(E_2 \wedge P_1) \wedge (E_1 \wedge P_2)$, which simply assumes explicitly the validity of the environment assumptions.

Then, the goal of lazy compositional verification, is to show that $(E_2 \wedge P_1) \wedge (E_1 \wedge P_2)$ can be refined to simply $P_1 \wedge P_2$: this would show explicitly that the environment assumptions are satisfied in the system which thus guarantees its functionalities. A drawback of this approach is that a module cannot be refined independently of the others, since each module is enlarged to comprehend the environment assumptions of the other modules, which must also be preserved in the refinement.

All in all, we believe that the liberal lazy approach can be a useful alternative to rely/guarantee compositionality in some cases, even if it basically amounts to shifting the burden of compositional verification to other, standard, methodologies and techniques for verification.

Hooman presents in [Hoo98] a framework to support the top-down design of distributed real-time systems. The framework is based on assertions (i.e. it is denotational) and also *mixed*, that is a specification in the framework can contain both assertions and programming constructs (namely computations).

The framework is semantic, in that it is based on descriptions given by simple *semantic primitives*. Moreover, it is parametric with respect to the time model, which can be, in particular, discrete or continuous. More precisely, the only assumption on the time model is that it is a strictly ordered set. The basic semantic primitive used to describe the behavior of a system (or of a module) is the *observation function*, which is simply a function from the time domain to a set of events: thus, it associates each time instant to the primitive facts that occur at that instant. A *computation* represents a history of a module, and it is constituted by a pair $(\alpha, \mathcal{OBS})$. α is a set of observable events, which are the events that may be observed during the behavior of the component. $\mathcal{OBS}$ is a set of observation functions of the events in α, that is a set of mappings from time to subsets of α (note that we have more than a single observation function, since we allow for nondeterminism).

Two different definitions of parallel composition of two computations are given. Consider two computations $(\alpha_1, \mathcal{OBS}_1)$ and $(\alpha_2, \mathcal{OBS}_2)$. According to the first definition, the composition of the two computations is obtained by taking the union $\alpha_1 \cup \alpha_2$ of the observable events, and the pointwise (i.e. at all time points) union of the observation functions, requiring that these functions agree on the events shared by both components (i.e. which are in $\alpha_1 \cap \alpha_2$). The second definition of parallel composition allows for "open" computations where the observed events can also be not in α: this means that they include arbitrary environment behaviors or, in other words, that $\mathcal{OBS}$ is now a mapping from time to subsets of all the possible events (including those not in α). In this case, the parallel composition is defined simply by the union $\alpha_1 \cup \alpha_2$ of the observable events, and the (pointwise) intersection of the observation functions. The two definitions of parallel composition, although different, are strictly related, and simple conditions,

basically involving the removal of the environment observables, permit to pass from a representation to the other. We indicate parallel composition with the operator //.

In Hooman's framework, a specification is a particular form of computation, defined by a pair $(\alpha, \mathcal{A})$. $\mathcal{A}$ is an assertion, that is a predicate over observation functions. Thus, the specification is a computation which has α as set of observable events, and all observation functions which satisfy the assertion $\mathcal{A}$. Note that no particular structure is given to a specification, and in particular the rely/guarantee paradigm is not adopted. Therefore, parallel composition basically reduces to set intersection, without allowing for mutual dependencies between components. On the other hand, *refinement* between specifications is indicated by the operator $\Rrightarrow$.

Now, the core of the framework consists in formulating a composition rule that permits to deduce soundly the specification of a parallel composition of two specifications. Let us consider two specifications $(\alpha_1, \mathcal{A}_1)$ and $(\alpha_2, \mathcal{A}_2)$. Then, their parallel composition is a refinement of the specification given by the union of the α's and the logical conjunction of the assertions, under a simple condition. Exactly, the soundness condition is that the assertion $\mathcal{A}_i$ only depends on the events in α_i, for $i = 1, 2$; this amounts to requiring that the two specifications are decoupled, and one does not predicate about the behavior of the events that belong to the other. Under this simple condition, the following refinement relation holds:

$$(\alpha_1, \mathcal{A}_1)//(\alpha_2, \mathcal{A}_2) \Rrightarrow (\alpha_1 \cup \alpha_2, \mathcal{A}_1 \wedge \mathcal{A}_2)$$

The paper also briefly discusses the hiding operation to selectively remove observable events from a specification or computation.

As it is clear even from this short presentation, Hooman's framework is very simple and quite general. Nonetheless, even if the semantic primitives are very basic notions, they naturally refer to semantic models such as those based on sequences of events and interleaving semantics (e.g. such as Abadi and Lamport's [AL95]). One advantage of the simplicity of the framework is that it can be easily implemented in a theorem prover. In fact, the paper completely formalizes the framework in PVS [ORS92], and uses the formalization in proving properties of a hybrid system. A less formal version of a very similar framework was proposed by the same author in [Hoo94].

4.4. Other Approaches to Compositionality

This section briefly summarizes approaches to compositionality other than those for descriptive formalisms, which have been the focus of the previous sections. In particular, we give minimal overviews of how the problem of composition arises for model-checking frameworks, for automata-based formalisms, and for Petri nets.

The problem with the scalability of formal methods does not affect only deductive methods, such as those that we focused on elsewhere in this thesis, but it appears also in algorithmic methods, and namely model checking techniques. The limitations of scalability of model checking manifest themselves in the *state explosion problem*: under conditions that happen often in practice, the size of the representation of the parallel composition of modules is exponential in the size of the individual components. Hence, composing modules of manageable sizes yields a global system whose size in unmanageable and cannot be verified automatically.

Compositional techniques try to soothe this problem by permitting the verification of components in isolation, while allowing to infer global properties of the overall system. We avoid a technical analysis of compositional model checking as it would be beyond the scope of this work. Let us refer to the work by Kupferman and Vardi [KV00], which is especially focused on rely/guarantee techniques and algorithmic and complex-theoretical aspects. [KV00] also contains several pointers to related literature on compositionality for model checking.

Just like logic-based formalisms, also automata-based models define a notion of composition among modules. However, whereas the composition semantics is very natural for logic languages, as it is usually reducible to logic conjunction, defining a notion of composition for automata is in general more problematic. One of the problems in achieving such a definition comes from the fact that automata can often be regarded as a synchronous formalism, that is where all the units of the system evolve "at the same time" [FMMR07]. This often implies that the synchronous composition of automata is "less natural" than for logic languages, and it requires one to deal with more technical difficulties.

Nonetheless, composition has been extensively studied for automata-based formalisms for real-time systems, such as timed automata (TA) [AD94] and Timed Input/Output Automata (TIOA) [KLSV06]. Of the two, TIOA are more general than TA: in fact, TA can be regarded as a restricted form of TIOA where automated verification is possible. On the

other hand, TIOA focus on providing a more general framework with a well-defined notion of input/output behavior, composition, and abstraction, to facilitate compositional reasoning. Whereas TA rely mainly on automated model-checking techniques for verification purposes, TIOA are supported by a wider array of tools, including theorem-proving tools [ALL+06], as well as code generators and testing tools [Gar06]. They are being used successfully in dealing with the composition of modules in the description of large systems [HALM06].

Some of the technical difficulties of composition may be soothed if asynchronous composition, where each component evolves independently of the others, is adopted. This is usually the case with timed Petri Nets. Juan and Tsai [JT02], besides briefly surveying a large spectrum of compositional techniques for automata-based formalisms, especially focus on the models of Multiset Labeled Transition Systems and timed Petri Nets. The book also considers experimental aspects, practically comparing some model checking tools on a set of common verification problems.

5. Compositional Inference Rules and Methodology for TRIO

This chapter presents a compositional framework for descriptive logic languages based on axiomatic/deductive techniques, and for the TRIO language in particular. The framework is based on a set of compositional inference rules, which are applicable within a methodology to specify and verify modular systems. On the one hand, the methodology exploits some TRIO language features to allow for an effective and natural way of structuring a large specification into classes. On the other hand, some specific compositional requirements are satisfied by exploiting the inference rules to handle compositional specifications written according to the rely/guarantee paradigm and presenting circular references.

We discussed in Section 3.2 why compositional inference rules need not be circular to be of practical interest. Some works reviewed in Chapter 4 give other evidence to this fact. Therefore, we present both non-circular inference rules (in Section 5.1) and circular inference rules (in Section 5.2). In each case, the user should choose the style of rule which is most appropriate for the system under development. We complement the presentation of each rule with an analysis of its completeness, as well as general considerations about features that permit or prevent completeness.

For simplicity of presentation, the inference rules of Sections 5.1 and 5.2 are referred to systems composed of two modules. Afterward, the same rules are generalized to systems of $N > 2$ modules in Section 5.3.

Finally, Section 5.4 shows how the previously introduced compositional rules can be exploited in practice. Namely, a methodology that prescribes how to tackle the compositional specification and verification of a modular system is formally presented. The following Chapter 6 will demonstrate the compositional framework in practice on two significant examples.

5.1. Non-Circular Compositional Inference Rules

This section presents two non-circular compositional inference rules, and proves their soundness and completeness. For the sake of clarity, let us briefly recall the notation we introduced in Section 3.3.1:

- we consider a system made of $N \geq 2$ modules;
- to each module $i = 1, \ldots, N$, we associate a local assumption E_i, and a local guarantee M_i;
- the N modules are composed (and connected) to form a global system;
- the global system has its own global assumption E, and its global guarantee M.

5.1.1. Non-Circular Inference Rules

For the sake of simplicity, let us first consider simple systems consisting of two modules only. The extension to the general case of $N \geq 2$ modules will be discussed separately, in Section 5.3.

First of all, the non-circular rules that we present here use regular implication as a compositional operator, as they are simple consequences of the logical rules of implication and conjunction. Even more generally, we are able to present their proofs in terms of two generic logical connectives denoted as $\sqsubseteq$ and $\sqcap$. $\sqsubseteq$ represents any operator which is an order relation (i.e., it is reflexive, anti-symmetric and transitive), while the $\sqcap$ connector represents an associative, commutative and idempotent operation which respects the order relation (i.e., such that if $A \sqsubseteq C$ and $B \sqsubseteq D$, then also $A \sqcap B \sqsubseteq C \sqcap D$ for any A, B, C, D). It is immediate to realize that logical implication $\Rightarrow$ and conjunction $\wedge$ respectively satisfy such properties.

Therefore, we have the two following proposition, that establish the soundness of the corresponding two compositional inference rules.

Proposition 5.1.1 (Non-Circular Rely/Guarantee Inference Rule 1)**.** *If:*

1. $E_2 \sqsubseteq M_2$
2. $E \sqcap M_1 \sqsubseteq E_2$
3. $E \sqcap M_1 \sqcap M_2 \sqsubseteq M$

4. $E \sqsubseteq M_1$

then, $E \sqsubseteq M$

Proof. The following formal derivation proves the proposition.

$*$)	$E \sqcap M_1 \sqsubseteq M_2$	by 2, 1 and transitivity of $\sqsubseteq$
$**$)	$E \sqsubseteq E$	by reflexivity of $\sqsubseteq$
	$E \sqcap E \sqsubseteq M_1 \sqcap E$	by 4, the previous one and the order-preservation of $\sqcap$
	$E \sqsubseteq M_1 \sqcap E$	by the previous one and the idempotence of $\sqcap$
	$E \sqsubseteq E \sqcap M_1$	by the previous one and the commutativity of $\sqcap$
	$E \sqsubseteq M_2$	by the previous one, $*$) and the transitivity of $\sqsubseteq$
	$E \sqcap E \sqsubseteq M_1 \sqcap M_2$	by 4, the previous one, and the order-preservation of $\sqcap$
	$E \sqsubseteq M_1 \sqcap M_2$	by the previous one and the idempotence of $\sqcap$
	$E \sqcap E \sqsubseteq E \sqcap M_1 \sqcap M_2$	by $**$), the previous one and the order-preservation of $\sqcap$
	$E \sqsubseteq E \sqcap M_1 \sqcap M_2$	by the previous one and the idempotence of $\sqcap$
	$E \sqsubseteq M$	by the previous one, 3 and transitivity of $\sqsubseteq$ $\square$

Notice that the rule of Proposition 5.1.1 is non-fully compositional, according to the definitions introduced in Section 3.3.2; instead, the following is.

Proposition 5.1.2 (Non-Circular Rely/Guarantee Inference Rule 2)**.** *If:*

1. *a)* $E_1 \sqsubseteq M_1$
   b) $E_2 \sqsubseteq M_2$

2. $M_1 \sqsubseteq E_2$

3. $E \sqcap M_1 \sqcap M_2 \sqsubseteq M$

4. $E \sqsubseteq E_1$

then, $E \sqsubseteq M$

Proof. The proof is very similar to that of Proposition 5.1.1. In brief, just assume E holds; then, E_1 by 4, M_1 by 1a, E_2 by 2, M_2 by 1b, M by 3. □

We remark again that, however simple the two above non-circular rules may seem, they are those used in practice in a lot of formal reasoning about composite systems. Indeed, their simplicity and wide applicability show that what we call "compositional reasoning" is naturally embedded in mathematical logic reasoning [Lam98].

5.1.2. Completeness of the Non-Circular Rules

Let us show that the inference rules of Propositions 5.1.1 and 5.1.2 are relatively complete. We need an extra assumption on the set of formulas over which the order relation $\sqsubseteq$ is defined: we require that there exists a *maximum element* in the ordered set, indicated as $\top$, that is a formula such that for any formula P, $P \sqsubseteq \top$ is true.[1] Moreover, $\top$ must be an identity element for the $\sqcap$ operation, that is for any formula P, $P \sqcap \top = \top \sqcap P = P$.

Notice that these assumptions are obviously satisfied by the logical value *true*, with respect to the logical implication $\Rightarrow$ and conjunction $\wedge$ respectively.

Theorem 5.1.3 (Completeness of Non-Circular Inference Rules)**.** *The non-circular inference rules in Proposition 5.1.1 and in Proposition 5.1.2 are relatively complete.*

Proof. Let us first consider Proposition 5.1.1. Assume $E \sqsubseteq M$ holds. Hence choose $M_1 = M$ and $E_2 = M_2 = \top$. Hypotheses 1 and 2 are trivially true because of the definition of $\top$ as maximum element of the order. Hypothesis 4 is $E \sqsubseteq M_1$, which follows from $E \sqsubseteq M$, $M \sqsubseteq M_1$ (since $M = M_1$) and transitivity of $\sqsubseteq$. Finally, hypothesis 3 is $E \sqcap M_1 \sqcap M_2 \sqsubseteq M$, which corresponds to $E \sqcap M \sqcap \top \sqsubseteq M$ because of the equivalences, and to $E \sqcap M \sqsubseteq M$ because of definition of identity. Now, since $E \sqsubseteq M$ and $M \sqsubseteq M$, the last hypothesis 3 also holds.

Let us now consider Proposition 5.1.2 and assume $E \sqsubseteq M$. If we choose $E_1 = E$, $M_1 = M$ and $M_2 = E_2 = \top$, conditions 1–4 correspond to:

1. a) $E \sqsubseteq M$, assumed true.
   b) $\top \sqsubseteq \top$, trivially true.

[1] Recall that $P = Q$ iff $P \sqsubseteq Q$ and $Q \sqsubseteq P$

2. $E \sqcap M \sqsubseteq \top$, trivially true.

3. $E \sqcap M \sqcap \top \sqsubseteq M$, equivalent to $E \sqsubseteq M$, assumed true.

4. $E \sqsubseteq E$, true by reflexivity of $\sqsubseteq$. □

5.1.3. Summary of Non-Circular Rules

Table 5.1 summarizes the two non-circular compositional inference rules we have presented above; for uniformity with the results of the following sections, we use implication and conjunction in place of the generic $\sqsubseteq$ and $\sqcap$ operators.

Rule 1 (Prop. 5.1.1)	**Rule 2 (Prop. 5.1.2)**
$E_2 \Rightarrow M_2$	$E_1 \Rightarrow M_1$, $E_2 \Rightarrow M_2$
$E \wedge M_1 \Rightarrow E_2$	$M_1 \Rightarrow E_2$
$E \wedge M_1 \wedge M_2 \Rightarrow M$	$E \wedge M_1 \wedge M_2 \Rightarrow M$
$E \Rightarrow M_1$	$E \Rightarrow E_1$
$E \Rightarrow M$	$E \Rightarrow M$

Table 5.1.: Non-circular compositional inference rules for two modules.

5.2. Circular Compositional Inference Rules

This section presents several circular compositional inference rules, proves their soundness, and discusses their completeness. More precisely, we are going to start in Section 5.2.1 by introducing a compositional operator $\twoheadrightarrow$ and present fully and non-fully compositional rules that use this operator. Then, Section 5.2.2 explores variations of the $\twoheadrightarrow$ that may be more suited to model certain classes of systems; inference rules for these variations are derived from the analogous ones in Section 5.2.1. Finally, Section 5.2.3 studies the completeness of the previously introduced compositional rules.

5.2.1. The Time Progression Compositional Operator

The Time Progression Operator

Let us introduce a simple compositional operator that allows us to define sound circular inference rules. It is called *time progression* operator, and

it is denoted by the symbol $\twoheadrightarrow$. Informally, $P \twoheadrightarrow Q$ is true for two time-dependent formulas P, Q if, whenever P has been true in the immediate past, then Q has also been true in the immediate past, is true now, and will continue to hold for some time in the immediate future. This is formalized by the following definition.

$$P \twoheadrightarrow Q \equiv \begin{cases} \text{UpToNow}(P) \Rightarrow \text{UpToNow}(Q) \wedge Q \wedge \text{NowOn}(Q) & \text{if } \mathbb{T} \text{ dense} \\ \text{UpToNow}(P) \Rightarrow \text{UpToNow}(Q) \wedge Q & \text{if } \mathbb{T} \text{ discrete} \end{cases}$$

Thus, the rely/guarantee specifications in our inference rules are in the form $E \twoheadrightarrow M$. Notice that this is a reasonable way to express a rely/guarantee specification: in fact, we say that the behavior of the module in the immediate future is influenced only by the behavior of the environment in the immediate past, so that if the environment stops behaving correctly, then the module can also stop behaving correctly only after "a while".

From a technical viewpoint, notice that the semantics of the time progression operator allows us to build a sort of "temporal induction" over the timeline, where the inductive step is allowed by the $Q \wedge \text{NowOn}(Q)$ part that "makes time progress" past the current instant (hence, the name). This will become apparent in the soundness proofs of the inference rules that use this operator.

Finally, let us also remark that a different operator was also named *time progression* and denoted with the same symbol in [FRMM07]. In Section 5.2.2 below we will show how the inference rule for that operator, also presented in [FRMM07], can be derived from those for the time progression operator introduced here.

In the remainder, recall that all TRIO formulas are implicitly universally quantified over time, that is closed with an Alw operator. In particular, this is the case for the hypotheses and conclusions of the rules that we are presenting.

Circular Inference Rules for the Time Progression Operator

Let us now present and prove the soundness of two circular inference rules for the time progression operator. The first rule is given in the following proposition; according to our taxonomy, it is a fully-compositional, non self-discharging, globally initialized, circular inference rule.

Let us also remark that, in practice, the initialization predicate S would be modeled in TRIO as a *unique event*, i.e., an event which happens exactly

once in time. This is rendered by the two formulas $\mathrm{Som}(S)$ and $S \Rightarrow \mathrm{AlwP}(\neg S) \wedge \mathrm{AlwF}(\neg S)$. However, the soundness of this inference rule, as well as that of the others that we present in the remainder of this chapter, do not depend on S being a unique event.

Proposition 5.2.1 (Rely/Guarantee Circular Inference Rule 1)**.** *If:*

1. *a)* $E_1 \twoheadrightarrow M_1$
   b) $E_2 \twoheadrightarrow M_2$
2. *a)* $M_1 \Rightarrow E_2$
   b) $M_2 \Rightarrow E_1$
3. $M_1 \wedge M_2 \Rightarrow M$
4. $S \Rightarrow \mathrm{UpToNow}(E_i) \vee \mathrm{UpToNow}(M_i)$*, for some* $i \in \{1, 2\}$

then $\mathrm{SomP_i}(S) \Rightarrow (E \twoheadrightarrow M)$.

Proof for dense time domains. Let t be the current instant, and let us assume that $\mathrm{SomP_i}(S)$ holds at t; thus, let $t' \leq t$ be (any) instant at which S held. We have to show that $E \twoheadrightarrow M$ holds at t.

First of all, let us prove that $\mathrm{UpToNow}(M_1 \wedge M_2)$, $M_1 \wedge M_2$, and $\mathrm{NowOn}(M_1 \wedge M_2)$ hold at t. To this end, we know that at t' we have $\mathrm{UpToNow}(E_i)$ or $\mathrm{UpToNow}(M_i)$ for some i, because of (4). Let us assume that $\mathrm{UpToNow}(E_i)$; this is without loss of generality, as if $\mathrm{UpToNow}(M_i)$, then (2) lets us conclude immediately that also $\mathrm{UpToNow}\big(E_{\hat{i}}\big)$ at t', where we adopt the convention of denoting by $\hat{i}$ the "other" index $\{1, 2\} \setminus \{i\} \ni \hat{i}$. Then, let us consider the consequences of (1) at t': since $\mathrm{UpToNow}(E_i)$, then $\mathrm{UpToNow}(M_i)$, M_i, and $\mathrm{NowOn}(M_i)$ also at t'. Thus, let us consider (1) and (2) again and see that also $\mathrm{UpToNow}\big(E_{\hat{i}}\big)$, $\mathrm{UpToNow}\big(M_{\hat{i}}\big)$, $E_{\hat{i}} \wedge M_{\hat{i}}$, $\mathrm{NowOn}\big(E_{\hat{i}}\big)$, and $\mathrm{NowOn}\big(M_{\hat{i}}\big)$.

Therefore, we have that $M_1 \wedge M_2$ holds until some instant $t'' > t'$ in the future w.r.t. t'; in other words, we can say that $\mathrm{UpToNow}(M_1 \wedge M_2)$ holds at t''. But then, $\mathrm{UpToNow}(E_1 \wedge M_2)$ holds at t'', because of (2). It is not difficult to see that we can repeat everything that we did at t' again at t''. Thus, we will have a strictly monotonic sequence of points $t' < t'' < t''' < \cdots$ such that $M_1 \wedge M_2$ holds throughout.

Next, we have to show that the sequence gets (at least) until a point $u > t$. We show this by contradiction: assume to the contrary that $M_1 \wedge M_2$

holds until point $\bar{t} \leq t$ only; that is NowOn$(M_1 \wedge M_2)$ is false at $\bar{t}$. Therefore, UpToNow$(M_1 \wedge M_2)$ holds at $\bar{t}$. Then, (2) lets us deduce that also UpToNow$(E_1 \wedge E_2)$ holds at $\bar{t}$. But then we consider (1) twice, once for module 1 and once for module 2, to conclude that both $M_1 \wedge M_2$ and NowOn$(M_1 \wedge M_2)$ hold. This is in contradiction with the hypothesis that $M_1 \wedge M_2$ held up until $\bar{t}$, as expected.

All in all, we have shown that $M_1 \wedge M_2$ hold from before t' until some $u > t$. Therefore, in particular UpToNow$(M_1 \wedge M_2)$, $M_1 \wedge M_2$, and NowOn$(M_1 \wedge M_2)$ all hold at t.

Finally, from (3) we easily infer that UpToNow$(M) \wedge M \wedge$ NowOn(M) at t. □

Proof for discrete time domains. We present a proof along the lines of the one for dense time domains. Alternatively, one could use induction to prove the same result for discrete time.

Let t be the current instant, and let us assume that SomPi(S) holds at t; thus, let $t' \leq t$ be (any) instant at which S held. We have to show that $E \twoheadrightarrow M$ holds at t.

First of all, let us prove that UpToNow$(M_1 \wedge M_2)$ and $M_1 \wedge M_2$ hold at t. To this end, we know that at t' we have UpToNow(E_i) or UpToNow(M_i) for some i, because of (4). Let us assume that UpToNow(E_i); this is without loss of generality, as if UpToNow(M_i), then (2) lets us conclude immediately that also UpToNow$\big(E_{\hat{i}}\big)$ at t', where we adopt the convention of denoting by $\hat{i}$ the "other" index $\{1,2\} \setminus \{i\} \ni \hat{i}$. Then, let us consider the consequences of (1) at t': since UpToNow(E_i), then UpToNow(M_i) and M_i also at t'. Thus, let us consider (1) and (2) again and see that also UpToNow$\big(E_{\hat{i}}\big)$, UpToNow$\big(M_{\hat{i}}\big)$, $E_{\hat{i}}$, and $M_{\hat{i}}$.

Therefore, we have that $M_1 \wedge M_2$ holds until $t'' = t' + 1$ included; in other words, we can say that UpToNow$(M_1 \wedge M_2)$ holds at $t'' + 1$. But then, UpToNow$(E_1 \wedge M_2)$ holds at $t'' + 1$, because of (2). It is not difficult to see that we can repeat everything that we did at t' again at $t'' + 1$. Thus, we will have a strictly monotonic sequence of points $t' < t' + 1 < t' + 2 < \cdots$ such that $M_1 \wedge M_2$ holds throughout.

Next, we have to show that the sequence gets (at least) until a point $u > t$. We show this by contradiction: assume to the contrary that $M_1 \wedge M_2$ holds until point $\bar{t} \leq t$ included only; that is NowOn$(M_1 \wedge M_2)$ is false at $\bar{t}$. Therefore, UpToNow$(M_1 \wedge M_2)$ holds at $\bar{t}+1$. Then, (2) lets us deduce that also
UpToNow$(E_1 \wedge E_2)$ holds at $\bar{t}+1$. But then we consider (1) twice, once for

module 1 and once for module 2, to conclude in particular that $M_1 \wedge M_2$. This is in contradiction with the hypothesis that $M_1 \wedge M_2$ held up until $\bar{t}$, as expected.

All in all, we have shown that $M_1 \wedge M_2$ hold from $t' - 1$ until some $u > t$. Therefore, in particular $\mathrm{UpToNow}(M_1 \wedge M_2)$ and $M_1 \wedge M_2$ all hold at t.

Finally, from (3) we easily infer that $\mathrm{UpToNow}(M) \wedge M$ at t. □

Notice that in the above proofs we never actually introduced any fact about the value of E; in other words it is as if we had proved the conclusion true $\twoheadrightarrow M$. Indeed, in this rule — and in some of the following ones — the global assumption E is introduced in the conclusion only in conformance with the other rules; in other words we make no assumptions on the global environment. On the contrary, other rules actually require E to hold in order to be sound; this is the case, for instance, of the rules of Proposition 5.2.4 and Proposition 5.2.7, which we will present in the following sections.

The second rule we present is non-fully compositional, self-discharging, globally initialized, and circular. Notice that our choice of compositional operator, combined with the fact that the rule is non-fully compositional, requires to add a term in the conclusion that requires the global assumption E to hold "always in the past". Otherwise (i.e., if E was false somewhere in the past), it would be impossible, in general, to carry out the discharging of local assumptions, since the rule is non-fully compositional. We prove the soundness of the rule only for dense time models; the proof for discrete time models can be developed along the same lines.

Proposition 5.2.2 (Rely/Guarantee Circular Inference Rule 2)**.** *If:*

1. *a)* $E_1 \twoheadrightarrow M_1$
   b) $E_2 \twoheadrightarrow M_2$
2. $E \wedge M_1 \wedge M_2 \Rightarrow E_1 \wedge E_2$
3. $M_1 \wedge M_2 \Rightarrow M$
4. $S \Rightarrow \mathrm{UpToNow}(E_1 \wedge E_2) \vee \mathrm{UpToNow}(M_1 \wedge M_2)$

then $\mathrm{SomP_i}(S) \wedge \mathrm{AlwP_e}(E) \Rightarrow (E \twoheadrightarrow M)$.

Proof for dense time domains. Let t be the current instant, and let us assume that $\mathrm{SomP_i}(S)$ and $\mathrm{AlwP_e}(E)$ hold at t. We show that $E \twoheadrightarrow M$ holds

at t; to this end, let us first show that UpToNow$(M_1 \wedge M_2)$, $M_1 \wedge M_2$, and NowOn$(M_1 \wedge M_2)$ hold at t.

Let $t' \leq t$ be an instant at which S holds. From (4), let us assume that UpToNow$(E_1 \wedge E_2)$ holds at t'. This is without loss of generality, since if UpToNow$(M_1 \wedge M_2)$, then also UpToNow$(E_1 \wedge E_2)$ at the same time, from (2) and the fact that AlwP$_{\mathrm{e}}(E)$ at $t \geq t'$.

Then, let us consider (1) at t'. We can infer that UpToNow$(M_1 \wedge M_2)$, $M_1 \wedge M_2$, and NowOn$(M_1 \wedge M_2)$ all hold at t'. Therefore, $M_1 \wedge M_2$ holds until some $t'' > t'$. If $t'' > t$, we are done proving the current goal; otherwise, we can iterate the reasoning and get to a new point $t''' > t''$.

The sequence of points $t' < t'' < t''' < \cdots$ must eventually reach a point $u > t$. The proof by contradiction goes just as in the case of the proof of Proposition 5.2.1.

So, finally we have that UpToNow$(M_1 \wedge M_2)$, $M_1 \wedge M_2$, and NowOn$(M_1 \wedge M_2)$ hold at t. The final step infers that also UpToNow(M), M, and NowOn(M) from (3). Thus, we have shown that $E \twoheadrightarrow M$ holds at t. □

Locally Initialized Circular Inference Rules

Let us now provide *locally initialized* variations of the two circular inference rules introduced above. They will be useful in discussing the completeness of the inference rules (see Section 5.2.3). Recall that, in locally initialized rules, the use of the predicate S "simulates" an origin on the time axis.

Notice that, in order to retain soundness, we have to introduce two small changes, other than local initializations. First, the initialization predicate in the conclusion excludes the current instant for S to occur; second, the initialization condition now requires an interval *in the future* where a local assumption or guarantee holds. This is required to ensure that all predicates are evaluated after the system has started, so that the locally initialized specification, dischargings, and global implementation formulas hold there.

Proposition 5.2.3 (Rely/Guarantee Circular Inference Rule 1(bis)). *If:*

1. *a)* SomP$_{\mathrm{e}}(S) \Rightarrow (E_1 \twoheadrightarrow M_1)$
   b) SomP$_{\mathrm{e}}(S) \Rightarrow (E_2 \twoheadrightarrow M_2)$
2. *a)* SomP$_{\mathrm{e}}(S) \Rightarrow (M_1 \Rightarrow E_2)$
   b) SomP$_{\mathrm{e}}(S) \Rightarrow (M_2 \Rightarrow E_1)$

3. $\mathrm{SomP_e}(S) \Rightarrow (M_1 \wedge M_2 \Rightarrow M)$

4. $S \Rightarrow \mathrm{NowOn}(E_i) \vee \mathrm{NowOn}(M_i)$, *for some* $i \in \{1, 2\}$

then $\mathrm{SomP_e}(S) \Rightarrow (E \twoheadrightarrow M)$.

Proof for dense time domains. Let t be the current instant, and let us assume that $\mathrm{SomP_e}(S)$ holds at t. To show that $E \twoheadrightarrow M$ holds at t, let us first show that $\mathrm{UpToNow}(M_1 \wedge M_2)$, $M_1 \wedge M_2$, and $\mathrm{NowOn}(M_1 \wedge M_2)$ hold at t.

Let $t' < t$ be an instant at which S holds. From (4), let us assume that $\mathrm{NowOn}(E_i)$ holds at t'. This is without loss of generality, since if $\mathrm{NowOn}(M_i)$, then also $\mathrm{NowOn}\big(E_{\hat{i}}\big)$ at the same time, from (2) and the fact that in the right-neighborhood of t' $\mathrm{SomP_e}(S)$ holds. Since $\mathrm{NowOn}(E_i)$ at t', then equivalently $\mathrm{UpToNow}(E_i)$ holds at some $t' + \epsilon$, for some $\epsilon > 0$.

Then, let us consider (1) at $t' + \epsilon$. We can infer that $\mathrm{UpToNow}(M_i)$, M_i, and $\mathrm{NowOn}(M_i)$ all hold at $t' + \epsilon$. Then, also $\mathrm{UpToNow}\big(E_{\hat{i}}\big)$ holds at $t' + \epsilon$ from (2); but then, by (1) again, also $\mathrm{UpToNow}\big(M_{\hat{i}}\big)$, $M_{\hat{i}}$, and $\mathrm{NowOn}\big(M_{\hat{i}}\big)$ all hold at $t' + \epsilon$.

At this point, the proof goes on exactly as for Proposition 5.2.1. In particular, we infer that $\mathrm{UpToNow}(M_1 \wedge M_2)$, $M_1 \wedge M_2$, and $\mathrm{NowOn}(M_1 \wedge M_2)$ are true at t, and we conclude that $E \twoheadrightarrow M$ by applying (3). □

Similar modifications are done to get a locally initialized circular inference rule analogous to that of Proposition 5.2.2. We omit the proof, which is however all similar to the one we have just provided.

Proposition 5.2.4 (Rely/Guarantee Circular Inference Rule 2(bis))**.** *If:*

1. *a)* $\mathrm{SomP_e}(S) \wedge \mathrm{AlwP_e}(E) \Rightarrow (E_1 \twoheadrightarrow M_1)$
   b) $\mathrm{SomP_e}(S) \wedge \mathrm{AlwP_e}(E) \Rightarrow (E_2 \twoheadrightarrow M_2)$

2. $\mathrm{SomP_e}(S) \wedge \mathrm{AlwP_e}(E) \Rightarrow (E \wedge M_1 \wedge M_2 \Rightarrow E_1 \wedge E_2)$

3. $\mathrm{SomP_e}(S) \wedge \mathrm{AlwP_e}(E) \Rightarrow (M_1 \wedge M_2 \Rightarrow M)$

4. $S \Rightarrow \mathrm{NowOn}(E_1 \wedge E_2) \vee \mathrm{NowOn}(M_1 \wedge M_2)$

then $\mathrm{SomP_e}(S) \wedge \mathrm{AlwP_e}(E) \Rightarrow (E \twoheadrightarrow M)$.

Finally, we present one more variation of the rule of Proposition 5.2.1. In this case, we write the global implementation formula in terms of the

$\twoheadrightarrow$ operator, rather than using simple implication. We may argue that the original Proposition 5.2.1 is probably in a form which is more likely to be applicable in practice, whereas the following rule overloads one hypothesis with a large share of the complexity involved in compositional reasoning. Nonetheless, we will show that the following strengthening allows us to have a *complete* inference rule, and in any case it may be useful in practice with systems where the link between local and global guarantees is more involved than simple implication.

Proposition 5.2.5 (Rely/Guarantee Circular Inference Rule 1(ter))**.** *If:*

1. *a)* $E_1 \twoheadrightarrow M_1$
   b) $E_2 \twoheadrightarrow M_2$
2. *a)* $M_1 \Rightarrow E_2$
   b) $M_2 \Rightarrow E_1$
3. $E \wedge M_1 \wedge M_2 \twoheadrightarrow M$
4. $S \Rightarrow \mathrm{UpToNow}(E_i) \vee \mathrm{UpToNow}(M_i)$, *for some* $i \in \{1, 2\}$

then $\mathrm{SomP_i}(S) \Rightarrow (E \twoheadrightarrow M)$.

Proof. The proof is exactly the same as the one of Proposition 5.2.1 until we have established that $\mathrm{UpToNow}(M_1 \wedge M_2)$ and $M_1 \wedge M_2$ all hold at the current instant t. Then, if also $\mathrm{UpToNow}(E)$ then clearly we infer that $\mathrm{UpToNow}(M) \wedge M$ at t, from (3). This concludes the proof. □

5.2.2. Other Compositional Operators and Compositional Rules

This section provides other circular compositional inference rules, exploring different compositional operators and other different features.

A Stronger Time Progression Operator

Let us provide a compositional inference rule very similar to that discussed and introduced in [FRMM07]. The rule exploits another compositional operator — also called "time progression" in [FRMM07] — that we denote with the symbol $\gg$ here. Its semantics is the following.

$$P \gg Q \equiv \begin{cases} \mathrm{AlwP_e}(P) \Rightarrow \mathrm{AlwP_i}(Q) \wedge \mathrm{NowOn}(Q) & \text{if } \mathbb{T} \text{ dense} \\ \mathrm{AlwP_e}(P) \Rightarrow \mathrm{AlwP_i}(Q) & \text{if } \mathbb{T} \text{ discrete} \end{cases}$$

Notice that $\gg$ is stronger than $\twoheadrightarrow$ in the following sense: if $P \twoheadrightarrow Q$ holds over the whole time axis, then $P \gg Q$ also holds, while the converse is not true, in general. In fact, if for instance P holds exactly over $(0, 10)$ and Q is always false, then it is true that $P \gg Q$ everywhere, as $\mathrm{AlwP_e}(P)$ is trivially false, but $P \twoheadrightarrow Q$ is false, as Q should hold over some set $(0, 10+\epsilon)$, for some $\epsilon > 0$.

Despite $\gg$ being stronger than $\twoheadrightarrow$ in the above sense, we cannot develop a valid inference rule for the new operator by simply replacing the time progression operator in Proposition 5.2.1 or 5.2.2 with the new compositional operator. In fact, in general we also have to change the initialization condition with one which "triggers" the application of the new operator. In particular, the $\gg$ operator requires its left-hand argument to hold over always in the past from the current instant; therefore, we require that at the system initialization either some local assumption or some local guarantee are true always in the past. Thus, we obtain an inference rule which is a slight variation of the one presented in [FRMM07] for the $\gg$ operator.

Proposition 5.2.6 (Rely/Guarantee Circular Inference Rule 3)**.** *If:*

1. *a)* $E_1 \gg M_1$
   b) $E_2 \gg M_2$
2. *a)* $M_1 \Rightarrow E_2$
   b) $M_2 \Rightarrow E_1$
3. $M_1 \wedge M_2 \Rightarrow M$
4. $S \Rightarrow \mathrm{AlwP_e}(E_i) \vee \mathrm{AlwP_e}(M_i)$, *for some* $i \in \{1, 2\}$

then $\mathrm{SomP_i}(S) \Rightarrow (E \gg M)$.

Proof sketch for dense time domains. Let us just sketch the beginning of the proof: the remainder is all similar to the previous proofs, as well as to that presented in [FRMM07].

Let t be the current instant and $t' \leq t$ be an instant at which S held. Then, without loss of generality we can assume that $\mathrm{AlwP_e}(E_i)$ at t'; otherwise, it would be $\mathrm{AlwP_e}(M_i)$, but then $\mathrm{AlwP_e}\big(E_{\hat{i}}\big)$ would follow from (2). Then, from (1) we infer that $\mathrm{AlwP_i}(M_i)$ and $\mathrm{NowOn}(M_i)$ at t'. Moreover, from (2) and (1) it is simple to deduce that also $\mathrm{AlwP_i}\big(E_{\hat{i}} \wedge M_{\hat{i}}\big)$ and $\mathrm{NowOn}\big(E_{\hat{i}} \wedge M_{\hat{i}}\big)$. All in all we have "advanced" until some time $t'' > t'$.

Then, the proof proceeds by the usual non accumulation argument. When we have finally shown that $\mathrm{AlwP_i}(M_1 \wedge M_2) \wedge \mathrm{NowOn}(M_1 \wedge M_2)$ at t, then $E \gg M$ follows straight from (3). □

Implication as a Compositional Operator

As we also discussed elsewhere [FRMM07], there are basically two ways to make a circular inference rule sound. One relies on writing specifications using an *ad hoc* operator that allows one to "propagate" the validity of some predicate over time, thus allowing a sort of temporal induction; this is the way we have followed with the time progression operator or, more generally, with other compositional operators. The other way to achieve soundness in presence of circularity is to rely on semantic properties of the underlying model or, equivalently, of the formulas that describe that model.

We follow this second way with the following inference rule. Rather than using an *ad hoc* compositional operator, we simply write rely/guarantee guarantee specifications using implication $\Rightarrow$ to link assumption and guarantee of a module. On the other hand, we introduce assumptions on the behavior of the local assumptions and guarantee.

Recall the definitions of right-continuous and left-continuous states, given in Section 2.1.4. For a time-dependent predicate P, we denote by $P \in \mathsf{ST\_}$ the fact that it is a state, by $P \in \mathsf{ST\_}_{\bullet}$ the fact that it is a left-continuous state, and by $P \in \mathsf{ST}_{\bullet}\mathsf{\_}$ the fact that it is a right-continuous state. Also notice that a state which is both left-continuous and right-continuous is constant over time.

The inference rule of the following proposition requires that a module has a local assumption or local guarantee that behaves as a left-continuous state, whereas the other module has a local assumption or local guarantee that behaves as a right-continuous state. These two facts combined allows us to substitute an "alternation" of left- and right-continuous predicates to the use of a time progression operator, in order to set up an induction over time. In a sense, the interplay between left- and right-continuous states and the way they are logically implied in the local specifications results in the constancy over time of the local specifications. Compare this rule to the one in Proposition 5.2.1, where no semantic assumptions are made on the local formulas, but a compositional operator other than $\Rightarrow$ is used.

Proposition 5.2.7 (Rely/Guarantee Circular Inference Rule 4)**.** *If:*

1. *a)* $E_1 \Rightarrow M_1$

b) $E_2 \Rightarrow M_2$

2. *a)* $M_1 \Rightarrow E_2$
   b) $M_2 \Rightarrow E_1$
3. $E \wedge M_1 \wedge M_2 \Rightarrow M$
4. $S \Rightarrow \mathrm{UpToNow}(E_i) \vee \mathrm{UpToNow}(M_i)$, *for some* $i \in \{1, 2\}$
5. $E_1 \in \mathsf{ST}_{\bullet\leftarrow}$ *or* $M_1 \in \mathsf{ST}_{\bullet\leftarrow}$ [2]
6. $E_2 \in \mathsf{ST}_{\rightarrow\bullet}$ *or* $M_2 \in \mathsf{ST}_{\rightarrow\bullet}$

then $\mathrm{SomP_i}(S) \Rightarrow (E \Rightarrow M)$.

Proof for dense time domains. Let t be the current instant, and let us assume that $\mathrm{SomP_i}(S)$ holds at t; thus, let $t' \leq t$ be (any) instant at which S held. We have to show that $E \Rightarrow M$ holds at t.

As usual, we focus on showing one "temporal inductive step", the non accumulation argument being all similar to that of the previous proofs. More precisely, let us show that $E_1 \wedge M_1 \wedge E_2 \wedge M_2$ and $\mathrm{NowOn}(E_1 \wedge M_1 \wedge E_2 \wedge M_2)$ hold at t'.

We assume that $\mathrm{UpToNow}(E_1)$ holds at t'. Indeed, this is without loss of generality: in fact, if $\mathrm{UpToNow}(M_2)$ then $\mathrm{UpToNow}(E_1)$ from (2b); if $\mathrm{UpToNow}(E_2)$ then $\mathrm{UpToNow}(M_2)$ from (1b); if $\mathrm{UpToNow}(M_1)$ then $\mathrm{UpToNow}(E_2)$ from (2a). But then, notice that $\mathrm{UpToNow}(E_1)$ at t' implies $\mathrm{UpToNow}(M_1)$ at t' from (1a), and thus all in all we have $\mathrm{UpToNow}(E_1 \wedge M_1 \wedge E_2 \wedge M_2)$ at t'.

Let us now consider (6), and let us distinguish two cases.

- If $E_2 \in \mathsf{ST}_{\rightarrow\bullet}$, then E_2 holds at t'. Therefore, also M_2 holds at t' for (1b); but then, (2b) implies that E_1 also holds, and finally (1a) implies that M_1 also holds at t'.
- If $M_2 \in \mathsf{ST}_{\rightarrow\bullet}$, then M_2 holds at t'. Therefore, also E_1 holds at t' for (2b); but then, (1a) implies that M_1 also holds, and finally (2a) implies that E_2 also holds at t'.

All in all, $E_1 \wedge M_1 \wedge E_2 \wedge M_2$ holds at t'.

The next step is to consider (5) and still distinguish two cases.

[2] Obviously, fixing which module is described by a right-continuous state is without loss of generality.

- If $E_1 \in \mathsf{ST}_{\bullet\leftarrow}$, then NowOn($E_1$) holds at t'. Therefore, also NowOn(M_1) holds at t' for (1a); but then, (2a) implies that NowOn(E_2) also holds, and finally (1b) implies that NowOn(M_2) also holds at t'.

- If $M_1 \in \mathsf{ST}_{\bullet\leftarrow}$, then NowOn($M_1$) holds at t'. Therefore, also NowOn(E_2) holds at t' for (2a); but then, (1b) implies that NowOn(M_2) also holds, and finally (2b) implies that NowOn(E_1) also holds at t'.

All in all, we have shown that $E_1 \wedge M_1 \wedge E_2 \wedge M_2$ and NowOn($E_1 \wedge M_2 \wedge E_2 \wedge M_2$) at t'. The remainder of the proof is as for the other propositions. □

Since in the above proof we never evaluated hypothesis (3) at instants in which $\mathrm{SomP_i}(S)$ is false, we can actually strengthen that hypothesis and get the following variation. By doing this, the rule becomes complete, as we will show in Section 5.2.3.

Proposition 5.2.8 (Rely/Guarantee Circular Inference Rule 4(bis))**.** *If:*

1. *a)* $E_1 \Rightarrow M_1$
   b) $E_2 \Rightarrow M_2$
2. *a)* $M_1 \Rightarrow E_2$
   b) $M_2 \Rightarrow E_1$
3. $\mathrm{SomP_i}(S) \Rightarrow (E \wedge M_1 \wedge M_2 \Rightarrow M)$
4. $S \Rightarrow \mathrm{UpToNow}(E_i) \vee \mathrm{UpToNow}(M_i)$, *for some* $i \in \{1, 2\}$
5. $E_1 \in \mathsf{ST}_{\bullet\leftarrow}$ *or* $M_1 \in \mathsf{ST}_{\bullet\leftarrow}$
6. $E_2 \in \mathsf{ST}_{\rightarrow\bullet}$ *or* $M_2 \in \mathsf{ST}_{\rightarrow\bullet}$

then $\mathrm{SomP_i}(S) \Rightarrow (E \Rightarrow M)$.

We conclude this section by pointing out that the notion of states (and right-continuous or left-continuous ones) is a meaningful one only when dealing with dense time models [GM01]. Therefore, the above rules are applicable to such time models only, and have no discrete-time counterparts.

A Circular Inference Rule Without Compositional Operator

Let us present one more circular inference rule. This rule differentiates itself from the other ones, in that it does not use a compositional operator and it does not make any assumptions on the behavior of the assumption and guarantee items. Moreover, it is substantially different than all the previous rules because it does not assert the validity of some global guarantee over a time interval that goes from the system initialization to the current time. Instead, it asserts that the system always responds in a timely manner by making true the global guarantee, provided the global assumption holds continuously for some time, and some of the local assumptions or guarantees are periodically initialized. Thus, in a sense, it describes a system with a fixed bounded response time to periodic initializations. In such systems the circularities are resolved *periodically*, rather than once for all at the beginning. Therefore, the rule is suitable to prove properties about systems composed of modules periodically responding to stimuli. In Section 6.3 we will demonstrate how to use the rule with an example of communication protocol.

According to our taxonomy, the rule is circular, non self-discharging, fully compositional, globally initialized (actually, not initialized at all), without compositional operator. Its soundness proof is rather simple, and requires no temporal induction, contrarily to the other rules we have presented.[3]

Proposition 5.2.9 (Rely/Guarantee Circular Inference Rule 5)**.** *If, for some fixed duration* $\mathrm{T_B} > 0$*:*

1. *a)* $E_1 \Rightarrow M_1$
   b) $E_2 \Rightarrow M_2$
2. *a)* $M_1 \Rightarrow E_2$
   b) $M_2 \Rightarrow E_1$
3. $E \wedge M_1 \wedge M_2 \Rightarrow M$
4. $\exists i \in \{1,2\} : \mathrm{WithinF}(E_i, \mathrm{T_B}) \vee \mathrm{WithinF}(M_i, \mathrm{T_B})$

[3] Notice that the conclusion formula $\mathrm{Lasts}(E, \mathrm{T_B}) \Rightarrow \mathrm{WithinF}(M, \mathrm{T_B})$ is a weaker version of the bounded *release* operator $\mathrm{Releases}_{<T}(P, Q)$. The operator is usually defined as $\forall 0 < t < T : \mathrm{Futr}(Q, t) \vee \mathrm{WithinF}(P, t)$. Therefore $\mathrm{Releases}_{<\mathrm{T_B}}(M, \neg E)$ implies $\mathrm{Lasts}(E, \mathrm{T_B}) \Rightarrow \mathrm{WithinF}(M, \mathrm{T_B})$, but not *vice versa*, as the latter is true whenever E holds only over intervals of length less than $\mathrm{T_B}$.

then $\text{Lasts}(E, \text{T}_\text{B}) \Rightarrow \text{WithinF}(M, \text{T}_\text{B})$.

Proof. Let t be a generic time instant at which $\text{Lasts}(E, \text{T}_\text{B})$ holds, and prove that $\text{WithinF}(M, \text{T}_\text{B})$. Notice that the same proof works for both dense and discrete time domains.

From (4), let us assume that $\text{WithinF}(E_1, \text{T}_\text{B})$ at t. This is without loss of generality: in fact, if $\text{WithinF}(M_2, \text{T}_\text{B})$, then $\text{WithinF}(E_1, \text{T}_\text{B})$ from (2b); if $\text{WithinF}(E_2, \text{T}_\text{B})$, then $\text{WithinF}(M_2, \text{T}_\text{B})$ from (1b); if $\text{WithinF}(M_1, \text{T}_\text{B})$, then $\text{WithinF}(E_2, \text{T}_\text{B})$ from (2a).

Thus, there exists a t' such that $t < t' < \text{T}_\text{B}$ and E_1 holds at t'. Therefore, $E_1 \wedge M_1 \wedge E_2 \wedge M_2$ holds at at t' by (1) and (2). Moreover, since $t < t' < \text{T}_\text{B}$ and $\text{Lasts}(E, \text{T}_\text{B})$ at t, then E also holds at t'. But then, M holds at t' by (3). Since $t < t' < \text{T}_\text{B}$, this implies that $\text{WithinF}(M, \text{T}_\text{B})$ at t. □

5.2.3. Completeness of Circular Inference Rules

This section analyzes the completeness of the circular inference rules presented above.

Theorem 5.2.10. *The rely/guarantee circular inference rule 1 of Proposition 5.2.1 is incomplete.*

Proof. Let σ, ϵ, μ be three Boolean basic time-dependent items; let us take $S = \sigma$, $E = \epsilon$, and $M = \mu$. Then, in order to show incompleteness, we provide a history for σ, ϵ, μ such that the conclusion of the inference rule holds, but no choice of E_i, M_i makes true all the premises of the rule.

The history is as follows. μ and ϵ are false everywhere, whereas σ is true only at some point s internal to the time domain (and it is false everywhere else). Notice that this history is definable in TRIO by the two formulas $\text{Alw}(\neg\mu \wedge \neg\epsilon)$ and $\text{Som}(\sigma \wedge \text{AlwP}(\neg\sigma) \wedge \text{AlwF}(\neg\sigma))$.[4]

Let us now realize that $E \twoheadrightarrow M$ is always true, since E is always false; therefore the conclusion of the inference rule holds *a fortiori*.

Let us now consider what happens at s. In order to make (4) true, let us assume that it is $\text{UpToNow}(E_1)$ at s. As usual, this is without loss of generality, since if $\text{UpToNow}(M_2)$ then $\text{UpToNow}(E_1)$ in order to

[4] For simplicity, assume a bi-infinite time domain such as $\mathbb{R}$ or $\mathbb{Z}$, so that there are no (finite) boundaries. Extensions to mono-infinite or even infinite time domains are routine.

satisfy (2b); if UpToNow(E_2) then UpToNow(M_2) in order to satisfy (1b); if UpToNow(M_1) then UpToNow(E_2) in order to satisfy (2a).

Then, UpToNow(E_1) at s implies UpToNow(M_1) at s to satisfy (1a), and therefore also UpToNow(M_2) at s to satisfy (1b, 2). Since UpToNow($M_1 \wedge M_2$) at s, then (3) requires that UpToNow(M) at s as well. But $M = \mu$ is false everywhere by assumption, so there is no choice of E_1, E_2, M_1, M_2 that is compatible with all the premises of the rule. □

Theorem 5.2.11. *The rely/guarantee circular inference rule 2 of Proposition 5.2.2 is incomplete.*

Proof. Let σ, ϵ, μ be three Boolean basic time-dependent items; let us take $S = \sigma$, $E = \epsilon$, and $M = \mu$. Then, in order to show incompleteness, we provide a history for σ, ϵ, μ such that the conclusion of the inference rule holds, but no choice of E_i, M_i makes true all the premises of the rule.

The history is as follows. μ is false everywhere, whereas σ is true only at some point s internal to the time domain (and it is false everywhere else). At s, ϵ is true on a non-empty interval on the left which is strictly contained within the time domain; that is, let us say that ϵ holds exactly on the interval $(s - \gamma, s)$ for some suitable $\gamma > 0$. Notice that the history is definable in TRIO.

Let us now realize that $\mathrm{AlwP_e}(E)$ is always false, since $(s-\gamma, s)$ is strictly contained in the time domain by hypothesis. Therefore, the conclusion of the inference rule coincides with an implication with false antecedent, and it is therefore trivially true.

Let us now consider what happens at s. In order to make (4) true, it must be either UpToNow($E_1 \wedge E_2$) or UpToNow($M_1 \wedge M_2$) at s. However, if UpToNow($E_1 \wedge E_2$) then also UpToNow($M_1 \wedge M_2$) by (1), so let us assume the latter without loss of generality.

Finally, notice that it is also UpToNow(E) at s, therefore it must be UpToNow(M) at s. But $M = \mu$ is false everywhere, thus we have a false premise. □

If we look carefully at the above incompleteness proofs, we notice that the main obstacle to achieving completeness is the impossibility of choosing suitable values for E_i, M_i *before the system is initialized*, that is when $\mathrm{SomP_i}(S)$ is false. As a consequence, by switching to locally initialized inference rules, it may be possible to achieve completeness. Nonetheless, having a locally initialized inference rule is neither a sufficient nor a necessary condition for completeness. Indeed, in the remainder we show that the

locally initialized inference rule 1(bis) of Proposition 5.2.3 is incomplete, whereas the locally initialized inference rule 2(bis) of Proposition 5.2.4 is complete. Moreover, we show that the globally initialized rule 1(ter) of Proposition 5.2.5 is also complete. Therefore, in general the relationship between the features of a compositional rule and its completeness is subtle and non straightforward.

Theorem 5.2.12. *The rely/guarantee circular inference rule 1(bis) of Proposition 5.2.3 is incomplete.*

Proof. As we did above, let σ, ϵ, μ be three Boolean basic time-dependent items; let us take $S = \sigma$, $E = \epsilon$, and $M = \mu$. In order to show incompleteness, we provide a history for σ, ϵ, μ such that the conclusion of the inference rule holds, but no choice of E_i, M_i makes true all the premises of the rule.

The history is as follows. μ and ϵ are false everywhere, whereas σ is true only at some point s internal to the time domain (and it is false everywhere else). Notice that the history is definable in TRIO.

Let us now realize that $E \twoheadrightarrow M$ is always true, since E is always false; therefore the conclusion of the inference rule holds *a fortiori*.

Let us now consider what happens at s. In order to make (4) true, let us assume that it is NowOn(E_1) at s. As usual, this is without loss of generality, since if NowOn(M_2) then NowOn(E_1) in order to satisfy (2b); if NowOn(E_2) then NowOn(M_2) in order to satisfy (1b); if NowOn(M_1) then NowOn(E_2) in order to satisfy (2a).

Then, NowOn(E_1) at s implies NowOn(M_1) at s to satisfy (1a), and therefore also NowOn(M_2) at s to satisfy (1b, 2). Since NowOn($M_1 \wedge M_2$) at s, then (3) requires that NowOn(M) at s as well. But $M = \mu$ is false everywhere by assumption, so there is no choice of E_1, E_2, M_1, M_2 that is compatible with all the premises of the rule. □

Notice that in the proofs of the following Theorem (5.2.13) we assume to deal only with non-Zeno items. We conjecture that the theorem holds also for Zeno items, but the proof would be even more involved — and not practically very interesting, as Zenoness is anyway a source of incompleteness on its own [GM01] — so we omit it for simplicity.

Theorem 5.2.13. *The rely/guarantee circular inference rule 1(ter) of Proposition 5.2.5 is complete.*

Proof. Let us assume that the conclusion $\text{SomP}_\text{i}(S) \Rightarrow (E \twoheadrightarrow M)$ holds. Then, let us define M_1 as follows.

$$M_1 \equiv \begin{cases} E \twoheadrightarrow M & \text{if } \text{SomP}_\text{i}(S) \\ \text{true} & \text{if } \text{Until}(M, S) \vee \text{Until}(\neg M \wedge \neg E, S) \text{ and } \text{AlwP}_\text{i}(\neg S) \\ \text{false} & \text{otherwise} \end{cases}$$

Equivalently, we can express the definition above with the TRIO formula:

$$\begin{aligned} &(\text{SomP}_\text{i}(S) \Rightarrow (M_1 \Leftrightarrow (E \twoheadrightarrow M))) \quad \wedge \\ &\quad (\text{AlwP}_\text{i}(\neg S) \wedge (\text{Until}(M, S) \vee \text{Until}(\neg M \wedge \neg E, S)) \Rightarrow M_1) \quad \wedge \\ &\quad\quad (\text{AlwP}_\text{i}(\neg S) \wedge \neg\text{Until}(M, S) \wedge \neg\text{Until}(\neg M \wedge \neg E, S) \Rightarrow \neg M_1) \end{aligned}$$

or, more concisely, with the equivalent formula:

$$\begin{aligned} M_1 \Leftrightarrow\ &(\text{SomP}_\text{i}(S) \wedge (E \twoheadrightarrow M)) \\ &\vee (\text{AlwP}_\text{i}(\neg S) \wedge (\text{Until}(M, S) \vee \text{Until}(\neg M \wedge \neg E, S))) \end{aligned}$$

Similarly, let us choose $M_2 = E_1 = E_2$ as follows.

$$M_2 = E_1 = E_2 \equiv \begin{cases} \text{true} & \text{if } \text{Until}(M, S) \text{ or } \text{Until}(\neg M \wedge \neg E, S) \text{ or } \text{SomP}_\text{i}(S) \\ \text{false} & \text{otherwise} \end{cases}$$

Then:

- Let us consider hypothesis (1a) at some instant t. We distinguish the following cases:
  - If $\text{SomP}_\text{e}(S)$ holds at t, then notice that the three formulas: $\text{UpToNow}(E \twoheadrightarrow M)$, $\text{NowOn}(E \twoheadrightarrow M)$, and $\text{UpToNow}(E_1)$ also hold at t. This is because $\text{SomP}_\text{e}(S)$ implies $\text{UpToNow}(\text{SomP}_\text{e}(S))$ and $\text{NowOn}(\text{SomP}_\text{e}(S))$. Therefore, (1a) is equivalent to $\text{UpToNow}(E_1) \Rightarrow \text{UpToNow}(E \twoheadrightarrow M) \wedge (E \twoheadrightarrow M) \wedge \text{NowOn}(E \twoheadrightarrow M)$, and it does hold at t.
  - If S at t (and $\neg\text{SomP}_\text{e}(S)$), then without loss of generality let us assume that $\text{UpToNow}(E_1)$ holds as well; if not, (1a) is trivially true since $\text{UpToNow}(\neg E_1)$ would follow for properties of Zeno items. But from the definition of M_1 notice that whenever $\text{UpToNow}(E_1)$ and S, then also $\text{UpToNow}(M_1)$. Moreover, after t it is $\text{SomP}_\text{e}(S)$, and thus $M_1 = (E \twoheadrightarrow M)$ holds there by assumption. Thus, at t it is also M_1 and $\text{NowOn}(M_1)$. This establishes (1a) in this case.

  - Otherwise $\mathrm{AlwP_i}(\neg S)$ at t. Again, without loss of generality we can assume that $\mathrm{UpToNow}(E_1)$. But then, whenever $\mathrm{AlwP_i}(\neg S)$, M_1 holds iff E_1 holds. So, if also $\mathrm{NowOn}(\neg S)$, (1a) holds at t. Otherwise, it must be $\mathrm{NowOn}(S)$ and $\neg S$ at t. In this case, $\mathrm{NowOn}(\mathrm{SomP_i}(S))$ holds at t, and thus $\mathrm{NowOn}(M_1)$ also holds at t (from the assumed conclusion of the inference rule and the definition of M_1). Thus finally, (1a) holds at t in this case as well.

- The reasoning for hypothesis (1b) is similar to that for hypothesis (1b), only a bit simpler since E_2 and M_2 are everywhere equal. We omit the details which are the same as in the previous step.

- Let us consider hypothesis (2a): if it is evaluated when $\mathrm{SomP_i}(S)$, then is reduces to $(E \twoheadrightarrow M) \Rightarrow \mathrm{true}$, which is true by hypothesis. If instead $\mathrm{AlwP_i}(\neg S)$, then either $\mathrm{Until}(M, S) \vee \mathrm{Until}(\neg M \wedge \neg E, S)$ or not. In the former case, (2a) reduces to an implication with true consequent; in the latter case it reduces to an implication with false antecedent. Both are tautologies.

- A similar reasoning goes for hypothesis (2b), which reduces to either $\mathrm{true} \Rightarrow \mathrm{true}$ or to $\mathrm{false} \Rightarrow \mathrm{false}$.

- Let us consider hypothesis (3) when $\mathrm{SomP_i}(S)$. Then, it can be rewritten as $E \wedge (E \twoheadrightarrow M) \twoheadrightarrow M$. Now, notice that if $E \twoheadrightarrow M$ then *a fortiori* $E \wedge (E \twoheadrightarrow M) \twoheadrightarrow M$, since the latter can be written as an implication with the same consequent but a more demanding antecedent. Since we are assuming that $E \twoheadrightarrow M$, we are done in this case.

  Otherwise, $\mathrm{AlwP_i}(\neg S)$, and let t be the instant at which we are evaluating (3). Now the analysis gets more involved. First of all, let us consider the case in which $\mathrm{AlwF_e}(\neg S)$. Thus S is always false; therefore, $\mathrm{Until}(M, S) \vee \mathrm{Until}(\neg M \wedge \neg E, S)$ is also false (since we are dealing with strong until), and thus $M_2 = E_1 = E_2 = \mathrm{false}$ everywhere. Therefore $\mathrm{UpToNow}(M_2)$ is always false, which implies that $E \wedge M_1 \wedge M_2 \twoheadrightarrow M$ is always true.

  Otherwise S is true somewhere in the future. Without loss of generality, since we are assuming non-Zeno items, let $s > t$ be the *next* time instant at which S or $\mathrm{NowOn}(S)$ holds. Now we further consider two cases:

- If Until(M, S) at t, then in particular M holds over the interval (t, s). Let us now consider the next point in the past from t at which M becomes false; let $u \leq t$ be this instant. Thus, M holds continuously over the interval (u, s) and is false before u (the value of M exactly at u is not relevant now). Notice that for all instants in (u, s), the formula UpToNow$(M) \wedge M \wedge$ NowOn(M) holds, since we are within an open interval. This implies that (3) can be rewritten as an implication with true consequent (by considering the definition of the $\twoheadrightarrow$ operator), which is therefore true throughout (u, s).

  We still have to evaluate (3) in the interval $(-\infty, u]$. Notice that in all these instants, the formula UpToNow$(M_1 \wedge M_2)$ is false; in fact, M_1 and M_2 are true at most at u and after it, but false before it since Until(M, S) no longer holds (recall that AlwP$_{\mathrm{i}}(\neg S)$ holds at $t \geq u$). Therefore, *a fortiori* UpToNow$(E \wedge M_1 \wedge M_2)$ is false at these instants. But then $E \wedge M_1 \wedge M_2 \twoheadrightarrow M$ is equivalent to an implication with false antecedent, which is trivially true.

- If instead Until$(\neg M \wedge \neg E, S)$ at t, then let us repeat the above reasoning for $\neg M \wedge \neg E$ in place of M. So, assume that M and E are false throughout (u, s) for some $u \leq t$. As a consequence, UpToNow$(M) \wedge M \wedge$ NowOn(M) is now false throughout (u, s), and so is UpToNow(E). Therefore, *a fortiori* UpToNow$(E \wedge M_1 \wedge M_2)$ is false at these instants. But then $E \wedge M_1 \wedge M_2 \twoheadrightarrow M$ is equivalent to an implication with false antecedent, which is trivially true.

  Similarly as the previous case, in the interval $(-\infty, u]$ the formula UpToNow$(M_1 \wedge M_2)$ is false. Thus, $E \wedge M_1 \wedge M_2 \twoheadrightarrow M$ is equivalent to an implication with false antecedent, which is trivially true.

- Notice that we have no other cases to consider. In particular if both Until(M, S) and Until$(\neg M \wedge \neg E, S)$ are false at t, then UpToNow$(E \wedge \neg M)$ holds at s. But this contradicts the assumption that the conclusion of the inference rule is true there.

- Finally, hypothesis (4). Let us consider any instant s at which S holds. At s, either UpToNow(M) or UpToNow$(\neg M)$, since we are dealing with non-Zeno items. If UpToNow(M), then UpToNow$(M_2 \wedge$

$E_1 \wedge M_1$), since Until(M, S) holds in a left-neighborhood of s. If UpToNow$(\neg M)$ then it must also be UpToNow$(\neg E)$, otherwise the conclusion
$\mathrm{SomP_i}(S) \Rightarrow (E \twoheadrightarrow M)$ would be false at s. Therefore, UpToNow$(M_2 \wedge E_1 \wedge M_1)$, since Until$(\neg M \wedge \neg E, S)$ holds in a left-neighborhood of s. □

Theorem 5.2.14. *The rely/guarantee circular inference rule 2(bis) of Proposition 5.2.4 is complete.*

Proof. The proof is all the same as the previous one: let us assume that the conclusion $\mathrm{SomP_e}(S) \wedge \mathrm{AlwP_e}(E) \Rightarrow (E \twoheadrightarrow M)$ holds. Then, let us define M_1 as $M_1 = E \twoheadrightarrow M$, and let $M_2 = E_1 = E_2 = \text{true}$ be all identically equal to true.

Therefore:

- Hypothesis (1a) is equivalent to $\mathrm{SomP_e}(S) \wedge \mathrm{AlwP_e}(E) \Rightarrow (E \twoheadrightarrow M)$ which is exactly the conclusion.
- Hypotheses (1b,2) are all equivalent to $\mathrm{SomP_e}(S) \wedge \mathrm{AlwP_e}(E) \Rightarrow \text{true}$, which is trivially true.
- Let us consider hypothesis (3). Without loss of generality, let us take any instant at which $\mathrm{SomP_e}(S)$ and $\mathrm{AlwP_e}(E)$ hold. Then, we have to establish that $(E \twoheadrightarrow M) \wedge \text{true} \Rightarrow M$. Clearly, this is the same as just M, since $E \twoheadrightarrow M$ holds from the conclusion. Now notice that $\mathrm{AlwP_e}(E)$ implies *a fortiori* that UpToNow(E).[5] Therefore, the conclusion allows us to assert that UpToNow$(M) \wedge M \wedge \mathrm{NowOn}(M)$, so that M is, in particular, true.
- Hypothesis (4) is trivially satisfied for $E_1 = E_2 = \text{true}$. □

The following result has been already proved in [FRMM07].

Theorem 5.2.15. *The rely/guarantee circular inference rule 3 of Proposition 5.2.6 is incomplete.*

[5] We are skipping a little technicality here: we are tacitly assuming that we are at a point internal to the time domain, otherwise it could be that $\mathrm{AlwP_e}(E)$ is trivially true, but UpToNow(E) is false since there is no non-empty interval to the left of the boundary. However, this is not a problem, as we just have to notice that $\mathrm{SomP_e}(S)$ being true implies that we are indeed in an internal point.

Proof. Let σ, ϵ, μ be three Boolean basic time-dependent items; let us take $S = \sigma$, $E = \epsilon$, and $M = \mu$. Then, in order to show incompleteness, we provide a history for σ, ϵ, μ such that the conclusion of the inference rule holds, but no choice of E_i, M_i makes true all the premises of the rule.

The history is as follows. μ and ϵ are false everywhere, whereas σ is true only at some point s internal to the time domain (and it is false everywhere else). Notice that the history is definable in TRIO.

Let us now realize that $E \gg M$ is always true, since $\mathrm{AlwP_e}(E)$ is always false; therefore the conclusion of the inference rule holds *a fortiori*.

Let us now consider what happens at s. Without adding details (as they are all similar as in the other proofs), one realizes that in order for (4) to be true, we must have $\mathrm{AlwP_e}(E_1 \wedge E_2 \wedge M_1 \wedge M_2)$ at s.

Then, (3) requires that $\mathrm{AlwP_e}(M)$ at s. But $M = \mu$ is false everywhere by assumption, so there is no choice of E_1, E_2, M_1, M_2 that is compatible with all the premises of the rule. □

Theorem 5.2.16. *The rely/guarantee circular inference rule 4 of Proposition 5.2.7 is incomplete, whereas the rule 4(bis) of Proposition 5.2.8 is complete.*

Proof. The incompleteness proof is along the usual lines. Let S be true exactly at some time s, M false everywhere, and E be true on some open interval $(s - \epsilon, s)$, and false everywhere else (in particular, this implies that $\mathrm{UpToNow}(E) \wedge \neg E$ holds at s). Notice that this implies that whenever $\mathrm{SomP_i}(S)$, then E is false, so $E \Rightarrow M$ holds; thus the conclusion of the inference rule holds everywhere. Then, by (4) and (1–2) it must be $\mathrm{UpToNow}(E_1 \wedge E_2 \wedge M_1 \wedge M_2)$ at s. Moreover, by (3) $\mathrm{UpToNow}(M)$ must also hold at s, a contradiction.

The completeness proof instead is as follows. Let us assume that the conclusion $\mathrm{SomP_i}(S) \Rightarrow (E \Rightarrow M)$ holds. Then, let us define M_1 as $M_1 = \mathrm{SomP_i}(S) \Rightarrow (E \Rightarrow M)$, and let $M_2 = E_1 = E_2 = \text{true}$ be all identically equal to true.

(1a) is the same as $\mathrm{SomP_i}(S) \Rightarrow (E \Rightarrow M)$, which holds by assumption. (1b) is trivially true, being an implication with true consequent. (2) are both trivially true, also being implications with true consequents. (3) can be rewritten as $\mathrm{SomP_i}(S) \Rightarrow (E \Rightarrow M)$, again true by assumption. (4) is satisfied everywhere by $E_1 = E_2 = \text{true}$. Finally, notice that a constant Boolean time-dependent item is both a right-continuous and a left-continuous state, so (5–6) is satisfied by E_1, E_2. □

Theorem 5.2.17. *The rely/guarantee circular inference rule 5 of Proposition 5.2.9 is complete.*

Proof. Let us assume that the conclusion $\mathrm{Lasts}(E, \mathrm{T_B}) \Rightarrow \mathrm{WithinF}(M, \mathrm{T_B})$ holds. Then, let us define $E_1 = E_2 = M_1 = M_2$ implicitly as follows.

$$\begin{aligned}(\mathrm{Lasts}(E, \mathrm{T_B}) \Rightarrow \mathrm{Lasts}(E_1 = E_2 = M_1 = M_2 = M, \mathrm{T_B})) \quad \wedge \\ (\mathrm{WithinF}(\neg E, \mathrm{T_B}) \Rightarrow \\ \mathrm{Until}(E_1 = E_2 = M_1 = M_2 = \neg E, \\ \mathrm{Lasts}(E, \mathrm{T_B}) \wedge E_1 = E_2 = M_1 = M_2 = \neg E) \\ \vee \mathrm{AlwF_e}(\mathrm{WithinF}(\neg E, \mathrm{T_B}) \wedge E_1 = E_2 = M_1 = M_2 = \neg E))\end{aligned}$$

Thus clearly (1–2) all reduce to propositional tautologies of the form $A \Rightarrow A$, since $E_1 = E_2 = M_1 = M_2$ everywhere.

Then, let us consider (4) first, and let t be any instant. At t, either $\mathrm{Lasts}(E, \mathrm{T_B})$ or $\mathrm{WithinF}(\neg E, \mathrm{T_B})$. In the former case, the conclusion lets us deduce that $\mathrm{WithinF}(M, \mathrm{T_B})$. But since also $\mathrm{Lasts}(E_1 = E_2 = M_1 = M_2 = M, \mathrm{T_B})$ at t, then $\mathrm{WithinF}(E_1 \wedge E_2 \wedge M_1 \wedge M_2, \mathrm{T_B})$, which satisfies (4). Otherwise, we have a $t < t' < t + \mathrm{T_B}$ such that $\neg E$ holds at t'. Then, it is either $\mathrm{AlwF_e}(E_1 = E_2 = M_1 = M_2 = \neg E \wedge \mathrm{WithinF}(\neg E, \mathrm{T_B}))$ at t, or not. In the former case, we have $E_1 = E_2 = M_1 = M_2$ at t', and thus (4) is satisfied. In the latter case, there exists an instant $t'' > t$ such that $E_1 = E_2 = M_1 = M_2 = \neg E$ holds until t'' *included*. If $t'' \geq t'$, then (4) is implied; otherwise $t'' < t'$, $\mathrm{Lasts}(E, \mathrm{T_B})$ holds at t'', which contradicts the fact that E is false at t'.

Next, let us discuss hypothesis (3), at a generic time instant t. If E is false at t, then the implication holds trivially.

Otherwise E is true at t. Let us distinguish two cases: (a) if there exists an interval $(p, p + \mathrm{T_B})$ such that $t \in (p, p + \mathrm{T_B})$ and E is true throughout $(p, p + \mathrm{T_B})$; (b) if this is not the case. Notice that if (b), then in particular there must exist an instant $t - \mathrm{T_B} < q < t$ such that $\mathrm{WithinF}(\neg E, \mathrm{T_B})$ holds at q.

Thus, if (a) is the case, then let us consider the instant p. At p $\mathrm{Lasts}(E, \mathrm{T_B})$ holds, therefore $\mathrm{Lasts}(E_1 = E_2 = M_1 = M_2 = M, \mathrm{T_B})$ also holds at p. Thus at $t \in (p, p + \mathrm{T_B})$: if M, then also $M_1 \wedge M_2$, thus (3) holds; if $\neg M$, then also $\neg M_1 \wedge \neg M_2$, thus (3) also holds, being an implication with false antecedent.

Otherwise, (b) is the case, and $\mathrm{WithinF}(\neg E, \mathrm{T_B})$ holds at q. Let us distinguish whether $\mathrm{Lasts}(\mathrm{WithinF}(\neg E, \mathrm{T_B}), t - q)$ holds at q or not. If it does, then surely $E_1 = E_2 = M_1 = M_2 = \neg E$ at t, and therefore (4)

reduces to $E \wedge \neg E \Rightarrow M$, which is trivially true having a contradiction in the antecedent. If it does not, then there exists some $q < q' < t$ such that $\text{Lasts}(E, \text{T}_\text{B})$ holds at q'. But this corresponds to case (a), a contradiction which concludes this branch as well. □

5.2.4. Summary of Circular Rules

Table 5.2 summarizes the various circular compositional inference rules we have presented above; an I/C letter after before the proposition number denotes if the rule is incomplete/complete.

5.3. Generalization to More Than Two Modules

This section discusses how the compositional inference rules presented in Section 5.1 and 5.2 can be generalized to handle a system with any number $N \geq 2$ of modules.

Let us point out that these generalizations are not strictly necessary to apply our compositional rules to systems with more than two modules. In fact, the rules can be applied by recursively partitioning the set of modules into two suitable sets, and apply the compositional rules by considering the two subsets as two (macro) modules. Nonetheless, the structure of a multi-modular system is often best exploited by considering the composition of all the modules at the same level. In fact, a well-designed system often has a modular partitioning where it is "natural" to associate some local properties to each of its modules. Therefore, we extend our compositional rules to handle such commonly encountered cases.

Throughout this section N is an integer greater than one, that represents the number of modules.

5.3.1. Non-Circular Inference Rules

The non-circular inference rules of Propositions 5.1.1 and 5.1.2 can be extended to handle any number of modules by introducing a way to describe a set of discharging formulas where the local assumption of each module is discharged by the local guarantee of another module, and no circularities are introduced. To this end, it is sufficient to describe a permutation of the set of modules where the local assumption of each module is discharged by the local guarantee of the module which follows in the permutation.

Rule 1 (Prop. 5.2.1 I)	**Rule 1(bis) (Prop. 5.2.3 I)**	**Rule 1(ter) (Prop. 5.2.5 C)**
$E_1 \twoheadrightarrow M_1,\ E_2 \twoheadrightarrow M_2$	$\mathrm{SomP_e}(S) \Rightarrow (E_1 \twoheadrightarrow M_1) \wedge (E_2 \twoheadrightarrow M_2)$	$E_1 \twoheadrightarrow M_1,\ E_2 \twoheadrightarrow M_2$
$M_1 \Rightarrow E_2,\ M_2 \Rightarrow E_1$	$\mathrm{SomP_e}(S) \Rightarrow (M_1 \Rightarrow E_2) \wedge (M_2 \Rightarrow E_1)$	$M_1 \Rightarrow E_2,\ M_2 \Rightarrow E_1$
$M_1 \wedge M_2 \Rightarrow M$	$\mathrm{SomP_e}(S) \Rightarrow (M_1 \wedge M_2 \Rightarrow M)$	$E \wedge M_1 \wedge M_2 \twoheadrightarrow M$
$S \Rightarrow \mathrm{UpToNow}(E_i) \vee \mathrm{UpToNow}(M_i)$	$S \Rightarrow \mathrm{NowOn}(E_i) \vee \mathrm{NowOn}(M_i)$	$S \Rightarrow \mathrm{UpToNow}(E_i) \vee \mathrm{UpToNow}(M_i)$
$\mathrm{SomP_i}(S) \Rightarrow (E \twoheadrightarrow M)$	$\mathrm{SomP_e}(S) \Rightarrow (E \twoheadrightarrow M)$	$\mathrm{SomP_i}(S) \Rightarrow (E \twoheadrightarrow M)$
Rule 2 (Prop. 5.2.2 I)	**Rule 2(bis) (Prop. 5.2.4 C)**	**Rule 3 (Prop. 5.2.6 I)**
$E_1 \twoheadrightarrow M_1,\ E_2 \twoheadrightarrow M_2$	$\mathrm{SomP_e}(S) \wedge \mathrm{AlwP_e}(E) \Rightarrow (E_1 \twoheadrightarrow M_1) \wedge (E_2 \twoheadrightarrow M_2)$	$E_1 \gg M_1,\ E_2 \gg M_2$
$E \wedge M_1 \wedge M_2 \Rightarrow E_1 \wedge E_2$	$\mathrm{SomP_e}(S) \wedge \mathrm{AlwP_e}(E) \Rightarrow (E \wedge M_1 \wedge M_2 \Rightarrow E_1 \wedge E_2)$	$M_1 \Rightarrow E_2,\ M_2 \Rightarrow E_1$
$M_1 \wedge M_2 \Rightarrow M$	$\mathrm{SomP_e}(S) \wedge \mathrm{AlwP_e}(E) \Rightarrow (M_1 \wedge M_2 \Rightarrow M)$	$M_1 \wedge M_2 \Rightarrow M$
$S \Rightarrow \mathrm{UpToNow}(E_1 \wedge E_2) \vee \mathrm{UpToNow}(M_1 \wedge M_2)$	$S \Rightarrow \mathrm{NowOn}(E_1 \wedge E_2) \vee \mathrm{NowOn}(M_1 \wedge M_2)$	$S \Rightarrow \mathrm{AlwP_e}(E_i) \vee \mathrm{AlwP_e}(M_i)$
$\mathrm{SomP_i}(S) \wedge \mathrm{AlwP_e}(E) \Rightarrow (E \twoheadrightarrow M)$	$\mathrm{SomP_e}(S) \wedge \mathrm{AlwP_e}(E) \Rightarrow (E \twoheadrightarrow M)$	$\mathrm{SomP_i}(S) \Rightarrow (E \gg M)$
Rule 4 (Prop. 5.2.7 I)	**Rule 4(bis) (Prop. 5.2.8 C)**	**Rule 5 (Prop. 5.2.9 C)**
$E_1 \Rightarrow M_1,\ E_2 \Rightarrow M_2$	$E_1 \Rightarrow M_1,\ E_2 \Rightarrow M_2$	$E_1 \Rightarrow M_1,\ E_2 \Rightarrow M_2$
$M_1 \Rightarrow E_2,\ M_2 \Rightarrow E_1$	$M_1 \Rightarrow E_2,\ M_2 \Rightarrow E_1$	$M_1 \Rightarrow E_2,\ M_2 \Rightarrow E_1$
$E \wedge M_1 \wedge M_2 \Rightarrow M$	$\mathrm{SomP_i}(S) \Rightarrow (E \wedge M_1 \wedge M_2 \Rightarrow M)$	$E \wedge M_1 \wedge M_2 \Rightarrow M$
$S \Rightarrow \mathrm{UpToNow}(E_i) \vee \mathrm{UpToNow}(M_i)$	$S \Rightarrow \mathrm{UpToNow}(E_i) \vee \mathrm{UpToNow}(M_i)$	$\mathrm{WithinF}(E_i \vee M_i, \mathrm{T_B})$
E_1 or $M_1 \in \mathsf{ST}_{\bullet\leftarrow}$, E_2 or $M_2 \in \mathsf{ST}_{\rightarrow\bullet}$	E_1 or $M_1 \in \mathsf{ST}_{\bullet\leftarrow}$, E_2 or $M_2 \in \mathsf{ST}_{\rightarrow\bullet}$	
$\mathrm{SomP_i}(S) \Rightarrow (E \Rightarrow M)$	$\mathrm{SomP_i}(S) \Rightarrow (E \Rightarrow M)$	$\mathrm{Lasts}(E, \mathrm{T_B}) \Rightarrow \mathrm{WithinF}(M, \mathrm{T_B})$

Table 5.2.: Circular compositional inference rules for two modules.

Thus, let us denote by π_N permutations of the set $\{1, \ldots, N\}$. By $\pi_N(i)$ we mean the ith element of the permutation, for $i \in \{1, \ldots, N\}$. For example if $N = 3$ and $\pi_3 = [3, 1, 2]$, then $\pi_3(1) = 3$, $\pi_3(2) = 1$ and $\pi_3(3) = 2$.

Let us first consider the extension of rule 1 of Proposition 5.1.1.

Proposition 5.3.1 (Non-Circular Rely/Guarantee Inference Rule 1N)**.** *If there exists a permutation π_N such that:*

1. *for all $i \in \{1, \ldots, N\}$: $E_i \sqsubseteq M_i$*
2. *for all $i \in \{1, \ldots, N-1\}$: $E \sqcap M_{\pi_N(i)} \sqsubseteq E_{\pi_N(i+1)}$*
3. *$E \sqcap M_1 \sqcap \cdots \sqcap M_N \sqsubseteq M$*
4. *$E \sqsubseteq M_{\pi_N(1)}$*

then $E \sqsubseteq M$

Proof sketch. Let us just provide a sketch, since the proof is easily derivable from that of Proposition 5.1.1. Assume that E holds, then $M_{\pi_N(1)}$ by (4). Then, by successively applying (1) and (2) for $i = \pi_N(1), \pi_N(2), \ldots, \pi_N(N)$, we get that all $E_{\pi_N(2)}$, $M_{\pi_N(2)}$, $\ldots$, $M_{\pi_N(N)}$ hold. Therefore, M holds by (3). □

The extension of Proposition 5.1.2 is very similar, the only difference being that we now have a fully compositional rule. For brevity, we omit the (obvious) proof.

Proposition 5.3.2 (Non-Circular Rely/Guarantee Inference Rule 2N)**.** *If there exists a permutation π_N such that:*

1. *for all $i \in \{1, \ldots, N\}$: $E_i \sqsubseteq M_i$*
2. *for all $i \in \{1, \ldots, N-1\}$: $M_{\pi_N(i)} \sqsubseteq E_{\pi_N(i+1)}$*
3. *$E \sqcap M_1 \sqcap \cdots \sqcap M_N \sqsubseteq M$*
4. *$E \sqsubseteq E_{\pi_N(1)}$*

then $E \sqsubseteq M$

Completeness. Let us remark that the completeness results about the two rules of Propositions 5.1.1 and 5.1.2 clearly carry on to Propositions 5.3.1 and 5.3.2, which therefore define *complete* inference rules. In fact, if we assume $E \sqsubseteq M$, then both Propositions 5.3.1 and 5.3.2 define a complete rule if we choose $M_{\pi_N(1)} = M$, $E_{\pi_N(1)} = E$, and all other E_i' s and M_i's equal to $\top$.

Summary. Table 5.3 summarizes the two above non-circular inference rules.

Rule 1N (Prop. 5.3.1 C)	**Rule 2N (Prop. 5.3.2 C)**
$\forall i \in \{1, \ldots, N\} : (E_i \Rightarrow M_i)$ $\forall i \in \{1, \ldots, N-1\} : (E \wedge M_{\pi_N(i)} \Rightarrow E_{\pi_N(i+1)})$ $E \wedge M_1 \wedge \cdots \wedge M_N \Rightarrow M$ $E \Rightarrow M_{\pi_N(1)}$	$\forall i \in \{1, \ldots, N\} : (E_i \Rightarrow M_i)$ $\forall i \in \{1, \ldots, N-1\} : (M_{\pi_N(i)} \Rightarrow E_{\pi_N(i+1)})$ $E \wedge M_1 \wedge \cdots \wedge M_N \Rightarrow M$ $E \Rightarrow E_{\pi_N(1)}$
$E \Rightarrow M$	$E \Rightarrow M$

Table 5.3.: Non-circular compositional inference rules for $N \geq 2$ modules.

5.3.2. Circular Inference Rules

Let us now generalize the circular compositional inference rules of Section 5.2 to the case of more than two modules. There are basically two modifications to introduce in each rule to generalize it; the underlying rationale for these two changes is common.

The first change is to the discharging formulas. Whereas in the two module case each module's assumption was usually discharged by means of the other module's guarantee, with $N > 2$ modules it is simpler and more natural to specify that the conjunction of all the modules' guarantees discharges the conjunction of all the modules' assumptions: this makes the rules self-discharging, but simpler to state as we do not have to specify complicated discharging relations. Clearly, variations are possible, but they may render the statement of the rule more involved. According to our general approach to compositionality, we recommend the formulation of such variations to be driven by the particular systems being modeled, rather than *a priori* for the sake of exploring different rules.

The second change is related to the first one. In fact, if the discharging of the assumptions depends on *all* the guarantees being true, we have to modify the initialization condition so that one can infer from S being true

that all the guarantees are also true at the same time. In practice, this can be usually implemented by requiring that the guarantee or the assumption of each module holds when S holds.

In the remainder of this section we present generalizations of the inference rules of Section 5.2 along these lines. We provide proofs for just a few of the propositions, the other proofs being routine. In the remainder, we denote the finite set $\{1, \ldots, N\}$ as $\mathcal{I}_N$.

Circular inference rules for the time progression operator.

Proposition 5.3.3 (Rely/Guarantee Circular Inference Rule 1N)**.** *If:*

1. *for all* $i \in \mathcal{I}_N$*:* $E_i \twoheadrightarrow M_i$
2. $\bigwedge_{i \in \mathcal{I}_N} M_i \Rightarrow \bigwedge_{i \in \mathcal{I}_N} E_i$
3. $\bigwedge_{i \in \mathcal{I}_N} M_i \Rightarrow M$
4. $S \Rightarrow \mathrm{UpToNow}(E_i) \vee \mathrm{UpToNow}(M_i)$*, for all* $i \in \mathcal{I}_N$

then $\mathrm{SomP_i}(S) \Rightarrow (E \twoheadrightarrow M)$.

Proof. Let t be the current instant, and let us assume that $\mathrm{SomP_i}(S)$ holds at t; thus, let $t' \leq t$ be (any) instant at which S held. We have to show that $E \twoheadrightarrow M$ holds at t.

For any $i \in \mathcal{I}_N$, either $\mathrm{UpToNow}(E_i)$ or $\mathrm{UpToNow}(M_i)$. In particular, let $I \subseteq \mathcal{I}_N$ the set of modules' indexes for which $\mathrm{UpToNow}(E_i)$ holds. Then, from (1) we can infer that also $\mathrm{UpToNow}(M_i)$ for all $i \in I$. Therefore, all in all we have that $\mathrm{UpToNow}\left(\bigwedge_{i \in \mathcal{I}_N} M_i\right)$ at t'. From (2) this implies that $\mathrm{UpToNow}\left(\bigwedge_{i \in \mathcal{I}_N} E_i\right)$ also holds at t'. Therefore, we exploit (1) to deduce that $\bigwedge_{i \in \mathcal{I}_N} M_i$ and $\mathrm{NowOn}\left(\bigwedge_{i \in \mathcal{I}_N} M_i\right)$ both hold at t' as well.

This provides the usual "temporal inductive step" as in all the other proofs for two modules. Therefore, we omit the remainder of the proof which amounts to the usual "non accumulation" argument, which is unaffected by the fact that we are now dealing with $N \geq 2$ modules. □

Since we now introduced self-discharging in the rule of Proposition 5.3.3, the only difference between Proposition 5.3.3 and the following is that we now have a non-fully compositional rule. Also notice that hypothesis (4) below could actually be relaxed to the same as in Proposition 5.3.3;

however, we leave as it is to conform with the original rule 2 of Proposition 5.2.2.

Proposition 5.3.4 (Rely/Guarantee Circular Inference Rule 2N)**.** *If:*

1. *for all* $i \in \mathcal{I}_N$*:* $E_i \twoheadrightarrow M_i$
2. $E \wedge \bigwedge_{i \in \mathcal{I}_N} M_i \Rightarrow \bigwedge_{i \in \mathcal{I}_N} E_i$
3. $\bigwedge_{i \in \mathcal{I}_N} M_i \Rightarrow M$
4. $S \Rightarrow \mathrm{UpToNow}\Big(\bigwedge_{i \in \mathcal{I}_N} E_i\Big) \vee \mathrm{UpToNow}\Big(\bigwedge_{i \in \mathcal{I}_N} M_i\Big)$

then $\mathrm{SomP_i}(S) \Rightarrow (E \twoheadrightarrow M)$.

The following rules 1N(bis), 1N(ter), and 2N(bis) are all straightforward extensions, so we introduce them without any comment or proof.

Proposition 5.3.5 (Rely/Guarantee Circular Inference Rule 1N(bis))**.** *If:*

1. *for all* $i \in \mathcal{I}_N$*:* $\mathrm{SomP_e}(S) \Rightarrow (E_i \twoheadrightarrow M_i)$
2. $\mathrm{SomP_e}(S) \Rightarrow (\bigwedge_{i \in \mathcal{I}_N} M_i \Rightarrow \bigwedge_{i \in \mathcal{I}_N} E_i)$
3. $\mathrm{SomP_e}(S) \Rightarrow (\bigwedge_{i \in \mathcal{I}_N} M_i \Rightarrow M)$
4. $S \Rightarrow \mathrm{NowOn}(E_i) \vee \mathrm{NowOn}(M_i)$*, for all* $i \in \mathcal{I}_N$

then $\mathrm{SomP_e}(S) \Rightarrow (E \twoheadrightarrow M)$.

Proposition 5.3.6 (Rely/Guarantee Circular Inference Rule 2N(bis))**.** *If:*

1. *for all* $i \in \mathcal{I}_N$*:* $\mathrm{SomP_e}(S) \wedge \mathrm{AlwP_e}(E) \Rightarrow (E_i \twoheadrightarrow M_i)$
2. $\mathrm{SomP_e}(S) \wedge \mathrm{AlwP_e}(E) \Rightarrow (E \wedge \bigwedge_{i \in \mathcal{I}_N} M_i \Rightarrow \bigwedge_{i \in \mathcal{I}_N} E_i)$
3. $\mathrm{SomP_e}(S) \wedge \mathrm{AlwP_e}(E) \Rightarrow (\bigwedge_{i \in \mathcal{I}_N} M_i \Rightarrow M)$
4. $S \Rightarrow \mathrm{NowOn}\Big(\bigwedge_{i \in \mathcal{I}_N} E_i\Big) \vee \mathrm{NowOn}\Big(\bigwedge_{i \in \mathcal{I}_N} M_i\Big)$

then $\mathrm{SomP_e}(S) \wedge \mathrm{AlwP_e}(E) \Rightarrow (E \twoheadrightarrow M)$.

Proposition 5.3.7 (Rely/Guarantee Circular Inference Rule 1N(ter))**.** *If:*

1. *for all* $i \in \mathcal{I}_N$*:* $E_i \twoheadrightarrow M_i$

2. $\bigwedge_{i\in\mathcal{I}_N} M_i \Rightarrow \bigwedge_{i\in\mathcal{I}_N} E_i$
3. $E \wedge \bigwedge_{i\in\mathcal{I}_N} M_i \twoheadrightarrow M$
4. $S \Rightarrow \mathrm{UpToNow}(E_i) \vee \mathrm{UpToNow}(M_i)$, *for all* $i \in \mathcal{I}_N$

then $\mathrm{SomP_i}(S) \Rightarrow (E \twoheadrightarrow M)$.

A stronger time progression operator. The extension of rule 3 of Proposition 5.2.6 is also straightforward, and gives a rule which is very similar to that introduced in [FRMM07].

Proposition 5.3.8 (Rely/Guarantee Circular Inference Rule 3N)**.** *If:*

1. *for all* $i \in \mathcal{I}_N$: $E_i \gg M_i$
2. $\bigwedge_{i\in\mathcal{I}_N} M_i \Rightarrow \bigwedge_{i\in\mathcal{I}_N} E_i$
3. $\bigwedge_{i\in\mathcal{I}_N} M_i \Rightarrow M$
4. $S \Rightarrow \mathrm{AlwP_e}(E_i) \vee \mathrm{AlwP_e}(M_i)$, *for all* $i \in \mathcal{I}_N$

then $\mathrm{SomP_i}(S) \Rightarrow (E \gg M)$.

Implication as a compositional operator. The circular rule 4 of Proposition 5.2.7 requires some more work to be generalized. In this case, in fact, the soundness of the rule relies on a mutual interplay between the two modules, where one is required to have an assumption (or guarantee) behaving as a left-continuous state, and the other module to have it behaving as a right-continuous state.

The idea to generalize this is thus to implement the same interplay between two sets of all modules in the system. More precisely, we can assume that the set of all modules $\mathcal{I}_N$ can be partitioned into two non empty subsets $L, R \subset \mathcal{I}_N$ such that any module $i \in L$ (resp. $i \in R$) has its assumption E_i or its guarantee M_i which is a left-continuous (resp. right-continuous) state. Then, we require that the guarantees of the modules in L can discharge all the assumptions of the modules in R, and *vice versa*. All in all, we have the following inference rule.

Proposition 5.3.9 (Rely/Guarantee Circular Inference Rule 4N)**.** *If, for two non empty subsets* $L, R \subset \mathcal{I}_N$ *that partition* $\mathcal{I}_N$*:*

1. *for all* $i \in \mathcal{I}_N$: $E_i \Rightarrow M_i$

2. a) $\bigwedge_{i\in L} M_i \Rightarrow \bigwedge_{i\in R} E_i$
   b) $\bigwedge_{i\in R} M_i \Rightarrow \bigwedge_{i\in L} E_i$
3. $E \wedge \bigwedge_{i\in\mathcal{I}_N} M_i \Rightarrow M$
4. $S \Rightarrow \mathrm{UpToNow}(E_i) \vee \mathrm{UpToNow}(M_i)$, *for all* $i \in \mathcal{I}_N$
5. $E_i \in \mathsf{ST}_{\bullet\leftarrow}$ *or* $M_i \in \mathsf{ST}_{\bullet\leftarrow}$, *for all* $i \in R$
6. $E_i \in \mathsf{ST}_{\rightarrow\bullet}$ *or* $M_i \in \mathsf{ST}_{\rightarrow\bullet}$, *for all* $i \in L$

then $\mathrm{SomP_i}(S) \Rightarrow (E \Rightarrow M)$.

Proof for dense time domains. Let t be the current instant, and let us assume that $\mathrm{SomP_i}(S)$ holds at t; thus, let $t' \leq t$ be (any) instant at which S held. We have to show that $E \twoheadrightarrow M$ holds at t.

As usual, we focus on showing one "temporal inductive step", the non accumulation argument being all similar to that of the previous proofs. More precisely, let us show that $\bigwedge_{i\in\mathcal{I}_N}(E_i \wedge M_i)$ and $\mathrm{NowOn}\Big(\bigwedge_{i\in\mathcal{I}_N}(E_i \wedge M_i)\Big)$ hold at t'.

First of all, let us show that $\bigwedge_{i\in L} \mathrm{UpToNow}(M_i)$. In fact, let i be any index $i \in L$; from (4) it is either $\mathrm{UpToNow}(E_i)$ or $\mathrm{UpToNow}(M_i)$ at t'. But, if $\mathrm{UpToNow}(E_i)$ for some i, then also $\mathrm{UpToNow}(M_i)$ from (1).

Next, from $\bigwedge_{i\in L} \mathrm{UpToNow}(M_i)$ and (2a) it follows that $\bigwedge_{i\in R} \mathrm{UpToNow}(E_i)$. Then, we consider (1) to deduce that $\bigwedge_{i\in R} \mathrm{UpToNow}(M_i)$. Thus, from (2b), also $\bigwedge_{i\in L} \mathrm{UpToNow}(E_i)$. So, all in all we have that $\bigwedge_{i\in\mathcal{I}_N} \mathrm{UpToNow}(E_i \wedge M_i)$ holds at t'.

For all $i \in L$, (6) lets us infer that also $E_i \vee M_i$ at t'. Moreover, from (1) we can actually state that $\bigwedge_{i\in L} M_i$ at t', and from (2a) it also follows that $\bigwedge_{i\in R} E_i$ at t'.

But then, for all $i \in R$, (5) — combined with (1) — lets us infer that also $\mathrm{NowOn}(M_i)$ at t'. Thus, also $\bigwedge_{i\in L} \mathrm{NowOn}(E_i)$ from (2b). (1) then implies that also $\bigwedge_{i\in L} \mathrm{NowOn}(M_i)$, and (2a) that $\bigwedge_{i\in R} \mathrm{NowOn}(E_i)$.

All in all, we have shown that $E_i \wedge M_i$ and $\mathrm{NowOn}(E_i \wedge M_i)$ both hold at t', for all $i \in \mathcal{I}_N$. The remainder of the proof is as for the other propositions. □

Proposition 5.3.10 (Rely/Guarantee Circular Inference Rule 4N(bis))**.** *If, for two non empty subsets* $L, R \subset \mathcal{I}_N$ *that partition* $\mathcal{I}_N$*:*

1. *for all* $i \in \mathcal{I}_N$*:* $E_i \Rightarrow M_i$

2. a) $\bigwedge_{i\in L} M_i \Rightarrow \bigwedge_{i\in R} E_i$
   b) $\bigwedge_{i\in R} M_i \Rightarrow \bigwedge_{i\in L} E_i$
3. $\mathrm{SomP_i}(S) \Rightarrow (E \wedge \bigwedge_{i\in\mathcal{I}_N} M_i \Rightarrow M)$
4. $S \Rightarrow \mathrm{UpToNow}(E_i) \vee \mathrm{UpToNow}(M_i)$, *for all* $i \in \mathcal{I}_N$
5. $E_i \in \mathsf{ST}_{\bullet\leftarrow}$ *or* $M_i \in \mathsf{ST}_{\bullet\leftarrow}$, *for all* $i \in R$
6. $E_i \in \mathsf{ST}_{\rightarrow\bullet}$ *or* $M_i \in \mathsf{ST}_{\rightarrow\bullet}$, *for all* $i \in L$

then $\mathrm{SomP_i}(S) \Rightarrow (E \Rightarrow M)$.

A circular inference rules without compositional operator. The extension of rule 5 of Proposition 5.2.9 is along the same lines of those we have just discussed. Moreover, as it was the case for other rules, the initialization formula could be made less restrictive at the price of complicating the discharging formulas. Indeed, in the version we present, the discharging formulas are not needed at all in the soundness proof.

Proposition 5.3.11 (Rely/Guarantee Circular Inference Rule 5N)**.** *If, for some fixed duration* $\mathrm{T_B} > 0$*:*

1. *for all* $i \in \mathcal{I}_N$*:* $E_i \Rightarrow M_i$
2. $E \wedge \bigwedge_{i\in\mathcal{I}_N} M_i \Rightarrow M$
3. $\mathrm{WithinF}\Big(\bigwedge_{i\in\mathcal{I}_N}(E_i \vee M_i), \mathrm{T_B}\Big)$

then $\mathrm{Lasts}(E, \mathrm{T_B}) \Rightarrow \mathrm{WithinF}(M, \mathrm{T_B})$.

Proof. Let t be a generic time instant at which $\mathrm{Lasts}(E, \mathrm{T_B})$ holds, and prove that $\mathrm{WithinF}(M, \mathrm{T_B})$.

From (3), let us assume that $\mathrm{WithinF}\Big(\bigwedge_{i\in\mathcal{I}_N} M_i, \mathrm{T_B}\Big)$ at t. This is without loss of generality: in fact, if $\mathrm{WithinF}(E_i, \mathrm{T_B})$ for some $i \in \mathcal{I}_N$, then $\mathrm{WithinF}(M_i, \mathrm{T_B})$ from (1). Thus, there exists a t' such that $t < t' < t + \mathrm{T_B}$ and $\bigwedge_{i\in\mathcal{I}_N} M_i$ holds at t'. Moreover, since $t < t' < t + \mathrm{T_B}$ and $\mathrm{Lasts}(E, \mathrm{T_B})$ at t, then E also holds at t'. But then, M holds at t' by (3). Since $t < t' < \mathrm{T_B}$, this implies that $\mathrm{WithinF}(M, \mathrm{T_B})$ at t. □

Notice the following: the above rule works for any variation of the Lasts and WithinF operators, provided the two variations coincide. In other words, the above rule works for any choice of $\mathrm{Lasts_{lr}}$ and $\mathrm{WithinF_{lr}}$ operator, where $l, r \in \{e, i\}$, provided l and r are the same in the two operators.

Table 5.4.: Circular compositional inference rules for $N \geq 2$ modules.

Rule 1N (Prop. 5.3.3 I)	**Rule 1N(bis) (Prop. 5.3.5 I)**	**Rule 1N(ter) (Prop. 5.3.7 C)**
$\forall i \in \mathcal{I}_N : (E_i \twoheadrightarrow M_i)$ $\bigwedge_{i\in\mathcal{I}_N} M_i \Rightarrow \bigwedge_{i\in\mathcal{I}_N} E_i$ $\bigwedge_{i\in\mathcal{I}_N} M_i \Rightarrow M$ $S \Rightarrow \forall i \in \mathcal{I}_N : (\mathrm{UpToNow}(E_i) \vee \mathrm{UpToNow}(M_i))$	$\mathrm{SomP_e}(S) \Rightarrow \forall i \in \mathcal{I}_N : (E_i \twoheadrightarrow M_i)$ $\mathrm{SomP_e}(S) \Rightarrow (\bigwedge_{i\in\mathcal{I}_N} M_i \Rightarrow \bigwedge_{i\in\mathcal{I}_N} E_i)$ $\mathrm{SomP_e}(S) \Rightarrow (\bigwedge_{i\in\mathcal{I}_N} M_i \Rightarrow M)$ $S \Rightarrow \forall i \in \mathcal{I}_N : (\mathrm{NowOn}(E_i) \vee \mathrm{NowOn}(M_i))$	$\forall i \in \mathcal{I}_N : (E_i \twoheadrightarrow M_i)$ $\bigwedge_{i\in\mathcal{I}_N} M_i \Rightarrow \bigwedge_{i\in\mathcal{I}_N} E_i$ $E \wedge \bigwedge_{i\in\mathcal{I}_N} M_i \twoheadrightarrow M$ $S \Rightarrow \forall i \in \mathcal{I}_N : (\mathrm{UpToNow}(E_i) \vee \mathrm{UpToNow}(M_i))$
$\mathrm{SomP_i}(S) \Rightarrow (E \twoheadrightarrow M)$	$\mathrm{SomP_e}(S) \Rightarrow (E \twoheadrightarrow M)$	$\mathrm{SomP_i}(S) \Rightarrow (E \twoheadrightarrow M)$
Rule 2N (Prop. 5.3.4 I)	**Rule 2N(bis) (Prop. 5.3.6 C)**	**Rule 3N (Prop. 5.3.8 I)**
$\forall i \in \mathcal{I}_N : (E_i \twoheadrightarrow M_i)$ $E \wedge \bigwedge_{i\in\mathcal{I}_N} M_i \Rightarrow \bigwedge_{i\in\mathcal{I}_N} E_i$ $\bigwedge_{i\in\mathcal{I}_N} M_i \Rightarrow M$ $S \Rightarrow \mathrm{UpToNow}\Big(\bigwedge_{i\in\mathcal{I}_N} E_i\Big) \vee \mathrm{UpToNow}\Big(\bigwedge_{i\in\mathcal{I}_N} M_i\Big)$	$\mathrm{SomP_e}(S) \wedge \mathrm{AlwP_e}(E) \Rightarrow \forall i \in \mathcal{I}_N : (E_i \twoheadrightarrow M_i)$ $\mathrm{SomP_e}(S) \wedge \mathrm{AlwP_e}(E) \Rightarrow (E \wedge \bigwedge_{i\in\mathcal{I}_N} M_i \Rightarrow \bigwedge_{i\in\mathcal{I}_N} E_i)$ $\mathrm{SomP_e}(S) \wedge \mathrm{AlwP_e}(E) \Rightarrow (\bigwedge_{i\in\mathcal{I}_N} M_i \Rightarrow M)$ $S \Rightarrow \mathrm{NowOn}\Big(\bigwedge_{i\in\mathcal{I}_N} E_i\Big) \vee \mathrm{NowOn}\Big(\bigwedge_{i\in\mathcal{I}_N} M_i\Big)$	$\forall i \in \mathcal{I}_N : (E_i \gg M_i)$ $\bigwedge_{i\in\mathcal{I}_N} M_i \Rightarrow \bigwedge_{i\in\mathcal{I}_N} E_i$ $\bigwedge_{i\in\mathcal{I}_N} M_i \Rightarrow M$ $S \Rightarrow \forall i \in \mathcal{I}_N : (\mathrm{AlwP_e}(E_i) \vee \mathrm{AlwP_e}(M_i))$
$\mathrm{SomP_i}(S) \Rightarrow (E \twoheadrightarrow M)$	$\mathrm{SomP_e}(S) \wedge \mathrm{AlwP_e}(E) \Rightarrow (E \twoheadrightarrow M)$	$\mathrm{SomP_i}(S) \Rightarrow (E \gg M)$
Rule 4N (Prop. 5.3.9 I)	**Rule 4N(bis) (Prop. 5.3.10 C)**	**Rule 5N (Prop. 5.3.11 C)**
$\forall i \in \mathcal{I}_N : (E_i \Rightarrow M_i)$ $\bigwedge_{i\in L} M_i \Rightarrow \bigwedge_{i\in R} E_i$, $\bigwedge_{i\in R} M_i \Rightarrow \bigwedge_{i\in L} E_i$ $E \wedge \bigwedge_{i\in\mathcal{I}_N} M_i \Rightarrow M$ $S \Rightarrow \forall i \in \mathcal{I}_N : (\mathrm{UpToNow}(E_i) \vee \mathrm{UpToNow}(M_i))$ $\forall i \in R : E_i$ or $M_i \in \mathsf{ST}_{\bullet\leftarrow}$, $\forall i \in L : E_i$ or $M_i \in \mathsf{ST}_{\rightarrow\bullet}$	$\forall i \in \mathcal{I}_N : (E_i \Rightarrow M_i)$ $\bigwedge_{i\in L} M_i \Rightarrow \bigwedge_{i\in R} E_i$, $\bigwedge_{i\in R} M_i \Rightarrow \bigwedge_{i\in L} E_i$ $\mathrm{SomP_i}(S) \Rightarrow (E \wedge \bigwedge_{i\in\mathcal{I}_N} M_i \Rightarrow M)$ $S \Rightarrow \forall i \in \mathcal{I}_N : (\mathrm{UpToNow}(E_i) \vee \mathrm{UpToNow}(M_i))$ $\forall i \in R : E_i$ or $M_i \in \mathsf{ST}_{\bullet\leftarrow}$, $\forall i \in L : E_i$ or $M_i \in \mathsf{ST}_{\rightarrow\bullet}$	$\forall i \in \mathcal{I}_N : (E_i \Rightarrow M_i)$ $\bigwedge_{i\in\mathcal{I}_N} M_i \Rightarrow \bigwedge_{i\in\mathcal{I}_N} E_i$ $E \wedge \bigwedge_{i\in\mathcal{I}_N} M_i \Rightarrow M$ $\mathrm{WithinF}\Big(\bigwedge_{i\in\mathcal{I}_N}(E_i \vee M_i), \mathrm{T_B}\Big)$
$\mathrm{SomP_i}(S) \Rightarrow (E \Rightarrow M)$	$\mathrm{SomP_i}(S) \Rightarrow (E \Rightarrow M)$	$\mathrm{Lasts}(E, \mathrm{T_B}) \Rightarrow \mathrm{WithinF}(M, \mathrm{T_B})$

Completeness. For the sake of brevity, we do not repeat the (in)completeness analysis performed in Section 5.2.3 on the generalized rules for $N \geq 2$ modules. Nonetheless, it is not difficult to realize that the very same results can be lifted from the two module to the N module case. In fact, firstly an incompleteness proof for a two module system implies the incompleteness of the generalization of the same rule (i.e., of the rule which reduces to the two module one if we choose $N = 2$). Secondly, the completeness proofs can be extended to the N module case by suitably instantiating all the E_i's and M_i's for $i > 2$ with the same values of E_2, M_2 in the two module case.

Summary. Table 5.4 summarizes the circular inference rules for N modules.

5.4. Compositional Methodology

This section presents a compositional methodology that exploits the previously introduced compositional inference rules in the verification of a large modular system.

Let us start by sketching the overall rationale behind the methodology. The specification of a large system is structured into classes. The fundamental behavior of the items of each class is captured by axiom formulas. The derived behavior of each class, which is to be verified, can be expressed by theorem formulas. In general, according to the rely/guarantee paradigm, we may need to relate the derived behavior of a class with certain properties of the (external) environment of that class. This can be achieved in essentially two ways. If the assumptions on the environment are temporally closed formulas (i.e., they express time-invariant properties), one may use TRIO assumption formulas to represent them. If, on the other hand, one has to express assumptions on the environment that are not invariant but temporally depend on the derived behavior they guarantee, we exploit one of the compositional operators presented for the rules of Section 5.3.[6] Then, the specification is completed by composing some classes together, as modules of the overall (global) system.

The verification burden for the global system consists essentially of two tasks. First, the assumptions that have been introduced according to the rely/guarantee paradigm have to be proved. For the sake of methodological distinction, we will generally use — whenever the object is an as-

[6]Notice that this operator can be ordinary implication.

sumption (rather than a theorem) — the verb "discharge" to mean "prove" (e.g., [Sha98]). Assumptions expressed as closed TRIO formulas are discharged by means of formulas of other classes. When doing this, it is important to avoid circularities, in order to guarantee the soundness of the whole deductive process. Conversely, assumptions expressed by means of the compositional operator are discharged using the corresponding *ad hoc* inference rule for that operator. If the rule is circular, it allows the appropriate handling of circularities. Thus, discharging these assumptions amounts to proving the truth of the premises of the inference rule, which can still be done by means of ordinary (temporal) deductive techniques. After discharging all the assumptions, the second stage of the global verification process consists in proving the theorems from axioms, assumptions, and already proved theorems. This is done first for theorems that are local (i.e., they can be proved within a single class), and then for global theorems (i.e., involving properties emerging from the combined behavior of some modules). This meets the overall verification goal.

5.4.1. Rely/Guarantee Specifications

Let us consider how to write the specification of a TRIO class C. The basic behavior of C is defined in terms of axioms over both visible and non-visible items, which rely on no assumptions, since they just state the very basic behavior of the class. Then, we state a number of *derived* properties of the class as theorems, that are going to be proved in the verification process.

Since in general a class describes an *open* component, it often happens that the truth of some theorem depends on assumptions about how the actual class environment behaves. In other words, some properties hold only if the modules that will constitute the external environment of the class satisfy some requirements. Therefore, according to the rely/guarantee paradigm, we make these assumptions explicit in writing the specification of a TRIO class, which becomes therefore a *rely/guarantee specification*.

Now, we distinguish between two different kinds of assumptions that one may need to express. The first (and simplest) case is that of assumptions which can be expressed as *temporally closed* formulas, that is formulas whose truth is invariant over the whole temporal axis. In such cases, the assumptions can be stated using the TRIO language construct of the *assumption formula*.

The other case, namely when the assumption is not expressed as a temporally closed formula, will be handled by using a compositional operator,

discussed in Sections 5.1–5.2.

Let us now introduce some notation that will help us describe these ideas more precisely. Let $\mathcal{AX}_C$, $\mathcal{AS}_C$ and $\mathcal{TH}_C$ denote the disjoint sets of axioms, assumptions and theorems of class C, respectively, and let $\mathcal{F}_C = \mathcal{AX}_C \cup \mathcal{AS}_C \cup \mathcal{TH}_C$ be the set of all formulas of C. Furthermore, for each set of formulas $\mathcal{F}_C$, we define $\mathcal{F}_C^{\vee} \subseteq \mathcal{F}_C$ as the set of *visible* formulas in $\mathcal{F}_C$, i.e., the formulas predicating over visible items only (see Section 2.2.3). Therefore, the complete specification of C is represented by the formula $\mathcal{AX}_C \cup \mathcal{AS}_C \vdash \mathcal{TH}_C$, which expresses the fact that the truth of the theorems of class C follows deductively from its axioms and assumptions.

To formalize rely/guarantee specifications where the assumption is not a temporally closed formula, one chooses a particular *compositional operator* among those introduced above, together with the corresponding inference rule. The compositional operator can simply be implication between two-dependent formulas, such as for the circular compositional inference rule 4N of Proposition 5.3.9 or for the non-circular inference rules of Section 5.1, or can have a more complicated semantics.

In the following section we show how to exploit modular rely/guarantee specifications — written along the lines we have laid in this section — in order to verify a composite systems built out of instances of these modules.

5.4.2. Modules Composition

This section completes the description of the compositional framework by formalizing the composition of TRIO classes which include both assumption formulas and rely/guarantee formulas written using the time progression operator. In a nutshell, we describe a method to prove that the assumptions of each class are met by the classes that constitute its actual environment. This is done by usual deductive techniques for temporally closed assumption formulas, while resorting to the inference rules of Tables 5.3 or 5.4 to handle rely/guarantee formulas. After discharging all the assumptions, it is possible to proceed on proving the theorems that are *global*, that is are about the combined behavior of the composed classes.

Overall Verification Process

Let us consider $n \geq 2$ TRIO classes $C_1, \ldots, C_n$. For all $i = 1, \ldots, n$, class C_i has a rely/guarantee specification expressed synthetically by the formula $\mathcal{AX}_i \cup \mathcal{AS}_i \vdash \mathcal{TH}_i$. Let C_{glob} be a class that encapsulates one instance for

each of the n classes C_i as modules of C_{glob}. The composition of the n modules is described by the logical conjunction of all the local specification formulas. Therefore TRIO modules are compositional, in that the semantics of the composition of classes is given by the logical conjunction of the semantics of the classes which are put together. In general, class C_{glob} adds its own axioms, assumptions and theorems, to those of its enclosed modules (this also allows for the recursive application of the method). Hence, C_{glob} is described by the formula $\mathcal{AX}_{\text{glob}} \cup \mathcal{AS}_{\text{glob}} \cup \bigcup_{j=1,\ldots,n} \mathcal{F}_j \vdash \mathcal{TH}_{\text{glob}}$.

Our overall verification goal can be detailed as follows.

- Verify each local (i.e., within each class C_i) specification.
- Discharge the local assumptions, that is prove that the assumptions of each class are satisfied by its actual environment (i.e., the other classes in the system).
- Verify the global specification, that is prove the global theorems from the local visible formulas, global axioms and assumptions.

It is simple to realize that the basic problem involved in the discharging of assumptions is that such reasoning may involve a *circularity* between assumptions and other formulas of the modules. As circularity prevents the verification process from being sound, these invalid deductions must be ruled out.

To this end, we introduce two "circularity-breaking" mechanisms, depending on what kind of assumptions we are considering. For assumptions written as temporally closed TRIO formulas, we require an *explicit* breaking of circularity: the verifier has to organize the discharging of these assumptions in such a way that circularity is simply avoided. Conversely, for assumptions appearing in rely/guarantee specifications using a compositional operator, we can exploit the corresponding inference rule for that operator. If that is a circular inference rule, it performs an *implicit* circularity breaking, thus ensuring that circularities arising in the hypotheses of the inference rule do not compromise the soundness of the final deduction. Otherwise, if it is a non-circular rule, the absence of circularity is explicitly required in the premises of the rules, so applying it also guarantees a sound deduction. All this must be done while respecting the requirements on the visibility of formulas.

Verification of the Composite Specification

In general, one can choose to any of the compositional operators we have introduced above to write local rely/guarantee specifications. Let us denote by $\succ$ the chosen operator. Then, each of the n component classes may have one or more rely/guarantee formulas of the form $E \succ M$ among its theorems. For each class C_j, $j = 1, \ldots, n$, let $\mathcal{TH}_j^{\mathrm{rg}} \subseteq \mathcal{TH}_j$ be the set of theorems we consider in the form $E \succ M$ of class C_j, i.e., for each theorem formula $F \in \mathcal{TH}_j$, $F \in \mathcal{TH}_j^{\mathrm{rg}}$ if F can be written as $F \equiv E \succ M$ for some formulas E and M. For simplicity, we do not address explicitly the case of specifications where different compositional operators are used for different formulas. However, these cases can be reduced to successive applications of our framework, one for each compositional operator, so our omission is without loss of generality.

Notice that, in principle, it may be that *any* formula can be written using the $\succ$ operator. For instance, if $\succ$ is the time progression operator $\twoheadrightarrow$ introduced in Section 5.2, any temporally closed formula G is equivalent to $\mathrm{Alw}(\mathrm{true} \twoheadrightarrow G)$, so the choice of which formulas to consider in $\mathcal{TH}_j^{\mathrm{rg}}$ is partially up to the specifier.[7]

Let us define m to be the number of rely/guarantee formulas over all classes: $m = \sum_{j=1,\ldots,n} |\mathcal{TH}_j^{\mathrm{rg}}|$. Moreover, $\mathcal{TH}_j^{\mathrm{nrg}}$ is defined as the complement set $\mathcal{TH}_j \setminus \mathcal{TH}_j^{\mathrm{rg}}$ for all $j = 1, \ldots, n$. The composite class C_{glob} also has its own rely/guarantee formula $E_{\mathrm{glob}} \succ M_{\mathrm{glob}}$ among its theorems $\mathcal{TH}_{\mathrm{glob}}$. It may be that the compositional operator used for the global rely/guarantee formula is different than that used for the local rely/guarantee formulas. Again, this is only a matter of notation in expressing the following verification method, but it can be done along the very same lines.

Next, let us define what is a *dependency* between two formulas. Let us consider a formal proof π: it consists of a finite sequence of formulas, together with their *justifications* (see, for example, [Men97]). We say that a formula χ *directly depends upon* another formula ϕ in the proof π, and write $\phi \rightsquigarrow_\pi \chi$, if and only if ϕ appears before χ in the proof and χ is the result of the application of an inference rule which uses ϕ. The transitive closure $\overset{+}{\rightsquigarrow}$ ("depends upon") of the $\rightsquigarrow$ relation is defined as usual.

The notion of dependency can be extended to a set of proofs Π: for any two formulas ϕ, χ we say that $\phi \overset{+}{\rightsquigarrow}_\Pi \chi$ if and only if there exists a proof

[7] Moreover, for the sake of simplicity, we do not address explicitly more convoluted cases where rely/guarantee specifications in the form $E \succ M$ appear as subformulas of larger formulas.

$\pi \in \Pi$ such that $\phi \mathrel{\not\rightarrow}_{\pi} \chi$. Note that the we will require the $\mathrel{\not\rightarrow}_{\Pi}$ relation to be irreflexive for the set Π of formal proofs constituting the verification process; this requirement will be expressed *a posteriori*.

Finally, let us illustrate the steps along which the verification of the composite specification can proceed.

1. Verify each local specification, that is prove that for all $k = 1, \ldots, n$:

$$\mathcal{AX}_k \cup \mathcal{AS}_k \vdash \mathcal{TH}_k$$

   From our perspective, this step is considered to be atomic, but obviously the compositional approach can be applied recursively to each module.

2. Show that the local assumptions can be discharged by means of global formulas, visible formulas of other classes, and local axioms and theorems.[8] Formally, this corresponds to proving that for all $k = 1, \ldots, n$:

$$\mathcal{AX}_k \cup \mathcal{TH}_k \cup \mathcal{F}_{\text{glob}} \cup \bigcup_{\substack{j=1,\ldots,n \\ j \neq k}} \mathcal{F}_j^{\vee} \vdash \mathcal{AS}_k$$

3. Prove that the global non-rely/guarantee theorems (i.e., not involving the $\succ$ operator) follow from the local visible formulas and from the global axioms, assumptions and other (i.e., rely/guarantee) theorems. In formulas, this means proving:

$$\mathcal{AX}_{\text{glob}} \cup \mathcal{AS}_{\text{glob}} \cup \mathcal{TH}_{\text{glob}}^{\text{rg}} \cup \bigcup_{j=1,\ldots,n} \mathcal{F}_j^{\vee} \vdash \mathcal{TH}_{\text{glob}}^{\text{nrg}}$$

4. Show that the initialization condition (see the generic rule of Table 3.1) can be proved. In order to prove the initialization condition, we allow one to use global formulas and local visible formulas. If we denote by `init` the initialization condition, this corresponds to proving that:

$$\mathcal{F}_{\text{glob}} \cup \bigcup_{i=1,\ldots,n} \mathcal{F}_i^{\vee} \vdash \texttt{init}$$

[8] Of course, if an assumption can be proved just from local axioms and theorems it should be a theorem, but it might happen that local formulas contribute, together with external ones, to the proof of an assumption.

5. Show that each local rely/guarantee formula has assumptions that can be discharged by means of global and local formulas, or by the global assumption, or by means of guarantees of other classes. This corresponds to proving the assumption discharging formula of Table 3.1. Formally, prove that for all $k = 1, \ldots, m$: for all $j = 1, \ldots, n$: if $(E_k \succ M_k) \in \mathcal{TH}_j^{\mathrm{rg}}$ then:

$$\mathcal{F}_{\mathrm{glob}} \cup \mathcal{F}_j \cup \bigcup_{i=1,\ldots,n} \mathcal{F}_i^{\vee} \vdash I_k^{\mathrm{dsc}} \Rightarrow \left(E_{\mathrm{glob}} \wedge \bigwedge_{i=1,\ldots,m} M_i \Rightarrow E_k \right)$$

6. Show that the global guarantee follows from the local guarantees of all modules, from the global assumption, and from global formulas and local visible formulas of any class. This corresponds to the global implementation formula of Table 3.1, i.e., prove that:

$$\mathcal{F}_{\mathrm{glob}} \cup \bigcup_{j=1,\ldots,n} \mathcal{F}_j^{\vee} \vdash I^{\mathrm{imp}} \Rightarrow \left(E \wedge \bigwedge_{j=1,\ldots,m} M_j \Rightarrow M_{\mathrm{glob}} \right)$$

7. Be sure that in all the above proofs (i.e., steps 1 to 6) there are no circular dependencies among any two closed formulas. This is the explicit circularity-breaking requirement: formally, it corresponds to checking that in the set Π of all the above proofs, for all formulas $\phi \in \bigcup_{k=1,\ldots,n}(\mathcal{AS}_k \cup \mathcal{TH}_k) \cup \mathcal{TH}_{\mathrm{glob}}$:

$$\neg(\phi \mathrel{\not\rightsquigarrow_{\Pi}} \phi)$$

   i.e., the relation $\not\rightsquigarrow_{\Pi}$ is *irreflexive*.

From the application of the above steps, thanks to the inference rule for the chosen compositional operator and the absence of circularities, the global verification of the composite specification is soundly completed.

In fact, steps 1–3 achieve the verification of the theorems in $\mathcal{TH}_{\mathrm{glob}}^{\mathrm{nrg}}$: since $\not\rightsquigarrow_{\Pi}$ is irreflexive, it induces a partial ordering on closed formulas. Thus, any global ordering of formulas compatible with the partial ordering defines a chain of proofs which all are sound, since they rely on ordinary (sound) inference rules and have no circular dependencies.

Then, steps 4–6 apply the inference rule for the chosen compositional operator (presented above in the general form of Table 3.1) by discharging

its hypotheses by means of other formulas; since we introduced sound inference rule for the various compositional operators, the theorem in $\mathcal{TH}^{\mathrm{rg}}_{\mathrm{glob}}$ is soundly proved as well, completing the verification of the global class.

6. Illustrative Examples of Compositional Proofs

This chapter demonstrates the application of the general "lightweight" approach to compositionality presented in the previous sections. The compositional framework is applied to three examples.

The first is a very simple two-module system, whose purpose is to introduce the reader to the compositional techniques as smoothly as possible, through a terse example. It is discussed in Section 6.1.

The second is a real-time version of the dining philosophers problem; the basics of the problem have already been introduced in Section 2.3: the following Section 6.2 completes the example introducing compositional techniques and methods.

The third illustrative example consists in a real-time formalization of (a part of) a peer-to-peer communication protocol. It is presented in Section 6.3.

The purpose of these examples is mainly that of being illustrations of how the techniques and methods, introduced in the previous chapters, can be applied. Obviously, they are not enough to assess the feasibility of those methods on practical industrial-strength systems. In fact, in order to allow practitioners to use them proficiently, other, formidable, practical problems should be tackled, such as how to decompose a system into modules, how to specify formally a large real-time system, and to what extent the approach can be automated. While leaving these important problems to future work, let us focus on providing some illustrative insights through the following examples.

6.1. A Simple Example

Let us consider a very simple module that inputs a Boolean signal and outputs another Boolean signal which "echoes" the input. The example is similar to that given in [AL95], and it is also presented, in a different form, in [Fur03b].

The class echoer. Formally, we introduce a class **echoer** to model the module. The class encapsulates two time-dependent predicates in and out that hold, respectively, whenever the input and output signals are "high". Then, we describe the behavior of the class through axioms; in other words we formalize the behavior of the Boolean signals as described above. Let us assume, for instance, that the timed behavior of the module is the following: out reacts to the input in becoming true by holding for some positive time span of length λ, where λ is a positive fixed constant. Note that we are considering this behavior as given (for instance by those who built, or will build, a realization of the module); therefore it is a primitive description that we capture through axioms. Using another kind of formula would be inappropriate in this case, according to this scenario. Then, the axiom that formalizes the given behavior is the following:

Axiom 11 (**echoer.echo**). $\mathsf{in} \Rightarrow \mathrm{Lasts}_{\mathrm{ie}}(\mathsf{out}, \lambda)$

Notice that, in our case, we have simply $\mathcal{AX}_{\mathrm{echoer}} = \{\mathbf{echo}\}$.

A system of two echoers. Now, let us build a composite system made of two modules of class **echoer**. The input and output signals are connected so that the out item of the first module is connected (directly, that is without delays) to the in item of the second module, and conversely the out item of the second module is connected to the in item of the first module. The resulting system is pictured in Figure 6.1.

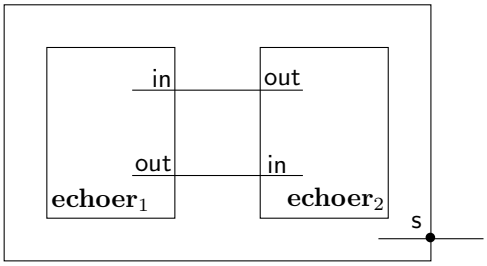

Figure 6.1.: A system of two **echoer**s.

As it can be seen in the picture, besides the items of the two modules, the global system also has a start event s. It holds whenever the system is started by "forcing" a high output signal of length λ in module 1. Formally, it is characterized by the following global axiom.

Axiom 12 (**start**). $\mathsf{s} \Rightarrow \mathrm{Lasted}_{\mathrm{ie}}(\mathbf{echoer}_1.\mathsf{out}, \lambda)$

This completes the set of global axioms, so that $\mathcal{AX}_{\mathrm{glob}} = \{\mathbf{start}\}$.

We complete the description by introducing a requirement of the global system, that we would like to prove from its components. It states that, if the system is "started" (by setting s to true), then from that moment on both output signals out stay true indefinitely in the future. Since this is a putative property, and it is the result of the global functioning of the system, we introduce it as a global theorem.

Theorem 13 (**req**). $\mathsf{s} \Rightarrow \mathrm{AlwF}(\mathbf{echoer}_1.\mathsf{out} \wedge \mathbf{echoer}_2.\mathsf{out})$

In the remainder of this section, we are going to sketch the proof of this requirement through our compositional method.

Rely/guarantee specifications. We choose the inference rule of Proposition 5.2.1 to carry out the verification. More precisely, we set the global assumption and guarantee to $E = \mathrm{true}$, and $M = \mathbf{echoer}_1.\mathsf{out} \wedge \mathbf{echoer}_2$, and the predicate S to be the event s. The first choice is reasonable as the global system is closed, and thus we need no global assumption; the second choice is instead suitable for our needs, as it is simple to realize that, from the conclusion of the inference rule $\mathrm{SomP_i}(\mathsf{s}) \Rightarrow (\mathrm{true} \twoheadrightarrow \mathbf{echoer}_1.\mathsf{out} \wedge \mathbf{echoer}_2.\mathsf{out})$, it is straightforward to infer the truth of theorem **req**.

Then, we have to set-up local rely/guarantee specifications according to Proposition 5.2.1. In other words, we have to write local specifications in terms of the $\twoheadrightarrow$ operator. Looking at the local axioms, we notice that the basic behavior of the modules entails that out reacts to in becoming true by staying true "for some time". Therefore, a derived behavior may be indeed captured by the $\twoheadrightarrow$ operator, as follows. Let us pick $E_i = \mathbf{echoer}_i.\mathsf{in}$ and $M_i = \mathbf{echoer}_i.\mathsf{out}$, for $i = 1, 2$. Then, we can convince ourselves that the following theorem is indeed provable within the **echoer** class.

Theorem 14 (**echoer.rg**). $\mathsf{in} \twoheadrightarrow \mathsf{out}$

This completes our local specifications; notice that this means that $\mathcal{TH}_{\mathrm{echoer}} = \{\mathbf{rg}\}$ and $\mathcal{AS}_{\mathrm{echoer}} = \emptyset$.

Finally, we set $\mathcal{TH}_{\mathrm{glob}} = \mathcal{TH}^{\mathrm{nrg}}_{\mathrm{glob}} \cup \mathcal{TH}^{\mathrm{rg}}_{\mathrm{glob}} = \{\mathbf{req}\} \cup \{\mathbf{rg\_glob}\}$, where theorem **rg_glob** coincides with the conclusion of the chosen inference rule, and it is therefore as follows.

Theorem 15 (**rg_glob**). $\mathrm{SomP_i}(\mathsf{s}) \Rightarrow (\mathrm{true} \twoheadrightarrow \mathbf{echoer}_1.\mathsf{out} \wedge \mathbf{echoer}_2.\mathsf{out})$

Verification. Finally, we verify the system through the steps of Section 5.4.

1. Step 1 corresponds to proving that **rg** is indeed a theorem of class **echoer**, that is it follows from axiom **echo**. We do not discuss this simple proof, focusing on the overall picture.

2. Step 2 is empty, since we have no local assumption formulas to prove.

3. Step 3 consists in proving that **req** follows from theorem **rg_glob** and axiom **start**. This is also ordinary theorem proving, and we do not discuss it.

4. Step 4 requires to prove the initialization condition for the chosen inference rule, that is:

$$\mathsf{s} \Rightarrow \mathrm{UpToNow}(\mathbf{echoer}_i.\mathsf{in}) \vee \mathrm{UpToNow}(\mathbf{echoer}_i.\mathsf{out})$$

   for some $i = 1, 2$. Now, global axiom **start** postulates that $\mathsf{s} \Rightarrow \mathrm{Lasted}_{\mathrm{ie}}(\mathbf{echoer}_1.\mathsf{out}, \lambda)$, which clearly subsumes the required step (for $i = 1$).

5. Step 5 consists in discharging the assumptions of each module. In our case, notice that $E_1 = M_2$ and $E_2 = M_1$, due to the connections in Figure 6.1. Therefore, proving that $\mathbf{echoer}_1.\mathsf{out} \Rightarrow \mathbf{echoer}_2.\mathsf{in}$ and $\mathbf{echoer}_2.\mathsf{out} \Rightarrow \mathbf{echoer}_1.\mathsf{in}$ is immediate.

6. Step 6 is trivial, since in our case $M_1 \wedge M_2 = M_{\mathrm{glob}}$ by definition.

7. Finally, the check that no circularities have been introduced in the above proofs is also simple, as there are no circular dependencies between axioms or theorems in the system.

This completes our compositional verification process, and proves that the requirement **req** is indeed satisfied by the global composite system. The following sections introduce more significant examples that are more complex.

6.2. Real-Time Dining Philosophers

Section 2.3 introduced the dining philosophers example by providing the axioms of the **philosopher** class, formalizing its basic postulated behavior.

Here, we complete the example by introducing local theorems and assumptions (Section 6.2.1), by stating local rely/guarantee behaviors expressed using a compositional operator (Section 6.2.2), by proving the theorems that are local to the the **philosopher** class (Section 6.2.3), by providing a global verification goal (Section 6.2.4), and (Section 6.2.5) by providing the complete verification of the stated goals, according to the guidelines introduced in Section 5.4. Finally, Section 6.2.6 sketches an informal comparison between the compositional proof of the example and another, non-compositional, proof of the same example.

In this specific example, we are going to use the $\gg$ compositional operator introduced in Section 5.2.2, and its corresponding circular inference rule 3N of Proposition 5.3.8 (we consider the N-module generalization).

All details of the proofs have been checked with the encoding of the TRIO language in the PVS proof checker [ORS92] (see [GM01,Fur03b] for some details of this encoding), even if we present them succinctly and in human-readable form. The interested reader can find the full PVS encoding of formulas and proofs in [FRMM05b].

6.2.1. Local Assumptions and Theorems

A derived property of the **philosopher** class is that there is always a time interval in which both forks are available to the philosopher. This is the first step in ensuring that the philosopher will eventually be able to acquire both forks and eat. In particular, the axioms allow us to prove this availability property for a time interval which occurs no later than $\mathrm{T_t} + 2\mathrm{T_e}$ time units from the current instant. This bound is sufficient to prove all of the following derived properties. The above property is expressed by the following TRIO theorem formula.

Theorem 16 (philosopher.fork_availability).
$\mathrm{WithinF_{ei}}(\mathrm{UpToNow}(\mathsf{available}(\mathrm{l}) \wedge \mathsf{available}(\mathrm{r})), \mathrm{T_t} + 2\mathrm{T_e})$

Clearly, the validity of this theorem cannot be guaranteed regardless of the behavior of the environment of this class. Therefore, we introduce three assumption formulas that suffice to deduce theorem **fork_availability**. Note that the choice of what formulas should be axioms and what assumptions is a methodological one, and it is up to the user. In this case, we realize that theorem **fork_availability** can be proved if we consider the behavior of the *other* classes (in the composite system); therefore, we choose to abstract the essential behavior of the other philosophers, needed

to guarantee this theorem, in the following assumptions (rather than axioms). This is methodologically advisable, as it permits us to decouple the facts needed to prove the theorem from the other information about the behaviors of the neighbor classes. Thus, first of all, we assume that, at any given time, each fork becomes available within $\mathrm{T_t} + \mathrm{T_e}$ time units or is already available and remains so for a sufficiently long (i.e., $\geq \mathrm{T_t}$) amount of time.

Assumption 17 (**philosopher.availability**).
$(\exists t \geq \mathrm{T_t} : \mathrm{Lasts}(\mathsf{available}(s), t)) \vee$
$\mathrm{WithinF_{ei}}(\mathrm{Becomes}(\mathsf{available}(s)), \mathrm{T_t} + \mathrm{T_e})$

Second, we assume that each fork is available, for a non-empty time interval, within $\mathrm{T_e}$ time units. This is basically like assuming that the adjacent philosophers eat for no longer than $\mathrm{T_e}$ time units.

Assumption 18 (**philosopher.availability_2**).
$\mathrm{WithinF}(\mathrm{UpToNow}(\mathsf{available}(s)), \mathrm{T_e})$

Finally, when a fork becomes available, we assume it to stay so for (at least) $\mathrm{T_t}$ time units; this corresponds to assuming that the thinking time of the neighbor philosophers is not shorter than $\mathrm{T_t}$.

Assumption 19 (**philosopher.lasting_availability**).
$\mathrm{Becomes}(\mathsf{available}(s)) \Rightarrow \mathrm{Lasts}(\mathsf{available}(s), \mathrm{T_t})$

Notice that each of these formulas expresses a temporally-closed property (for instance, considering assumption **availability_2**, at *any given time* each fork is available within $\mathrm{T_e}$ time units).

Finally, let us introduce another theorem of the **philosopher** class which will be used in its verification. The truth of this theorem is a direct consequence of the axioms of the class, requiring no assumptions. It states that any philosopher is always in one of two situations: either he/she is hungry (i.e., $\mathrm{T_t}$ time units without eating have passed), or no more than $\mathrm{T_t} + \mathrm{T_e}$ time units have elapsed since the last time he/she ate or he/she began eating.

Theorem 20 (**philosopher.always_eating_or_not**). $\mathsf{hungry} \vee$
$\mathrm{WithinP_{ii}}((\exists t > \mathrm{t_e} : \mathrm{Lasted}(\mathsf{eating}, t)) \vee \mathrm{Becomes}(\mathsf{eating}), \mathrm{T_t} + \mathrm{T_e})$

6.2.2. Rely/Guarantee Specification

Formulas **availability**, **availability_2** and **lasting_availability** of the previous section express the assumptions that each philosopher makes about the behavior of his/her neighbors. In turn, the philosopher must guarantee to them that he/she will not be unfair and will periodically release the forks. This requirement is expressed by the two theorems **taking_turns** and **taking_turns_2**, that are analogues to the assumptions **availability** and **availability_2**, while assumption **lasting_availability** corresponds to axiom **thinking_duration** (see Section 2.3.2).

Theorem 21 (**philosopher.taking_turns**).
$(\exists t \geq \mathrm{T_t} : \mathrm{Lasts}(\neg \mathsf{holding}(s)\,, t)) \vee$
$\mathrm{WithinF_{ei}}(\mathrm{Becomes}(\neg \mathsf{holding}(s))\,, \mathrm{T_t} + \mathrm{T_e})$

Theorem 22 (**philosopher.taking_turns_2**).
$\mathrm{WithinF}(\mathrm{UpToNow}(\neg \mathsf{holding}(s))\,, \mathrm{T_e})$

Up to this point, only assumptions expressed as temporally closed formulas (e.g., assumption **availability**) have been considered. Now, we have to express a new fundamental local rely/guarantee property of each philosopher: it is a non-starvation property requiring that, under the assumption of a regular availability of the forks, we can guarantee that, after the system starts, the philosopher eats regularly.

In such a case, it is convenient to express the assumption as a time-dependent formula, and link it to the guarantee using the $\gg$ operator: we relate the availability of the forks in the past to the occurrence of the eating sessions in the immediate future. Hence, let us take:

$$E_k = \mathrm{WithinF_{ei}}(\mathrm{UpToNow}(\mathsf{available}(\mathrm{l}) \wedge \mathsf{available}(\mathrm{r}))\,, \mathrm{T_t} + 2\mathrm{T_e})$$

and:

$$M_k = \mathrm{SomP_i}(\mathsf{start}) \Rightarrow (\exists t > \mathrm{t_e} : \mathrm{Within_{ii}}(\mathrm{Lasts}(\mathsf{eating}, t)\,, \mathrm{T_t} + 2\mathrm{T_e}))$$

Thus, the following theorem expresses the local non-starvation property in rely/guarantee form: as long as both forks are available within a bounded time interval (i.e., $\mathrm{T_t} + 2\mathrm{T_e}$), the philosopher is able to eat for a sufficiently long (i.e., $> \mathrm{t_e}$) time, within the same bound, provided the system has started.

Theorem 23 (philosopher.regular_eating_rg).
$\mathrm{WithinF_{ei}}(\mathrm{UpToNow}(\mathsf{available(l)} \wedge \mathsf{available(r)}), \mathrm{T_t} + 2\mathrm{T_e})$
$\gg \quad (\mathrm{SomP_i}(\mathsf{start}) \Rightarrow (\exists t > \mathrm{t_e} : \mathrm{Within_{ii}}(\mathrm{Lasts}(\mathsf{eating}, t), \mathrm{T_t} + 2\mathrm{T_e})))$

This theorem completes the specification of the **philosopher** class.

6.2.3. Local Verification

We can finally verify the whole specification, according to the method described in Section 5.4. Let us start from step 1, which prescribes to prove each local specification, i.e., each local theorem.

Besides the lastly introduced theorem **regular_eating_rg**, we have a total of four theorems in class **philosopher**. Theorems **taking_turns[_2]** and **always_eating_or_not** are proved directly from the axioms of the class, while theorem **fork_availability** relies on some of the assumptions of the class.

Moreover, in all proofs we assume that the thinking time of each philosopher is larger than twice the eating time: $\mathrm{T_t} > 2\mathrm{T_e}$.[1] This condition allows one to avoid the race conditions, and is referred to as assumption **TCCs** (for "Type Correctness Constraints", *à la* PVS).

Similarly to what was discussed when presenting Theorem **fork_availability**, one might try to determine which are the "weakest" inequalities that have to be assumed for the following proof to hold. A detailed discussion of these issues is out of the scope of the present paper, and we leave it to future work.

In summary, the whole *proof dependencies* for the **philosopher** class are displayed in Figure 6.2: whenever we use a formula α in deducing the validity of another formula β, there is a proof dependency between the two formulas, graphically represented by an arrow going from α to β. For simplicity, we did not represent the dependencies from axioms **thinking_def** and **hungry** in Figure 6.2, as they just define elementary equivalences.

For the sake of brevity, we do not show the proof of any of these theorems, except for **regular_eating_rg** which is proved below, and whose overall case structure is given in Figure 6.3. However, Appendix A reports the proofs of the other theorems in human-readable form, while PVS proofs are available in [FRMM05c].

[1] After all, they are philosophers, not gourmands! (Unless they are Epicureans, one may argue...).

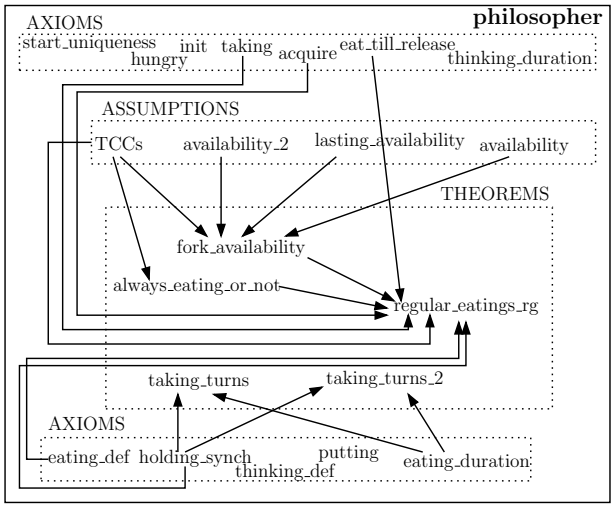

Figure 6.2.: Proof dependencies in class **philosopher**

Proof of ***philosopher.regular_eating_rg****.* By exploiting the definition of the $\gg$ operator, we assume $\mathrm{AlwP_e}(\mathrm{WithinF_{ei}}(\mathrm{UpToNow}(\mathsf{available}(\mathrm{l}) \wedge \mathsf{available}(\mathrm{r})), \mathrm{T_t}+2\mathrm{T_e}))$ as hypothesis and set our goal to proving $\mathrm{AlwP_i}(F)$ and $\mathrm{NowOn}(F)$ separately, where F is:

$$F \equiv (\exists t > \mathrm{t_e} : \mathrm{Within_{ii}}(\mathrm{Lasts}(\mathsf{eating}, t)\,, \mathrm{T_t} + 2\mathrm{T_e})) \vee \mathrm{AlwP_i}(\neg\mathsf{start})$$

an equivalent statement of the implication. First of all, we notice that, because of theorem **fork_availability (16)**, we can actually strengthen our current hypothesis to be $\mathrm{AlwP_i}(\mathrm{WithinF_{ei}}(\mathrm{UpToNow}(\mathsf{available}(\mathrm{l}) \wedge \mathsf{available}(\mathrm{r})), \mathrm{T_t} + 2\mathrm{T_e}))$, that is including the current instant. Now, let us first prove $\mathrm{AlwP_i}(F)$ from the hypothesis, the axioms and the other (already proved) theorems of the class. Let t be the generic time instant at which the hypothesis holds. We have to prove that F holds for all time instants less than or equal to t, so let $u \leq t$ a generic time instant before t. Let us consider theorem **always_eating_or_not (20)** at time u and perform a case analysis. The proof is split into two branches whether $\mathsf{hungry} \equiv \mathrm{Lasted}(\neg\mathsf{eating}, \mathrm{T_t})$ or $\mathrm{WithinP_{ii}}(\mathrm{Becomes}(\mathsf{eating}) \vee \ldots, \mathrm{T_t} + \mathrm{T_e})$ holds at u.

The first branch considers the hypothesis instantiated at time u, that is $\mathrm{WithinF_{ei}}(\mathrm{UpToNow}(\mathsf{available}(\mathrm{l}) \wedge \mathsf{available}(\mathrm{r}))\,, \mathrm{T_t} + 2\mathrm{T_e})$ at u. Therefore, let us make explicit the existentially quantified time variable of the

WithinF operator and name it f. Notice that $0 < f \leq \mathrm{T_t} + 2\mathrm{T_e}$ and we can write that UpToNow(available(l) $\wedge$ available(r)) at time $u + f$. Now, the proof is further split into two branches whether or not the state eating never becomes true for all time instants since u to $u + f$. In the first case $\mathrm{Lasts_{ie}}(\neg\mathrm{Becomes}(\mathsf{eating}), f)$ at u. Therefore, we can deduce from basic properties of state items that eating is always false from u to $u + f$. eating was also false for $\mathrm{T_t}$ time units in the past at u in this branch of the proof. Therefore, a short time before $u + f$ the philosopher is hungry, and the forks are available in the immediate past and in the immediate future. Let $u+f-\epsilon$ be this time instant (where $\epsilon > 0$ is sufficiently small) and consider axiom **acquire (3)** at this time. We immediately conclude that both forks are taken at $u+f-\epsilon$ and therefore Becomes(holding(l) $\wedge$ holding(r)) is true at $u+f-\epsilon$. Furthermore, we can consider axiom **eat_till_release (4)** at time $u+f-\epsilon$ and deduce that there exists $r > \mathrm{t_e}$ such that Lasts(eating, r) holds at $u + f - \epsilon$. Since $|f - \epsilon| \leq \mathrm{T_t} + 2\mathrm{T_e}$ this branch of the proof is concluded.

The other branch of the proof considers the case in which there is a time instant g between u and $u + f$ where Becomes(eating) is true. Since eating becomes true at g, we can apply axiom **eat_till_release (4)** and deduce that there exists $r > \mathrm{t_e}$ such that Lasts(eating, r) holds at g. Since $u \leq g < u + f \leq u + (\mathrm{T_t} + 2\mathrm{T_e})$, this branch of the proof is also concluded.

Let us now consider the case in which $\mathrm{WithinP_{ii}}(\mathrm{Becomes}(\mathsf{eating}) \vee (\exists t > \mathrm{t_e} : \mathrm{Lasted}(\mathsf{eating}, t)), \mathrm{T_t} + \mathrm{T_e})$ holds at u. This means that there is a generic time instant $v : u - (\mathrm{T_t} + \mathrm{T_e}) \leq v \leq u$ where Becomes(eating) or $\exists t > \mathrm{t_e} : \mathrm{Lasted}(\mathsf{eating}, t)$ holds. In the first case, axiom **eat_till_release (4)** lets us conclude that eating lasts for a time at least as long as $\mathrm{t_e}$ starting from time instant v. Since $v \geq u - (\mathrm{T_t} + \mathrm{T_e}) \geq u - (\mathrm{T_t} + 2\mathrm{T_e})$, this branch of the proof is concluded. In the second case, eating was true for $r > \mathrm{t_e}$ time units, starting from v in the past; hence the whole branch of the proof is concluded.

The case NowOn(F) is proved similarly to the first branch, but with different base time instantiations and some technicalities that we do not discuss here. □

This concludes the proof of $\mathcal{AX}_{\mathbf{phil}} \cup \mathcal{AS}_{\mathbf{phil}} \vdash \mathcal{TH}_{\mathbf{phil}}$, i.e., step 1 of Section 5.4.

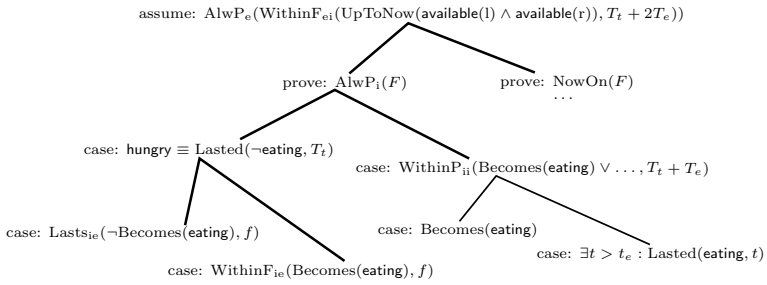

Figure 6.3.: Proof structure for Theorem **regular_eating_rg**

6.2.4. Global Rely/Guarantee Specification

Let us now complete the *composite specification* of the dining philosophers.

More precisely, we formulate the global property to be verified, which is expressed by theorem **liveness_rg** of the composite class **dining_N**. It simply states that each philosopher in the array eats regularly, unless he/she has not started yet. Notice that in our example $E_{\text{glob}} = E_{\textbf{dining\_N}} = \text{true}$ since the composite system is closed, and $M_{\text{glob}} = M_{\textbf{dining\_N}}$ coincides with the following formula.

Theorem 24 (dining_N.liveness_rg).
$\forall k \in [0..N-1]:\quad (\,\text{SomP}_\text{i}(\textbf{Philosophers}[k].\mathsf{start})$
$\Rightarrow\ (\exists t > \text{t}_\text{e} : \text{Within}_\text{ii}(\text{Lasts}(\textbf{Philosophers}[k].\mathsf{eating}, t)\,, \text{T}_\text{t} + 2\text{T}_\text{e}))\,)$

6.2.5. Global Verification

Having completed the global composite specification, so that we are now able to consider step 2 of the verification process. Each local assumption is discharged by a visible theorem or axiom of the modules adjacent to the current philosopher, as shown in Figure 6.4. Therefore, step 2 is completed without circularities involved, since **thinking_duration** is an axiom and **taking_turns[_2]** are both proved directly from axioms local to each class (i.e., they do not rely circularly on other assumptions).

Step 3 is empty in our example, since the only global theorem we have to prove is theorem **liveness_rg** in $\mathcal{TH}^{\text{rg}}_{\textbf{dining\_N}}$.

Let us consider step 4, where in particular we prove that local rely/guarantee assumption E_k is initialized, that is we show that hypothesis 4 of Proposition 5.3.8 holds for E_k. Since the local theorems have already

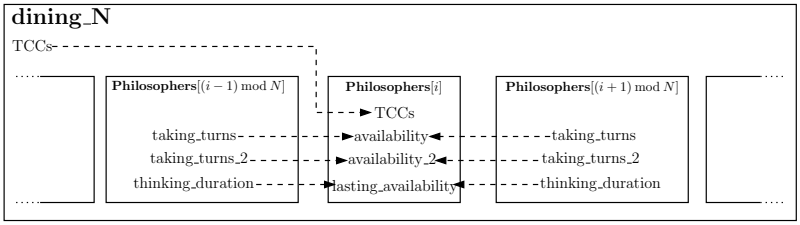

Figure 6.4.: Discharging of assumptions in global class **dining_N**

been proved without circularities, we can use **fork_availability** to complete this step. The theorem simply states that the desired property $E_k = \text{WithinF}_{\text{ei}}(\text{UpToNow}(\mathsf{available}(\text{l}) \wedge \mathsf{available}(\text{r})), \text{T}_\text{t} + 2\text{T}_\text{e})$ always holds, which subsumes the initialization condition, and requires no explicit mention to initialization predicate S.

Step 5 requires to discharge the E_k's by means of other formulas, in order to fulfill hypothesis 2 of Proposition 5.3.8. In this case as well, theorem **fork_availability** for module k works correctly since it predicates the validity of the E_k's over the whole temporal axis.

Step 6 is also very simple, since $M_{\text{glob}} = \forall k \in \{1, \ldots, n\} : M_k$ in our case, so that the implication of this step holds trivially. As a consequence, hypothesis 3 of Proposition 5.3.8 is shown to hold.

Finally, let us recall that hypothesis 1 of Proposition 5.3.8 is subsumed by step 1 of our method, since the formulas $E_k \gg M_K$ are local theorems **regular_eating_rg**

As discussed above and shown in the proof dependencies pictures, no circularities arise in proving the local formulas, so we conclude that theorem **liveness_rg** soundly holds as a consequence of the inference rule of Proposition 5.3.8 and according to the steps in Section 5.4.

6.2.6. Discussion

As hinted above, the actual verification was carried out with PVS support: this section aims at sketching an analysis of the complexity of the process. First, however, let us notice that the bare number of proof commands is hardly a complexity measure of some interest for comparing compositional rules and methods, as it is often strongly influenced by factors which do not pertain to the methodology, such as the encoding of the logic in the proof checker, the number of additional auxiliary lemmas generated in building

the proofs, etc. Conversely, it would be meaningful to compare our proof with a non-compositional one, carried out with the same basic TRIO/PVS prover, in order to understand the benefits of a compositional methodology in building our verification, and in order to assess the significance of the philosophers example as a benchmark for compositional verification. Indeed, there are cases where non-compositional methods may perform better in practice then compositional ones; in such cases the additional verification burden required by a compositional framework is not traded off by any benefit in proof simplification or reuse [dRdBH+01, Lam98].

Under this respect, let us compare the complexity of our verification with the cost of a non-compositional one. The basic problem with a non-compositional proof is that we cannot exploit encapsulation and reuse. Therefore, there is no distinction between local and global items and everything is "flattened" at the same level of visibility. In the case of the dining philosopher problem, we can overcome this problem by "simulating" modularization at the global level. In other words, we have to carefully parameterize each item with respect to an index which separates different "instances" of the philosopher.

Moreover, and most importantly, we must devise a way to replace the use of the $\gg$ operator by temporally closed formulas only. As we pointed out above, writing rely/guarantee formulas using the $\gg$ operator permits to use an *ad hoc* inference rule which can handle circularities between assumptions and guarantees of the various local formulas without need for an *explicit* circularity breaking to be provided by the verifier. Therefore, replacing specifications written using the time progression operator is more difficult the tighter the circularity between the local E_k's and M_k's is. This is usually the case when we compose modules which are instances of the *same class*: having the same formulas by definition, the circularity is likely to arise between connected classes for symmetry reasons. This was the case in the philosopher example, albeit with a significant simplification, namely the fact that the local assumptions E_k's can be shown to hold over the whole temporal axis, as discussed above. This simplification made it possible to build a non-compositional solution, though still with considerable effort. More generally, we point out that a system made of a (possibly large) set of instances of the same class, connected with some circularities, is representative of a vast category of real systems. In particular, those representing communication protocols between a large number of hosts, such as peer-to-peer protocols. Thus, we believe that the philosophers problem

can be regarded as a good abstraction of some of the core problems arising in verifying such systems.

We carried out the unstructured, non-compositional proof of the philosophers problem;[2] the result has been a proof of length roughly comparable to — or even a bit shorter than — the compositional one, but fragmented into more intermediate lemmas, with more assumptions and more intricate proof dependencies. Another feature that distinguishes the compositional proof from the non-compositional one is the fact that the former is repetitive while the latter is intricate. In other words, the compositional proof has a transparent, intelligible structure made of several similar parts, indicating that it is indeed simpler to manage for the human user who can easily understand when previous proof patterns can be applied again with minor modifications. All in all, even if the number of proof commands was not dramatically different in the two proofs, the complexity of the non-compositional one, considering also the mental effort and difficulty in managing the proof, was much greater. Furthermore, our experience with the compositional proof of the same property has guided and helped the building of the non-compositional one: we believe that doing the non-compositional proof first would have been really hard and time-consuming.

6.3. A Peer-to-Peer Communication Protocol

This section illustrates our practical approach to compositionality through another example: the compositional verification of the peer to peer communication protocol BitTorrent (BT) [Coh01, Coh03]. Actually, we model and verify only a very small part of the behaviors allowed by the complex protocol, in a specific situation — which is nonetheless likely to happen in practice —, and we provide a high-level description which abstracts away from most implementation details, focusing on timing aspects.[3]

The overall goal of this example is not only to show the application of the inference rules in practice, but also to demonstrate our general approach to compositionality, as laid out in the previous chapters. To this end, we provide two different, alternate verifications of the protocol. The first approach is the most straightforward and standard: we select a suitable

[2] It is available in PVS form [FRMM05c].

[3] For another, completely different example of verification of the BitTorrent protocol, based on the approximation of process algebras through differential equations, we refer the reader to [Dug06].

inference rule, among those provided in the previous chapter, and we apply it to verify the system. Although compositionality succeeds in shifting the most part of the verification burden from global to local, the application of the inference rules is nonetheless highly nontrivial, and requires some considerable ingenuity.

The second approach is instead more unconventional, in that it does not use any previously defined inference rule. On the contrary, it builds a new inference rule by varying the existing rules in a way which fits the particular specification we are trying to verify. In this case the verification effort required in applying the new rule is minimal, although this is of course traded-off against ingenuity required in designing the new inference rule, and in verifying its correctness.

Then, we conclude this section with a brief comparison between the two approaches that highlights the advantages and disadvantages of both. We believe that our example, however small, constitutes an evidence of the importance of having a flexible approach to compositionality, and of the necessity of focusing on real examples of applications on a case-by-case basis, rather than looking for a "one-size-fit-all" compositional rule.

In practice, the example is carried out as follows. After providing suitable axiomatic specification of the classes involved in describing the protocol, we state a global theorem that we want to prove about the composition of the various classes. Then, we provide two different proofs of the theorem, both according to the general methodology of Section 5.4 The first proof is an application of the inference rule 5N of Proposition 5.3.11, that we have discussed above. The second proof, instead, pursues in full the practical approach to compositionality that we have advocated in Section 3.2. Therefore, instead of figuring out how to fit our system specification within a predefined inference rule, we develop a new rule, combining the known results about the previously introduced rules with the specific structure of the formulas that model our system.

6.3.1. Protocol Basics

Let us describe the basic features of the BT protocol, namely those that we are going to specify and use for verification.

BT is a peer-to-peer communication protocol for distributing files over networks among a number of hosts. Hosts are partitioned among seeds and peers. Let N_s be the number of *seeds*, and N_p be the number of *peers* in the system. Usually, it is $N_p \gg N_s$, although this is not necessary for the

correct functioning of the protocol.

When the protocol is started, each seed possesses an entire copy of the file to be distributed; let us represent the file as split into P packets, numbered from 0 to $P-1$. The peers, instead, do not initially have any part of the file; their goal is to get it. In order to do that, each peer sends periodically a (random) packet to any peer it is connected to. After a peer has received (and thus stored) at least a packet, it also periodically sends a packet to any other peer it is connected to. The connections among peers and between seeds and peers change dynamically and are coordinated by a dedicated host called *tracker*. For simplicity, we do not represent explicitly the tracker in our model, but simply specify minimal assumptions about the dynamics of connections between hosts.

Let us describe how the connections between hosts are managed in our model; recall that this is a special case of the real BT protocol. At any time, the set of all peers is partitioned into a number of clusters. In each cluster, the peers are connected in a circle, and there is at least one peer which is connected to a seed and can receive data from it. Moreover, the connections change only periodically. More precisely, we will ensure that, before the clusters are changed, every peer in the cluster has received at least a packet (that is there has been at least a "passing around" of packets over the circle of peers in the cluster).

Notice that the connections do not represent physical connections, which would hardly be dynamic; instead, they represent coordination between peers. Moreover, we neglect the transmission time over the connections, which are considered instantaneous (this is acceptable, since the transmission times can be modeled directly in the specification of each host as re-transmission delays).

6.3.2. System Specification

Let us now introduce the classes of our BT specification. Notice that in the remainder we describe a real-time strengthening of the real BT protocol; the latter does not have any hard real time constraint built in, as it relies on the TCP/IP protocol which is "best-effort".

The seed class. The seeds in the system are represented by instances of the **seed** class, which is pictured in Figure 6.5. This class is parametric with respect to the number P of packets of the file, and the send time T_s. Moreover, it has a single event item $\mathsf{send}(i)$, which is true whenever

the seed sends the packet number $i \in \{0, \ldots, P-1\}$ over its connections. The axiomatization of the class is very simple: every T_s time units (at most), a packet is nondeterministically sent over the connections. This is represented by the following axiom.[4]

Axiom 25 (**seed.sending**). $\exists i : \mathrm{WithinF}(\mathsf{send}(i), T_s)$

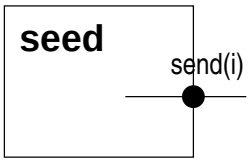

Figure 6.5.: Interface of the **seed** class

The peer class. The peers are represented by instances of the **peer** class, pictured in Figure 6.6. The **peer** class is also parametric with respect to the number P of packets of the file, and the send time T_s. It has a send event, representing its sending a packet over its connection, and a recv event, representing the receiving of a packet (from another peer or from a seed). Moreover, the class has a non-visible stored state: $\mathsf{stored}(i)$ being true represents the fact that the packet i has been received and is therefore stored locally; this is formalized by the following axioms (i is a variable of type $\{0, \ldots, P-1\}$).

Axiom 26 (**peer.sending**). $\mathsf{stored}(i) \Leftrightarrow \mathrm{SomP_i}(\mathsf{recv}(i))$

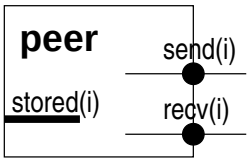

Figure 6.6.: Interface of the **peer** class

The send and recv actions happen according to axioms **peer.recv_to_send** and **peer.send_to_recv**: the former states that a recv triggers a send of

[4] In the remainder of the specification, we do not mention explicitly the types of the predicates arguments whenever this is unambiguous. For example, we simply write $\mathsf{send}(i)$ assuming implicitly that $i \in \{0, \ldots, P-1\}$.

a (nondeterministically chosen) packet within T_s time units; the latter conversely states that a send only happens if a packet has been received in the past, and the packet is stored locally.

Axiom 27 (peer.recv_to_send). $\exists i : \mathsf{recv}(i) \Rightarrow \exists j : \mathrm{WithinF}(\mathsf{send}(j), T_s)$

Axiom 28 (peer.send_to_recv).
$\mathsf{send}(i) \Rightarrow \mathsf{stored}(i) \ \wedge \ \exists j : \mathrm{WithinP}(\mathsf{recv}(j), T_s)$

The system class. Now, let us describe how the modules are composed into a global system, representing the network. The seeds are N_s instances of the **seed** class that populate an array named Se, and the peers are N_p instances of the **peer** class that populate an array named Pe.

To represent the connections among hosts in the system, we cannot exploit TRIO connections which are static. On the contrary, we introduce two state predicates $\mathsf{connected_p}(p_1, p_2)$ and $\mathsf{connected_s}(s, p)$ to indicate connections between two peers, and between a seed and a peer, respectively. The meaning of the two predicates is obvious: whenever i is connected to j, it means that what i sends, it is received by j. This is stated by the global axioms **sp_connections** and **pp_connections**.

Axiom 29 (sp_connections).
$\mathsf{connected_s}(s, p) \Rightarrow (\mathsf{Se}[s].\mathsf{send}(i) = \mathsf{Pe}[p].\mathsf{recv}(i))$

Axiom 30 (pp_connections).
$\mathsf{connected_p}(p_1, p_2) \Rightarrow (\mathsf{Pe}[p_1].\mathsf{send}(i) = \mathsf{Pe}[p_2].\mathsf{recv}(i))$

Finally, we have to define how the connections change over time. This is an important part of the specification, and requires some nontrivial details.

Let $K = \{0, \ldots, N_p - 1\}$ be the set of all peers. We postulate that, at any time, there exist k (time-dependent) sets $K_1, \ldots, K_k$, named *clusters*, such that:

1. The clusters form a partition of K;
2. Each cluster is such that all its peers are connected to form a ring;
3. For each cluster, there is *at least* a peer in the cluster which is connected to some seed.

Conditions C(1–3) are formalized by axiom **clustering** below. In order to present it, we first introduce the following notation.[5]

- The successor of index p modulo m (i.e., over a set of m elements numbered from 0 to $m-1$) is denoted by $\text{next}_m(p) \equiv (p+1) \bmod m$.
- We denote a permutation of the m-element set $\{0, \ldots, m-1\}$ as a function $\pi_m : \{0, \ldots, m-1\} \to \{0, \ldots, m-1\}$.
- We associate to every cluster K_i its size $k_i \equiv |K_i|$.

Axiom 31 (**clustering**).

$$\exists 0 < k \leq N_p : \exists K_1, \ldots, K_k :$$

$$\textit{(C1)} \left(\begin{array}{l} \forall i \in \{1, \ldots, k\} : K_i \subseteq K \ \wedge \bigcup_{i=1,\ldots,k} K_i = K \\ \wedge \ \forall i, j \in \{1, \ldots, k\} : i \neq j \Rightarrow K_i \cap K_j = \emptyset \end{array} \right) \wedge$$

$$\textit{(C2)} \left(\begin{array}{l} \forall i \in \{1, \ldots, k\} : \exists \widetilde{\pi}_{k_i} : \forall j \in \{0, \ldots, k_i - 1\} : \\ \qquad \mathsf{connected_p}(\widetilde{\pi}_{k_i}(j), \widetilde{\pi}_{k_i}(\text{next}_{k_i}(j))) \end{array} \right) \wedge$$

$$\textit{(C3)} \left(\begin{array}{l} \forall i \in \{1, \ldots, k\} : \exists j \in \{0, \ldots, k_i - 1\} : \\ \qquad \exists s \in \{0, \ldots, N_s - 1\} : \ \mathsf{connected_s}(s, j) \end{array} \right)$$

Notice that, from now on, with a little abuse of notation, we will treat the K_i's as globally available items, that is we will be able to reference them in any module of the system; notice that this is only a shortcut to represent easily the sets, whose precise formalization in TRIO would not be difficult but would introduce some additional notational overhead.

6.3.3. Rely/Guarantee Specifications

Let us now describe a global specification property of the system. Then, we also provide suitable specifications local to each peer module, from which the correctness of the global specification can be inferred compositionally.

The form of the local specifications depends on the inference rule that we intend to use. Therefore, in this section we first of all introduce the

[5] In the following axiom, we slightly abuse TRIO notation by using higher-order quantifications, namely on the functions/sets K_i and π_{k_i}. Notice, however, that the same meaning can be retained with suitable first-order predicates and quantifications, but with a much more cumbersome notation, which we prefer to avoid for clarity.

global specification, and then we provide two different sets of local specifications: Section 6.3.3 describes those suitable to apply the inference rule of Proposition 5.3.11, whereas the following Section 6.3.3 introduces a new *ad hoc* inference rule, and shows how the basic specification of the system can be seen as a local rely/guarantee specification suitable to be used with this new inference rule.

Global Specification

Our overall verification goal is to show a liveness property about the peers, that is that every peer eventually sends a packet, within an upper bound equal to the value $N_p T_s$. We choose this particular property as it allows us to focus on temporal properties, and thus to demonstrate best the use of our inference rules for temporal logic. An orthogonal concern, which we omit for brevity, would be the proof of the the correctness of the protocol. However, let us notice that similar, compositional techniques could be used for this purpose as well.

Intuitively, our verification goal cannot be guaranteed without introducing constraints on how the connections among peers vary. Basically, we need that the connections do not change "too often", if we want the system to actually converge towards some goal. More precisely, let us introduce a predicate $\mathrm{const}(p, t)$ which holds whenever its first argument keeps its current value over an interval of given length given by its second argument. Therefore, for any time-dependent predicate F with domain D, we have the following definition (as usual, *bb* can be any of *ee*, *ei*, etc.).

$$\mathrm{const}_{\mathrm{bb}}(F, T) \;\equiv\; \exists d \in D : \mathrm{Lasts}_{\mathrm{bb}}(F(d), T)$$

Through this notation, we formulate the following global specification in a rely/guarantee form: if the connections stay unchanged over a time interval of length $N_p T_s$, then every peer eventually sends a packet.

Theorem 32 (global_send_liveness).

$$\mathrm{const}_{\mathrm{ie}}(\mathsf{connected_p}(p_1, p_2) \wedge \mathsf{connected_s}(s, p), N_p T_s) \quad \Rightarrow \\ \mathrm{WithinF}_{\mathrm{ie}}(\exists i : \mathsf{Pe}[p].\mathsf{send}(i)\,, N_p T_s)$$

Local Specifications for a Predefined Inference Rule

In order to apply the inference rule 5N of Proposition 5.3.11, we now set up suitable rely/guarantee specifications, local to each **peer** class.

The following theorem **peer.bounded_send_recv** provides such a specification. As usual, $i, j \in \{0, \ldots, P-1\}$ and k indicates the size of the cluster the current module is in.

Theorem 33 (**peer.bounded_send_recv**).
$\exists j : \mathrm{WithinF}(\mathsf{recv}(j), kT_s) \Rightarrow \exists j : \mathrm{WithinF}(\mathsf{send}(j), kT_s)$

This formula is probably a bit counterintuitive, since it establishes a relationship between two events that are both in the future. However remember that we are dealing with an abstract specification.

Now that we have introduced the local rely/guarantee specification through theorem **peer.bounded_send_recv**, we have to prove it by means of the other formulas of the **peer** class, to show that it is indeed a valid local specification. This correspond to step 1 of the method of Section 5.4.

In order to do that, we introduce an assumption formula which we will discharge by means of axioms of other **peer** classes. The formula, named **peer.late_recv**, states that if a late (i.e., later than $(k-1)T_s$ time units) recv is received, a send is also present within the overall kT_s time period.

Assumption 34 (**peer.late_recv**).
$\exists j : \mathrm{Futr}(\mathrm{WithinF}(\mathsf{recv}(j), T_s), (k-1)T_s) \Rightarrow \exists j : \mathrm{WithinF}(\mathsf{send}(j), kT_s)$

The need for this assumption will be clear in the following proof.

*Proof of Theorem **peer.bounded_send_recv**.* For simplicity, let 0 be the current time instant. Assume the antecedent of the implication holds, that is that $\mathsf{recv}(i)$ holds at a time instant $0 < t < kT_s$ from the current time instant, for any i. We distinguish two cases: whether $t > (k-1)T_s$ or not.

In the former case, $\mathsf{recv}(i)$ holds at a time instant $(k-1)T_s < t < T_s$. This means that at $(k-1)T_s$ the formula $\mathrm{WithinF}(\mathsf{recv}(i), T_s)$ holds, which corresponds to the antecedent of formula **peer.late_recv**. Therefore the consequent holds, which is what we have to prove.

The latter case is the one in which the time instant in which $\mathsf{recv}(i)$ holds is $t \leq (k-1)T_s$. Let us consider axiom **peer.recv_to_send** at time t. We can infer that $\mathrm{WithinF}(\mathsf{send}(j), T_s)$ at t, for some j. In other words, there is a time instant $t < u < t+T_s$ such that $\mathsf{send}(j)$ at u. It is simple to realize that $u < (k-1)T_s + T_s = kT_s$, in this branch of the proof. Therefore, it is true that $\mathrm{WithinF}(\mathsf{send}(j), kT_s)$ at the current time. □

Making Up a New Inference Rule

In accordance with the guidelines we have proposed in the previous sections, let us find suitable rely/guarantee inference rule and local specifications according to try to the goal of our verification, stated in Section 6.3.3.

Local specifications. Let us seek a suitable formula to serve as local specification of the modules. More precisely, we need a specification for each peer, as we have many of them in each cluster. We notice that axiom **peer.recv_to_send** is a local specification about the behavior of the send item in response to a behavior of the recv. Therefore, it is a suitable rely/guarantee local specification.

A suitable compositional inference rule. Let us now build a suitable inference rule to prove the truth of the global specification theorem **global_send_liveness** from the truth of the local specifications axioms **peer.recv_to_send**. Let us summarize the characteristics that it should possess.

- it should be a rely/guarantee rule, as we have both a global specification and the local specifications that are in rely/guarantee form;
- the link between the assumptions and the guarantees of the formulas should be given by a temporal relationship expressed through the WithinF operator;
- the rule must be circular, as this mirrors the circular nature of the connections among the peers in each cluster, each of whose assumption can be satisfied by the preceding peer;
- the rule need not be self-discharging, as the assumption of each module is discharged solely by the guarantee of the module that precedes it;
- it must be non-fully compositional, as the global assumption about the constancy of the connections over time defines what is the state of the connections, and thus it is required in discharging local assumptions;
- there is no notion of initialization required, since we are proving a "liveness" property, similarly as in the inference rule of Proposition 5.3.11.

All in all, let us introduce the following circular inference rule, whose structure of assumptions and guarantee mirrors closely the one of the local and global specification formulas in our example. We provide a proof of the soundness of this inference rule in Appendix B

Proposition 6.3.1. *If, for some fixed duration* $\mathrm{T_B} > 0$*:*

1. *for all* $i \in \mathcal{I}_N$*:* $E_i \Rightarrow \mathrm{WithinF}(M_i, \mathrm{T_B})$
2. *for all* $i \in \mathcal{I}_N$*:* $E \wedge M_i \Rightarrow E_{\texttt{if } i<N \texttt{ then } i+1 \texttt{ else } 1}$
3. $\bigwedge_{i \in \mathcal{I}_N} \mathrm{WithinF}(M_i, N\mathrm{T_B}) \Rightarrow M$
4. $\mathrm{WithinF}(E_i, \mathrm{T_B})$*, for some* $i \in \mathcal{I}_N$

then $\mathrm{Lasts}(E, N\mathrm{T_B}) \Rightarrow M$*.*

6.3.4. Application of the Compositional Rule

In this section, we provide the verification of the BitTorrent example through the applications of compositional inference rules with the local rely/guarantee specifications we have presented in the previous subsection.

As we already mentioned, we provide two correctness proofs. The first one uses inference rule 5N of Proposition 5.3.11; the second one exploits the inference rule of Proposition 6.3.1 introduced beforehand.

Using a Predefined Inference Rule

Our goal is to compose the various theorems **peer.bounded_send_recv** (one for each instance of the **peer** class) using the compositional inference rule of Proposition 5.3.11 to deduce the validity of the global specification.

In our system, we can identify two levels at which to prove the liveness property for the peers: intra-cluster and inter-cluster. To wit: since at any time the set of all peers is divided into clusters, we can perform the global proof in two steps:

- first, prove the liveness property within each cluster, independently of the others;
- then, show that the composition of the intra-cluster liveness properties yields the overall desired liveness property of the system.

Therefore, we are going to first apply the compositional verification method to each cluster separately.

Intra-cluster proofs. Consider the (generic) cluster of size k and the the following choices for assumption and guarantee formulas. Notice that modules in the inference rule are numbered from 1, while peers (and seeds) are numbered from 0. However, for simplicity, we now assume to have a 0-based numbering in the modules: filling the gap is straightforward.[6]

- $E_i^k = \exists j : \mathrm{WithinF}(\mathsf{Pe}[i].\mathsf{recv}(j)\,, kT_s)$;
- $M_i^k = \exists j : \mathrm{WithinF}(\mathsf{Pe}[i].\mathsf{send}(j)\,, kT_s)$;
- $\mathrm{Lasts}_{\mathrm{ie}}\big(E^k, T_s\big) = \mathrm{const}_{\mathrm{ie}}(\mathsf{connected_p}(p_1, p_2) \wedge \mathsf{connected_s}(s, p), kT_s)$, for all p_1, p_2, p in the cluster and some seed s;
- $M^k = \bigwedge_{i=1,\ldots,k} M_i^k$;
- $N = k$ and $T_B^k = kT_s$.

Let us now apply the inference rule of Proposition 5.3.11 for the $\mathrm{Lasts}_{\mathrm{ie}}$ and $\mathrm{WithinF}_{\mathrm{ie}}$ operators. Therefore, the we split the proof into lemmas, each corresponding to a condition of the inference rule, and thus also to steps 1 and 4 through 6 of the compositional method of Section 5.4. More precisely, notice that step 5 is empty in our case, since there is no requirement on assumption discharging for the inference rule 5N of Proposition 5.3.11.

Lemma 6.3.2 (Proof step 1)**.**
$\exists j : \mathrm{WithinF}(\mathsf{Pe}[i].\mathsf{recv}(j)\,, kT_s) \Rightarrow \exists j : \mathrm{WithinF}(\mathsf{Pe}[i].\mathsf{send}(j)\,, kT_s)$

Lemma 6.3.3 (Proof step 6)**.** $E^k \wedge \bigwedge_i \exists j : \mathrm{WithinF}(\mathsf{Pe}[i].\mathsf{send}(j)\,, kT_s)$
$\Rightarrow \bigwedge_i \exists j : \mathrm{WithinF}(\mathsf{Pe}[i].\mathsf{send}(k)\,, kT_s)$

Lemma 6.3.4 (Proof step 4)**.**
$\mathrm{WithinF}_{\mathrm{ie}}(\bigwedge_i((\exists j : \mathrm{WithinF}(\mathsf{Pe}[i].\mathsf{recv}(j)\,, kT_s))$
$\vee\, (\exists j : \mathrm{WithinF}(\mathsf{Pe}[i].\mathsf{send}(j)\,, kT_s))), kT_s)$

For brevity, we present the proofs of the three lemmas in Appendix B.

[6] We add the superscript k for notational clarity.

Conclusion of intra-cluster proofs. So far, we have proved that, for each cluster K_i of size k_i:

$$\mathrm{const}_{\mathrm{ie}}(\mathsf{connected_p}(p_1,p_2) \wedge \mathsf{connected_s}(s,p), k_i T_s) \Rightarrow \mathrm{WithinF}_{\mathrm{ie}}\left(\bigwedge_{j\in K_i} \exists l : \mathrm{WithinF}(\mathsf{Pe}[j].\mathsf{send}(l), k_i T_s), k_i T_s\right)$$

Actually, an analysis of the performed proofs shows that all the considered events occurred within a time bound of $k_i T_s$ time units, that is, equivalently, the outermost WithinF is always instantiated at the current time. In other words, we can actually claim a strengthening of the above formula, and namely

$$\mathrm{const}_{\mathrm{ie}}(\mathsf{connected_p}(p_1,p_2) \wedge \mathsf{connected_s}(s,p), k_i T_s) \Rightarrow \bigwedge_{j\in K_i} \exists l : \mathrm{WithinF}_{\mathrm{ie}}(\mathsf{Pe}[j].\mathsf{send}(l), k_i T_s)$$

Proof of step 2. We also notice that we can discharge the Assumption **peer.late_recv** intra-cluster. This corresponds to completing step 2 of the method of Section 5.4. The proof of it is similar to that of Lemma 6.3.4, and is therefore not discussed. Moreover, notice that the discharging does not involve any circularity, as it ought to be.

Proof of step 3. Finally, let us note that step 3 is empty in our case, as we have no non-rely/guarantee global theorem. This concludes the verification at the intra-cluster level.

Inter-cluster proofs. We are now ready to put together the various intra-cluster proofs to deduce the validity of theorem **global_send_liveness**.

Before doing that, let us state simple proper of the operators $\mathrm{const}(\cdot,\cdot)$ and WithinF. The proofs are very simple and are therefore omitted.

Lemma 6.3.5. *For any n predicates $P_1,\ldots,P_n$, time distances $T_1,\ldots,T_n$ and $T \geq \max_{i=1,\ldots,n} T_i$:*

$$\mathrm{const}_{\mathrm{bb}}\left(\bigwedge_{i=1,\ldots,n} P_i, T\right) \Rightarrow \bigwedge_{i=1,\ldots,n} \mathrm{const}_{\mathrm{bb}}(P_i, T_i)$$

Lemma 6.3.6. *For any nm predicates P_i^j for $i = 1, \ldots, n$ and $j = 1, \ldots, m$, time distances $T_1, \ldots, T_n$ and $T \geq \max_{i=1,\ldots,n} T_i$:*

$$\bigwedge_{i=1,\ldots,n} \bigwedge_{j=1,\ldots,m} \mathrm{WithinF}_{\mathrm{bb}}\left(P_i^j, T_i\right) \Rightarrow \bigwedge_{\substack{i=1,\ldots,n \\ j=1,\ldots,m}} \mathrm{WithinF}_{\mathrm{bb}}\left(P_i^j, T\right)$$

Finally, we prove the global liveness theorem.

Proof of theorem ***global_send_liveness****.* Assume the antecedent

$$\mathrm{const}_{\mathrm{ie}}(\mathsf{connected_p}(p_1, p_2) \wedge \mathsf{connected_s}(s, p), N_p T_s)$$

By axiom **clustering**, it is obvious that $k_i \leq N_p$, for every i and at any time. Therefore, $\max_i k_i T_s \leq N_p T_s$, and we can apply Lemma 6.3.5, so $\bigwedge_{i=1,\ldots,k} \mathrm{const}_{\mathrm{ie}}(\mathsf{connected_p}(p_1, p_2) \wedge \mathsf{connected_s}(s, p), k_i T_s)$ holds.

This means that the antecedents of every intra-cluster result is satisfied. Therefore, we have that:

$$\bigwedge_{i=1,\ldots,k} \bigwedge_{j \in K_i} \exists l : \mathrm{WithinF}_{\mathrm{ie}}(\mathsf{Pe}[j].\mathsf{send}(l), k_i T_s)$$

Now, we apply Lemma 6.3.6 and finally conclude:

$$\bigwedge_{p=1,\ldots,N_p} \exists l : \mathrm{WithinF}_{\mathrm{ie}}(\mathsf{Pe}[p].\mathsf{send}(l), N_p T_s)$$

where the last rearrangement of indices is justified by the fact that the K_i's are a partition of the set of all N_p peers (axiom **clustering**). □

Using a Made Up New Inference Rule

We already have all the ingredients to apply the compositional inference rule of Proposition 6.3.1 in order to prove theorem **global_send_liveness**.

As we did in the previous proof, let us split the proof between intra-cluster and inter-cluster proofs. Actually, the inter-cluster proof is simple, and it is just like as it was done in Section 6.3.4, the only difference being that we now deal with *ee* variations of the $\mathrm{const}_{\mathrm{ee}}(\cdot, \cdot)$ and $\mathrm{WithinF}_{\mathrm{ee}}$ operators. This is however immediately reducible to the previously seen case, as it is simple to check that $\mathrm{const}_{\mathrm{ie}}(\cdot, \cdots)$ implies $\mathrm{const}_{\mathrm{ee}}(\cdot, \cdot)$ for the same arguments, and $\mathrm{WithinF}_{\mathrm{ee}}$ implies $\mathrm{WithinF}_{\mathrm{ie}}$, for the same argument. Thus, let us just carry out the intra-cluster proof.

Intra-cluster proof. Let us consider a generic cluster of size k; let us instantiate the inference rule of Proposition 6.3.1 as follows. Notice that modules in the rule are numbered from 1, while peers (and seeds) are numbered from 0. Thus, we associate the number $i+1$ in the rule to peer number i.

- $E_i = \exists j : \mathsf{Pe}[i-1].\mathsf{recv}(j)$;
- $M_i = \exists j : \mathsf{Pe}[i-1].\mathsf{send}(j)$;
- $E = \mathrm{const}_{\mathrm{ee}}(\mathsf{connected_p}(p_1, p_2) \wedge \mathsf{connected_s}(s, p), kT_s)$, for all p_1, p_2, p in the cluster, and some seed s;
- $M = \bigwedge_{i \in \mathcal{I}_N} \mathrm{WithinF}(M_i, kT_s)$;
- $N = k$ and $\mathrm{T_B} = T_s$.

Applying the inference rule of Proposition 6.3.1 amounts to the following steps, respectively corresponding to steps 1, 5, 6, and 4 of Section 5.4:

1. Prove that: $\exists j : \mathsf{Pe}[i-1].\mathsf{recv}(j)$
   $\Rightarrow \mathrm{WithinF}(\exists j : \mathsf{Pe}[i-1].\mathsf{send}(j), T_s)$, for all $i \in \{1, \ldots, k\}$.
2. Prove that: $E \wedge \exists j : \mathsf{Pe}[i-1].\mathsf{send}(j)$
   $\Rightarrow \exists j : \mathsf{Pe}[\mathtt{if}\ i-1 < k-1\ \mathtt{then}\ i\ \mathtt{else}\ 0].\mathsf{recv}(j)$,
   for all $i \in \{1, \ldots, k\}$.
3. Prove that: $\bigwedge_{i \in \mathcal{I}_N} \mathrm{WithinF}(\exists j : \mathsf{Pe}[i-1].\mathsf{send}(j), kT_s)$
   $\Rightarrow \bigwedge_{i \in \mathcal{I}_N} \mathrm{WithinF}(\exists j : \mathsf{Pe}[i-1].\mathsf{send}(j), kT_s)$.
4. Prove that: $\mathrm{WithinF}(\exists j : \mathsf{Pe}[i-1].\mathsf{recv}(j), \mathrm{T_B})$,
   for some $i \in \{1, \ldots, k\}$.

We now orderly provide the proof justifications.

1. For any $i \in \{1, \ldots, k\}$, this is exactly axiom **peer.recv_to_send** for module $i-1$.
2. For any $i \in \{1, \ldots, k\}$, this is immediately implied by the connection axiom **clustering**, together with the definition of the connection predicate in axiom **pp_connections**, for module $i-1$ and its successor in the clustering.

3. This is trivial as the left-hand side of the implication is identical to the right-hand side.

4. This is a direct consequence of axiom **clustering** again, together with axiom **seed.sending**.

Finally, note that steps 2 and 3 are empty, since we have neither local assumption formulas to prove, nor global non-rely/guarantee theorems. □

Discussion. Let us compare the two different verification processes we have detailed in this section.

The first approach used a previously defined inference rule, namely that of Proposition 5.3.11. As we have seen, the application of the rule has been nontrivial, and has required some considerable ingenuity, as well as the proof of several details. On the other hand, the application of the inference rule in the second verification has been straightforward and with little technical complications. This does not mean that in this case the complexity of verification has "disappeared"; in fact, we had to invest considerable ingenuity in devising a new compositional inference rule, one that was simple to apply for the system being verified. So, the two different solutions have resolved the trade-off between difficulty in building a rule and difficulty in applying it in two different ways.

In general, it is difficult — and perhaps highly subjective as well — to assess which of the two approaches is preferable. Ideally, one should consider both approaches, and choose according to the particular features of the system under consideration, as well as his/her specific skills and attitudes. All in all, the most important "lesson" that we can draw from this example is about the importance of flexibility in pursuing compositional verification, in accordance with what we have extensively advocated in the previous chapters.

Part II.

Integration

7. Integration as a Generalization of Compositionality

The first part of this thesis developed a framework for the compositional specification and verification of real-time modular systems, based on the language TRIO, a very expressive formal language, which allows one to deal with different aspects and different systems.

Nonetheless, the ultimate goal of applying formal methods to the description and analysis of large and heterogeneous systems inevitably requires being able to model different parts of the system using disparate notations and languages. In fact, it is usually the case that different modules are naturally described using diverse formal techniques, each one tailored to the specific nature of that component.

Indeed, the past decades have seen the birth and proliferation of a plethora of different formal languages and techniques [FM05, FMMR07], each one usually focused on the description of a certain kind of systems and hinged on a specific approach. On the one hand, this proliferation is advantageous, as it allows the user to choose the notation and methodology that is best suited for her needs and that matches her intuition. However, this is also inevitably a hurdle to the true scalability in the application of formal techniques, since we end up having heterogeneous descriptions of large systems, where different modules, described using distinct notations, have no definite global semantics when put together. Therefore, we need to find ways to *integrate* dissimilar models into a global description which can then be analyzed.

Integration can be framed as a natural *generalization* of the idea of compositionality: whereas the latter deals with specifying and verifying systems whose modules are formalized with the same language, integration aims at providing ways to specify and verify systems composed of modules formalized in different languages. Different languages means, in general, not only differences of syntax, but also — and more crucially — of semantics. Thus, two modules specified with languages referring to different semantics models have no *a priori* defined meaning when put together.

As an example, let us consider a module describing a queue as an abstract data type, specified in a formal language such as Larch [GHGJ93]. Then, let us consider another component consisting of a real-time scheduler that uses a queue to store incoming data, whose temporal behavior is described with, say, a temporal logic such as TRIO. It is not clear at all how the verification results about functional properties of the queue data type can be applied to proving properties of the scheduler manipulating the queue.

However sketchily, this shows that integration techniques must relate formally the interpretations of the various languages, in a way which is consistent. Specifying exactly and formally what is the relation between the formal interpretation is very important, as this passage is often done in practice only informally, or through some rule of thumb [BGH+04]. In particular, the relation should allow the verifier to infer facts in a certain semantics, deriving them from properties that hold in another semantics. This should be done without introducing new formalisms to represent the heterogeneous system: integration stresses separation of concerns in the development of a specification, as the modules describing different parts of the systems, that obey different models, may be developed independently and then joined together to have a global description of the system, in the same vein as compositionality.

Integration for hybrid systems. The description of real-time systems, which requires a quantitative modeling of time, introduces a particularly relevant instance of the problem of integration. Commonly, real-time systems are composed of some parts representing physical environmental processes and some others being digital computing modules. The former ones have to model physical quantities that vary continuously over time, whereas the latter ones are digital components that are updated periodically at every (discrete) clock tick. Hence, a natural way to model the physical processes is by assuming a *continuous-time* model, and using a formalism with a compliant semantics, whereas digital components would be best described using a *discrete-time* model, and by adopting a formalism in accordance. Thus, the need to integrate continuous-time formalisms with discrete-time formalisms, which is the goal of this second Part.

Systems where continuous- and discrete-time components coexist are sometimes called *hybrid systems* [Ant00]. Let us notice that our notion of hybrid system is different than that of *hybrid automata* [AHLP00]. In fact, in hybrid automata the same system undergoes an evolution which

consists of alternations between continuous-time evolution (called "flow"), and discrete switching (called "jumps"); on the contrary we are interested in describing systems where the continuous-time and the discrete-time components evolve in parallel, within the same "global" time. However, we share with the work on hybrid automata the overall goal of describing systems where both discrete- and continuous-time dynamics occur.

Integration through the notion of sampling invariance. As we discussed in Section 2.1, TRIO can be interpreted over both continuous-time models and discrete-time ones: each formula of the language can be interpreted in one of the two classes of models. Therefore, our integration framework considers modules written using TRIO under different choices for the time domain.

It is not difficult to realize that the discrete-time semantics and the continuous-time one are unrelated in general, in that the same formula unpredictably changes its models when passing from one semantics to another. On the contrary, integration requires different formulas to describe parts of the *same* system, thus referring to unique underlying models.

To this end, we base our integration framework on the notion of *sampling invariance* of a specification formula, introduced in Section 8.2. Informally, we say that a temporal logic formula is sampling invariant when its discrete-time models coincide with the samplings of all its continuous-time models. The *sampling* of a continuous-time model is a discrete-time model obtained by observing the continuous-time model at periodic instants of time.

The justification for the notion of sampling invariance stems from how real systems are made. In fact, in a typical system the discrete-time part (e.g., a controller) is connected to the (probed) environment by a sampler, which communicates measurements of some physical quantities to the controller at some periodic time rate (see Figure 7.1). The discrete-time behaviors that the controller sees are samplings of the continuous-time behaviors that occur in the system under control. Our notion of sampling invariance captures abstractly this fact in relating a continuous-time formula to a discrete-time one, thus mirroring what happens in a real system.

Uses of the integration framework. Once we have a sampling invariant specification, we can integrate discrete-time and continuous-time parts, thus being able, among other things, to resort to verification in a discrete-

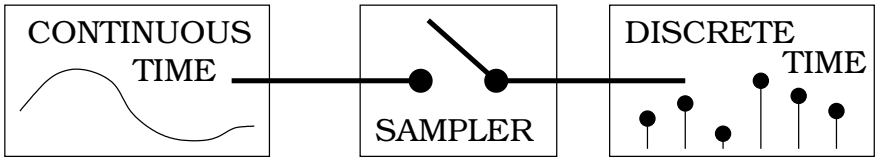

Figure 7.1.: A system with a sampler.

time model, which often benefits from more automated approaches, while still being able to describe naturally physical processes in a continuous-time model.

Another approach that can be achieved with a sampling-invariant language which is the subset of a more expressive one (TRIO, in our case) is one based on *refinement*. In the process of formal modeling of a digital component — and of a computer program in particular — one would start with a very high-level description that would refer actions and events to the "real, ideal, physical" time. The final implementation, however, will have to refer to a more concrete, *measured* view of time, such as the one achievable through periodic readings of an imperfect clock. The refinement of the specification from the ideal to the concrete could then move from full TRIO to the sampling-invariant subset of it; this latter description would be closer to the "implemented" view of time, and would therefore facilitate its realization.

A full development of these ideas is outside the scope of this thesis. Some of them have been pursued elsewhere [FRS+06], and others belong to future work. In the remainder, we develop the integration framework in Chapter 8. The framework is based on a subset of the TRIO language whose formulas comply with the notion of sampling invariance. The following Chapter 9 analyzes some features of the integration framework introduced beforehand. In particular, it discusses the expressiveness of the subset of the TRIO language introduced for integration, clarifies the characterization of behaviors that has been introduced in order to achieve integration, and compares the notion of sampling invariance with other notions of discretization that have been introduced in the literature. Finally, Chapter E reviews literature related to our work about integration. In particular, it focuses on works about the expressiveness of formalisms (both logic and automata) close to the subset of TRIO we define in the Chapter 8.

8. A Framework for Discrete- and Continuous-Time Integration

This chapter presents our framework for integrating discrete-time and continuous-time modules. As we discussed informally in the previous Chapter 7, the framework relies on a the notion of *sampling invariance* to link discrete-time behaviors to continuous-time ones.

Full-fledged TRIO is a very expressive language, and therefore it allows one to express complex properties that do not conform to the notion of sampling invariance. Therefore, in Section 8.1 we introduce a proper subset of the TRIO language, that we call $^{\mathbb{R}}_{\mathbb{Z}}$TRIO.[1] We fully define the semantics of $^{\mathbb{R}}_{\mathbb{Z}}$TRIO, parametrically with respect to the chosen time domain.

Then, Section 8.2 defines formally the concept of sampling invariance of a TRIO formula (Definition 8.2.1). Section 8.3 completes the presentation of the integration framework by presenting sufficient conditions under which $^{\mathbb{R}}_{\mathbb{Z}}$TRIO formulas achieve the sampling invariance property. Finally, Section 8.4 presents an example of integration for a simple controlled reservoir system made of two modules.

8.1. $^{\mathbb{R}}_{\mathbb{Z}}$TRIO: A Simple Metric Temporal Logic

For the purposes of integration, we consider a proper subset of the TRIO language, which is strictly less expressive and which we call $^{\mathbb{R}}_{\mathbb{Z}}$TRIO.

8.1.1. Syntax

As we illustrated in Section 2.1, TRIO is based on a single modal operator named Dist. $^{\mathbb{R}}_{\mathbb{Z}}$TRIO is instead based on the two primitive temporal operators Until and Since, as well as the usual propositional connectives. For simplicity, we also introduce $^{\mathbb{R}}_{\mathbb{Z}}$TRIO as a purely propositional language.

Let us define formally the syntax of $^{\mathbb{R}}_{\mathbb{Z}}$TRIO. Let Ξ be a set of time-dependent *conditions*. These are basically Boolean expressions obtained

[1] You can read it as "ar-zee-TRIO".

by functional combination of basic time-dependent items with constants. We are going to define them precisely later on (in Section 8.3), for now let us just assume that they are time-dependent formulas whose truth value is defined at any given time. Let us consider a set $\mathbb{S}$ of constants symbols. We denote *intervals* by expressions of the form $\langle l, u \rangle$, with l, u constants from $\mathbb{S}$, $\langle$ a left parenthesis from $\{(, [\}$, and $\rangle$ a right parenthesis from $\{),]\}$; let $\mathcal{I}$ be the set of all such intervals. Then, if $\xi, \xi_1, \xi_2 \in \Xi$, $I \in \mathcal{I}$, $\langle \in \{(, [\}$, and $\rangle \in \{),]\}$, well-formed formulas ϕ are defined recursively as follows.

$$\phi \quad ::= \quad \xi \mid \text{Until}_{I\rangle}(\phi_1, \phi_2) \mid \text{Since}_{I\langle}(\phi_1, \phi_2) \mid \neg\phi \mid \phi_1 \wedge \phi_2$$

From these basic operators, let us define a number of *derived* operators: in Table 8.1 we define the most common ones.[2] As it will be clear after providing a semantic definition of $^{\mathbb{R}}_{\mathbb{Z}}$TRIO formulas, derived $^{\mathbb{R}}_{\mathbb{Z}}$TRIO operators share the same semantics as the TRIO ones, the only difference being the primitive operators from which they are defined.

8.1.2. Semantics

In defining $^{\mathbb{R}}_{\mathbb{Z}}$TRIO semantics we assume that constants symbols in $\mathbb{S}$ are interpreted naturally as numbers from the time domain $\mathbb{T}$ plus the symbols $\pm\infty$, which are treated as usual. Correspondingly, intervals $\mathcal{I}$ are interpreted as intervals of $\mathbb{T}$ which are closed/open to the left/right (as usual square brackets denote an included endpoint, and round brackets denote an excluded one).

Then, we define the semantics of a $^{\mathbb{R}}_{\mathbb{Z}}$TRIO formula using as interpretations mappings from the time domain $\mathbb{T}$ to the domain D the basic items map their values to. Let $\mathcal{B}_{\mathbb{T}}$ be the set of all such mappings, which we call *behaviors*, and let $b \in \mathcal{B}_{\mathbb{T}}$ be any element from that set. If we denote by $\xi|_{b(t)}$ the truth value of the condition ξ at time $t \in \mathbb{T}$ according to behavior b, we can define the semantics of $^{\mathbb{R}}_{\mathbb{Z}}$TRIO formulas as follows. We write $b \models_{\mathbb{T}} \phi$ to indicate that the behavior b is a model for formula ϕ under the time model $\mathbb{T}$. Thus, let us define the semantics for the generic time model

[2] Implication $\Rightarrow$, disjunction $\vee$ and double implication $\Leftrightarrow$ are defined as usual. Moreover δ is a positive real constant that will be defined shortly. Finally, for simplicity we assume that $I = \langle l, u \rangle$ is such that $l, u \geq 0$ or $l, u \leq 0$ in the definition of the $\exists t \in I$ and $\forall t \in I$ operators.

$\mathbb{T}$.

$b(t) \models_{\mathbb{T}} \xi$	iff	$\xi\vert_{b(t)}$
$b(t) \models_{\mathbb{T}} \text{Until}_{I\rangle}(\phi_1, \phi_2)$	iff	there exists $d \in I$ such that $b(t+d) \models_{\mathbb{T}} \phi_2$ and, for all $u \in [0, d\rangle$ it is $b(t+u) \models_{\mathbb{T}} \phi_1$
$b(t) \models_{\mathbb{T}} \text{Since}_{I\langle}(\phi_1, \phi_2)$	iff	there exists $d \in I$ such that $b(t-d) \models_{\mathbb{T}} \phi_2$ and, for all $u \in \langle -d, 0]$ it is $b(t+u) \models_{\mathbb{T}} \phi_1$
$b(t) \models_{\mathbb{T}} \neg\phi$	iff	$b(t) \not\models_{\mathbb{T}} \phi$
$b(t) \models_{\mathbb{T}} \phi_1 \wedge \phi_2$	iff	$b(t) \models_{\mathbb{T}} \phi_1$ and $b(t) \models_{\mathbb{T}} \phi_2$
$b \models_{\mathbb{T}} \phi$	iff	for all $t \in \mathbb{T}$: $b(t) \models_{\mathbb{T}} \phi$

Thus, a $^{\mathbb{R}}_{\mathbb{Z}}$TRIO formula ϕ constitutes the specification of a system, representing exactly all behaviors that are models of the formula. We denote by $[\![\phi]\!]_{\mathbb{T}}$ the set of all models of formula ϕ with time domain $\mathbb{T}$, i.e., $[\![\phi]\!]_{\mathbb{T}} \equiv \{b \in \mathcal{B}_{\mathbb{T}} \mid b \models_{\mathbb{T}} \phi\}$.

8.2. Sampled Behaviors and Sampling Invariance

We want to relate the models of a formula with a continuous-time domain to those of the same formula with a discrete-time domain. To do that, we introduce the notion of *sampling of a continuous-time behavior*. Basically, given a continuous-time behavior $b \in \mathcal{B}_{\mathbb{R}}$, we define its *sampling* as the discrete-time behavior $\sigma_{\delta,z}[b] \in \mathcal{B}_{\mathbb{Z}}$ that agrees with b at all integer time instants corresponding to multiples of a constant $\delta \in \mathbb{R}_{>0}$[3] from a basic offset $z \in \mathbb{R}$. We call δ the *sampling period* and z the *origin* of the sampling. More precisely, we have the following definition:

$$\forall k \in \mathbb{Z}: \quad \sigma_{\delta,z}[b](k) \;\equiv\; b(z + k\delta)$$

8.2.1. Sampling Invariance

Now, given a $^{\mathbb{R}}_{\mathbb{Z}}$TRIO formula ϕ we want to relate the set $[\![\phi]\!]_{\mathbb{Z}}$ of its discrete-time models to the set of behaviors obtained by sampling its continuous-time models $[\![\phi]\!]_{\mathbb{R}}$. Ideally, we would like that, for some sampling period δ and origin z, the following holds.

[3] We use the symbol $\mathbb{R}_{>c}$, for some $c \in \mathbb{R}$, to denote the set $\{x \in \mathbb{R} \mid x > c\}$, and similar ones for integer numbers.

- For any continuous-time behavior $b \in \mathcal{B}_{\mathbb{R}}$ which is a model for ϕ, the sampling of b is a model for ϕ in discrete time, that is:

$$b \in [\![\phi]\!]_{\mathbb{R}} \quad \Rightarrow \quad \sigma_{\delta,z}[b] \in [\![\phi]\!]_{\mathbb{Z}}$$

  When this holds, we say that ϕ is *closed under sampling*.

- For any discrete-time behavior $b \in \mathcal{B}_{\mathbb{Z}}$ which is a model for ϕ, any continuous-time behavior, whose sampling coincides with b, is a model for ϕ in continuous time, that is:

$$b \in [\![\phi]\!]_{\mathbb{Z}} \quad \Rightarrow \quad \forall b' : \left(\sigma_{\delta,z}\left[b'\right] = b \;\Rightarrow\; b' \in [\![\phi]\!]_{\mathbb{R}}\right)$$

  When this holds, we say that ϕ is *closed under inverse sampling*.

When all the formulas of a language are closed both under sampling and under inverse sampling we say that the language is *sampling invariant*. We briefly note that it is possible to explore variations of the above definitions, for example by allowing to pick a sampling period and/or an origin which depend on the particular behavior being considered. We leave the discussion of the impact of these variations to future work.

8.2.2. On Continuous versus Discrete Semantics

Actually, the above requirement is a too demanding one, as there are a number of facts indicating that it is not possible, in general, to achieve it, unless we adopt a logic language with such a limited expressiveness that it is of no practical use. In this subsection we briefly hint at some of these facts: notice that the exposition is deliberately a bit loose, as we only want to motivate the developments that will follow.

Cardinality. The first fact is a simple cardinality argument, that reminds us that the mappings from $\mathbb{R}$ are "many more" than the mappings from $\mathbb{Z}$. In fact, for any domain D of size $d = |D|$, the cardinality of the set of all mappings $\mathbb{Z} \to D$ is $d^{\aleph_0}$, whereas the set of all mappings $\mathbb{R} \to D$ has the much larger cardinality $d^{\mathcal{C}} = d^{2^{\aleph_0}}$.

Time Units and Sampling Period. Another issue that needs to be dealt with is the problem of time units. Let us illustrate it with the $\frac{\mathbb{R}}{\mathbb{Z}}$TRIO formula $\mathrm{Lasts}(A, 1/2)$, where A is a primitive condition. In a discrete-time

setting, we would like the formula to mean: for all (integer) time distances that fall in an interval between the two time instants corresponding to the sampling instants closest to 0 and 1/2, etc. Hence, we should actually adopt the formula $\text{Lasts}(A, 1/2\delta)$ as discrete-time counterpart, dividing every *time constant* by the sampling period.

Conversely, there is another change in the time constants — probably less manifest — that must be introduced when interpreting a discrete-time formula in continuous time. To give the intuition, consider the formula $\text{Lasts}_{\text{ii}}(A, 1)$ in discrete time: A holds for two consecutive time steps (including the current one). In continuous time, when evaluating the corresponding $\text{Lasts}_{\text{ii}}(A, \delta)$ between two consecutive sampling instants (cf. instant t in Figure 8.1(b)), we can only state a *weaker* property about A holding, namely that A holds in a *subset* of the δ-wide interval in the future. On basic formulas, this requirement is rendered as a *shrinking* (or *stretching*, for existential formulas) of the intervals bounds by one discrete time unit.

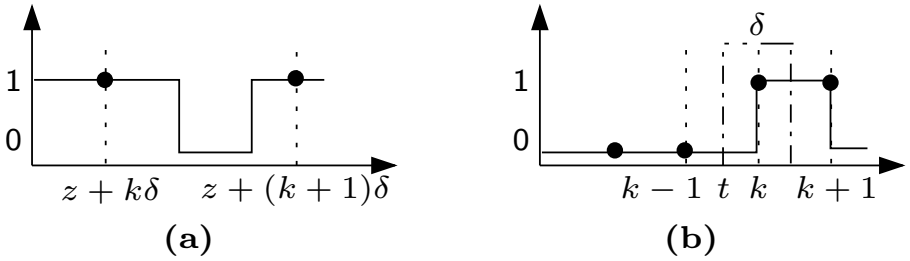

Figure 8.1.: (a) Change detection failure; (b) Moving interval.

In order to do this, in Section 8.2.3 we are going to define how to modify the time constants in any $\frac{\mathbb{R}}{\mathbb{Z}}$TRIO formula when passing from continuous to discrete time and vice versa. Let us name *adaptation function* this simple translation rule, as it just adapts the time bounds in our formulas, without changing its structure. We denote the *adaptation* function from continuous to discrete time as $\eta_\delta^{\mathbb{R}}\{\cdot\}$, and the converse function from discrete to continuous time as $\eta_\delta^{\mathbb{Z}}\{\cdot\}$.

Items Velocity and Change Detection. A basic intuition that should be developed about the above idea of sampling is that, whenever the values of the basic items of our specification — whose evolution over time is described by behaviors — change "too fast" with respect to the chosen sampling

period δ, it is impossible to guarantee that those changes are "detected" at sampling instants, so that the properties of the behavior are preserved in the discrete-time setting.

Consider, for instance, the example of Figure 8.1(a). There, a Boolean-valued item changes its values twice (from true to false and then back to true) within the interval, of length δ, between two adjacent sampling instants. Therefore, any continuous-time formula predicating about the value of the item within those instants may be true in continuous time and false for the sampling of the behavior, which only "sees" two consecutive true values. Instead, we would like that the "rate of change" of the items values is slow-paced enough that every change in the value of an item is detected at some sampling instant, before it changes again.

To this end, let us introduce a *constraint* on the continuous-time behaviors of a formula: only behaviors conforming to the constraint can be invariant under sampling. The precise form of the constraint to be introduced depends on the kind of items we are dealing with in our specification (namely, whether they take values to discrete or continuous domains); we will define it precisely in the next Section 8.2.3. In practice, however, the constraint is expressed by an additional $^{\mathbb{R}}_{\mathbb{Z}}$TRIO formula χ, which depends, in general, on the particular specification we have written; we call χ the *behavior constraint*.

8.2.3. Constrained Behaviors and Adaptation

In order to define a subset of $^{\mathbb{R}}_{\mathbb{Z}}$TRIO which is sampling invariant, it is convenient to assume that every formula is written in a suitable normal form. We are going to show that every $^{\mathbb{R}}_{\mathbb{Z}}$TRIO formula can be put in normal form, provided we can introduce auxiliary basic items. How this impacts the expressiveness of the language will be discussed in Section 9.2.3.

Normal Form. $^{\mathbb{R}}_{\mathbb{Z}}$TRIO formulas in *normal form* are written according to the following grammar, where $\xi, \xi_1, \xi_2 \in \Xi$ are primitive conditions (defined formally in Section 8.3).

$$\begin{aligned} \phi \quad ::= \quad & \xi \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 \mid \mathrm{Until}_{I\rangle}(\xi_1, \xi_2) \mid \mathrm{Since}_{I\langle}(\xi_1, \xi_2) \\ & \mathrm{Releases}_{I\rangle}(\xi_1, \xi_2) \mid \mathrm{Released}_{I\langle}(\xi_1, \xi_2) \end{aligned}$$

Notice that the normal form prohibits nesting of temporal operators and negations of formulas. Moreover, we have an additional syntactic restriction which applies to discrete-time formulas only (that is, to formulas that are meant to be interpreted in discrete time): we require that the Until and Since operators always use an included boundary, while the Releases and Released operators always use an excluded boundary. In other words, we consider only discrete-time formulas using the temporal operators $\text{Until}_{I]}(\cdot)$, $\text{Since}_{I[}(\cdot)$, $\text{Releases}_{I)}(\cdot)$, and $\text{Released}_{I(}(\cdot)$.

Despite the above restrictions, any $^{\mathbb{R}}_{\mathbb{Z}}$TRIO formula can be expressed with this $^{\mathbb{R}}_{\mathbb{Z}}$TRIO language fragment, with the additional requirement that we are allowed to introduce new auxiliary basic items in the specification, as we are going to show shortly. First, for the ease of exposition of the following proofs, let us introduce explicitly the semantics for the operators of the normal form which are not basic $^{\mathbb{R}}_{\mathbb{Z}}$TRIO operators, namely Releases, Released, and $\vee$.

$b(t) \models_{\mathbb{T}} \text{Releases}_{I\rangle}(\xi_1, \xi_2)$	iff	for all $d \in I$ it is either $b(t+d) \models_{\mathbb{T}} \xi_2$ or there exists $u \in [0, d\rangle$ such that $b(t+u) \models_{\mathbb{T}} \xi_1$
$b(t) \models_{\mathbb{T}} \text{Released}_{I\langle}(\xi_1, \xi_2)$	iff	for all $d \in I$ it is either $b(t-d) \models_{\mathbb{T}} \xi_2$ or there exists $u \in \langle -d, 0]$ such that $b(t+u) \models_{\mathbb{T}} \xi_1$
$b(t) \models_{\mathbb{T}} \phi_1 \vee \phi_2$	iff	$b(t) \models_{\mathbb{T}} \phi_1$ or $b(t) \models_{\mathbb{T}} \phi_2$

In Appendix D.1, we show how any $^{\mathbb{R}}_{\mathbb{Z}}$TRIO formula can be expressed using just the above operators, possibly introducing auxiliary items.

Semantic Constraint. The precise form of the semantic constraint to be introduced depends on the kind of items we are dealing with in our specification, and namely whether they take values to discrete or dense domains. We are going to discuss separately the two cases in Section 8.3. In a nutshell, we constrain the dynamics of the basic items and/or conditions of the specification by requiring that at least δ time units (where δ is the chosen sampling period) elapse between each pair of instants of time at which the items change their values. This restriction — which mandates a "maximum rate of change" of the items' values — assures that every change in the truth value of a formula is propagated until either the previous or the next sampling instant, before being possibly reversed by another change. Therefore, there is no information loss by observing the system just at the sampling instants, where by *information* we (informally) mean the truth

value of a specification formula (which is the only description of the system that we consider).

In practice, the semantic constraint is expressed by an additional $^{\mathbb{R}}_{\mathbb{Z}}$TRIO formula χ, which in general depends on the particular specification we have written. Thus, whenever we write a specification ϕ, we are actually considering the (more restrictive) specification $\phi \wedge \chi$, which only has models with the desired qualities. We can see the semantic constraint χ as a price that we have to pay in order to have a sampling invariant specification. Clearly, whenever the specification already entails the semantic constraint, sampling invariance comes simply at no price. In future work, we will discuss how the semantic constraint impacts on the description of real systems, as well as on the limitations it imposes on the properties that a system can have.

Adaptation Function. We define an *adaptation* function $\eta_\delta^{\mathbb{R}}\{\cdot\}$ which translates formulas by scaling the time constants, appearing as endpoints of the intervals, to their values divided by the sampling period δ. We also define its "inverse" $\eta_\delta^{\mathbb{Z}}\{\cdot\}$, which translates valid discrete-time formulas to valid continuous-time ones, by scaling back the endpoints. Notice that, in discrete time, it suffices to define the adaptation rule for closed intervals. Thus, for any formula respecting the all of the above syntactic restrictions, adaptation is defined formally inductively on the structure of the formulas of $^{\mathbb{R}}_{\mathbb{Z}}$TRIO as follows.

$$
\begin{array}{lcl}
\eta_\delta^{\mathbb{R}}\{\xi\} & \equiv & \xi \\
\eta_\delta^{\mathbb{R}}\{\mathrm{Until}_{\langle l,u\rangle}(\phi_1,\phi_2)\} & \equiv & \mathrm{Until}_{[\lfloor l/\delta\rfloor,\lceil u/\delta\rceil]}\big(\eta_\delta^{\mathbb{R}}\{\phi_1\},\eta_\delta^{\mathbb{R}}\{\phi_2\}\big) \\
\eta_\delta^{\mathbb{R}}\{\mathrm{Since}_{\langle l,u\rangle\langle}(\phi_1,\phi_2)\} & \equiv & \mathrm{Since}_{[\lfloor l/\delta\rfloor,\lceil u/\delta\rceil]\langle}\big(\eta_\delta^{\mathbb{R}}\{\phi_1\},\eta_\delta^{\mathbb{R}}\{\phi_2\}\big) \\
\eta_\delta^{\mathbb{R}}\{\mathrm{Releases}_{\langle l,u\rangle\rangle}(\phi_1,\phi_2)\} & \equiv & \mathrm{Releases}_{\langle l',u'\rangle]}\big(\eta_\delta^{\mathbb{R}}\{\phi_1\},\eta_\delta^{\mathbb{R}}\{\phi_2\}\big) \\
 & & \text{where } l' = \begin{cases} \lfloor l/\delta\rfloor & \text{if } \langle \text{ is } (\\ \lceil l/\delta\rceil & \text{if } \langle \text{ is } [\end{cases} \\
 & & \text{and } u' = \begin{cases} \lceil u/\delta\rceil & \text{if } \rangle \text{ is }) \\ \lfloor u/\delta\rfloor & \text{if } \rangle \text{ is }] \end{cases} \\
\eta_\delta^{\mathbb{R}}\{\mathrm{Released}_{\langle l,u\rangle\langle}(\phi_1,\phi_2)\} & \equiv & \mathrm{Released}_{\langle l',u'\rangle[}\big(\eta_\delta^{\mathbb{R}}\{\phi_1\},\eta_\delta^{\mathbb{R}}\{\phi_2\}\big) \\
 & & \text{where } l' = \begin{cases} \lfloor l/\delta\rfloor & \text{if } \langle \text{ is } (\\ \lceil l/\delta\rceil & \text{if } \langle \text{ is } [\end{cases} \\
 & & \text{and } u' = \begin{cases} \lceil u/\delta\rceil & \text{if } \rangle \text{ is }) \\ \lfloor u/\delta\rfloor & \text{if } \rangle \text{ is }] \end{cases} \\
\eta_\delta^{\mathbb{R}}\{\phi_1 \wedge \phi_2\} & \equiv & \eta_\delta^{\mathbb{R}}\{\phi_1\} \wedge \eta_\delta^{\mathbb{R}}\{\phi_2\} \\
\eta_\delta^{\mathbb{R}}\{\phi_1 \vee \phi_2\} & \equiv & \eta_\delta^{\mathbb{R}}\{\phi_1\} \vee \eta_\delta^{\mathbb{R}}\{\phi_2\}
\end{array}
$$

Notice that in the discrete-to-continuous adaptation of the Until and Since operators, the adapted interval $\langle (l-1)\delta, (u-1)\delta\rangle$ can indifferently be taken to include or exclude its endpoints.

$$
\begin{array}{lcl}
\eta_\delta^{\mathbb{Z}}\{\xi\} & \equiv & \xi \\
\eta_\delta^{\mathbb{Z}}\{\mathrm{Until}_{[l,u]]}(\phi_1,\phi_2)\} & \equiv & \mathrm{Until}_{\langle (l-1)\delta,(u+1)\delta\rangle]}\big(\eta_\delta^{\mathbb{Z}}\{\phi_1\},\eta_\delta^{\mathbb{Z}}\{\phi_2\}\big) \\
\eta_\delta^{\mathbb{Z}}\{\mathrm{Since}_{[l,u][}(\phi_1,\phi_2)\} & \equiv & \mathrm{Since}_{\langle (l-1)\delta,(u+1)\delta\rangle[}\big(\eta_\delta^{\mathbb{Z}}\{\phi_1\},\eta_\delta^{\mathbb{Z}}\{\phi_2\}\big) \\
\eta_\delta^{\mathbb{Z}}\{\mathrm{Releases}_{[l,u])}(\phi_1,\phi_2)\} & \equiv & \mathrm{Releases}_{[(l+1)\delta,(u-1)\delta])}\big(\eta_\delta^{\mathbb{Z}}\{\phi_1\},\eta_\delta^{\mathbb{Z}}\{\phi_2\}\big) \\
\eta_\delta^{\mathbb{Z}}\{\mathrm{Released}_{[l,u]\langle}(\phi_1,\phi_2)\} & \equiv & \mathrm{Released}_{[(l+1)\delta,(u-1)\delta]\langle}\big(\eta_\delta^{\mathbb{Z}}\{\phi_1\},\eta_\delta^{\mathbb{Z}}\{\phi_2\}\big) \\
\eta_\delta^{\mathbb{Z}}\{\phi_1 \wedge \phi_2\} & \equiv & \eta_\delta^{\mathbb{Z}}\{\phi_1\} \wedge \eta_\delta^{\mathbb{Z}}\{\phi_2\} \\
\eta_\delta^{\mathbb{Z}}\{\phi_1 \vee \phi_2\} & \equiv & \eta_\delta^{\mathbb{Z}}\{\phi_1\} \vee \eta_\delta^{\mathbb{Z}}\{\phi_2\}
\end{array}
$$

Sampling Invariance: A Definition

Finally, according to the above ideas, we can formulate a definition of sampling invariance which is the one we are actually going to use in the remainder.

Definition 8.2.1 (Sampling Invariance)**.** Given a formula ϕ, a behavior constraint formula χ, two adaptation functions $\eta_\delta^{\mathbb{R}}\{\cdot\}$ and $\eta_\delta^{\mathbb{Z}}\{\cdot\}$, a sampling period δ, and an origin z, we say that:

- ϕ is *closed under sampling* iff for any continuous-time behavior $b \in \mathcal{B}_{\mathbb{R}}$:
$$b \in [\![\phi \wedge \chi]\!]_{\mathbb{R}} \quad \Rightarrow \quad \sigma_{\delta,z}[b] \in [\![\eta_{\delta}^{\mathbb{R}}\{\phi\}]\!]_{\mathbb{Z}}$$
- ϕ is *closed under inverse sampling* iff for any discrete-time behavior $b \in \mathcal{B}_{\mathbb{Z}}$:
$$b \in [\![\phi]\!]_{\mathbb{Z}} \quad \Rightarrow \quad \forall b' \in [\![\chi]\!]_{\mathbb{R}} : \left(\sigma_{\delta,z}\left[b'\right] = b \Rightarrow b' \in [\![\eta_{\delta}^{\mathbb{Z}}\{\phi\} \wedge \chi]\!]_{\mathbb{R}}\right)$$
- A language is *sampling invariant* iff all the formulas of the language are closed under sampling (when interpreted in the continuous-time domain) and are closed under inverse sampling (when interpreted in the discrete-time domain).

8.3. Sampling Invariant Specifications

In this section we formulate a sufficient condition for the sampling invariance of a $^{\mathbb{R}}_{\mathbb{Z}}$TRIO specification formula ϕ. This condition involves the definition of a suitable constraint formula χ on the continuous-time behaviors, as well as a definition of the forms of the conditions ξ that can appear in $^{\mathbb{R}}_{\mathbb{Z}}$TRIO formulas. More precisely, we distinguish two cases, whether we are dealing with time-dependent items which take values onto a discrete set, or with time-dependent items which take values onto a dense set.

Let us introduce this idea with more precision. We assume that every specification is built out of primitive time-dependent items from a (finite) set Ψ. For example one such item may represent — by a Boolean value — the state of a light bulb (on or off), another one may instead represent — by a real value — the measure of the temperature in a room, etc. So, every time-dependent item ψ_i in Ψ is a mapping from time $\mathbb{T}$ to a suitable domain D_i, for $i = 1, \ldots, n$, where n is the number of primitive time-dependent items. Therefore, every behavior $b \in \mathcal{B}_{\mathbb{T}}$ represents the evolution over time of the values of *all* primitive items; in other words, every behavior $b \in \mathcal{B}_{\mathbb{T}}$ is a mapping from time $\mathbb{T}$ to the domain $D \equiv D_1 \times \cdots \times D_n$. Note that we do not consider the definition of more complicated time-dependent items, such as time-dependent functions, which are part of standard TRIO. These would not be difficult to introduce, but we leave them out of the present work, in favor of a simpler exposition. The next two subsections will draw sufficient conditions for sampling invariance in the two cases of when every D_i is a discrete set (Section 8.3.1) and when some D_i is a dense set (Section 8.3.2).

8.3.1. Discrete-valued Items

Let us consider the case in which all time-dependent items in Ψ have discrete co-domains, i.e., ψ_i is a time-dependent item taking values to the discrete set D_i, for all $i = 1, \ldots, n$. Thus, behaviors are mappings $\mathbb{T} \rightarrow D$ where $D \equiv D_1 \times \cdots \times D_n$. We also assume that a total order is defined on each D_i. Although this assumption could be avoided, we introduce it for uniformity of presentation; still, one can easily extend the present exposition to unordered sets as well.

Conditions. Let Λ be a set of constants from the sets D_i's, and Γ be a set of functions[4] having domains in subsets of D and co-domains in some D_i's. Then, if $\lambda \in \Lambda$, $f, f_1, f_2 \in \Gamma$, and $\bowtie \in \{=, <, \leq, >, \geq\}$, conditions ξ can be defined recursively as follows, assuming type compatibility is respected:

$$\xi \ ::= \ f(\psi_1, \ldots, \psi_n) \bowtie \lambda \mid f_1(\psi_1, \ldots, \psi_n) \bowtie f_2(\psi_1, \ldots, \psi_n) \mid \neg\xi \mid \xi_1 \wedge \xi_2$$

Abbreviations and notational conventions are introduced in the obvious ways.

Constraint on Behaviors. The constraint on behaviors is expressed by Formula $\chi_\circ$ and basically requires that the values of the items in Ψ vary over time in such a way that changes happen at most every δ time units, i.e., the value of each item is held for δ, at least. This can be expressed formally with the following $^{\mathbb{R}}_{\mathbb{Z}}$TRIO[5] formula.[6]

$$\chi_\circ \triangleq \forall v \in D : \ (\langle \psi_1, \ldots, \psi_n \rangle = v \\ \Rightarrow \ \mathrm{WithinP_{ii}}(\mathrm{Lasts_{ii}}(\langle \psi_1, \ldots, \psi_n \rangle = v, \delta), \delta))$$

A Sufficient Condition for Sampling Invariance. We finally prove a sufficient condition for sampling invariance of $^{\mathbb{R}}_{\mathbb{Z}}$TRIO formulas in the following.

[4] These are of course time-independent functions, in TRIO terms.

[5] Actually, $^{\mathbb{R}}_{\mathbb{Z}}$TRIO as we defined it above is purely propositional, while $\chi_\circ$ uses a universal quantification on non-temporal variables. However, later it will be shown why this modification is a natural and acceptable generalization which is orthogonal to sampling invariance.

[6] $\mathrm{WithinP_{ii}}(\phi, \tau) \triangleq \exists u \in [-\tau, 0] : \mathrm{Dist}(\phi, u)$, and $\mathrm{Lasts_{ii}}(\phi, \tau) \triangleq \forall u \in [0, \tau] : \mathrm{Dist}(\phi, u)$.

Theorem 8.3.1 (Sampling Invariance for Discrete-valued Items)**.** *Normal-form* $\frac{\mathbb{R}}{\mathbb{Z}}$*TRIO is sampling invariant, for items mapping to a* discrete *set* D*, with respect to the behavior constraint* $\chi_\circ$*, the adaptation functions* $\eta_\delta^{\mathbb{R}}\{\cdot\}$ *and* $\eta_\delta^{\mathbb{Z}}\{\cdot\}$*, for any sampling period* δ *and origin* z*.*

Proof sketch. Let us sketch the outline of the proof and refer to Appendix D.2 for all the details.

Let ϕ be any $\frac{\mathbb{R}}{\mathbb{Z}}$TRIO formula: the proof is split into two main parts: first we show that any $\frac{\mathbb{R}}{\mathbb{Z}}$TRIO formula ϕ, interpreted in the continuous-time domain, is closed under sampling; then we show that any formula ϕ, interpreted in the discrete-time domain, is closed under inverse sampling.

To prove *closure under sampling*, let b be a continuous-time behavior in $[\![\chi_\circ]\!]_{\mathbb{R}}$, $\phi' = \eta_\delta^{\mathbb{R}}\{\phi\}$, and b' be the sampling $\sigma_{\delta,z}[b]$ of behavior b with the given origin and sampling period. For a generic sampling instant $t = z + k\delta$, one can show that $b(t) \models_{\mathbb{R}} \phi$ implies $b'(k) \models_{\mathbb{Z}} \phi'$, by induction on the structure of ϕ. Even if all the details are quite convoluted, the overall idea is fairly terse: assuming ϕ holds we infer that some condition ξ holds at some instant u that depends on the bounds of the time intervals involved in ϕ; then, condition $\chi_\circ$ (or $\chi_\bullet^\phi$) ensures that the truth of ξ is held at least until the previous or the next sampling instant (w.r.t. u); therefore, the discrete-time formula ϕ', whose time bounds have been relaxed by the adaptation function $\eta_\delta^{\mathbb{R}}\{\cdot\}$, matches this sampling instant and is thus shown to hold at k.

To prove *closure under inverse sampling* let b be a discrete-time behavior, $\phi' = \eta_\delta^{\mathbb{Z}}\{\phi\}$, and b' be a continuous-time behavior such that $b' \in [\![\chi_\circ]\!]_{\mathbb{R}}$ and $b = \sigma_{\delta,z}[b']$ for the given origin and sampling period. The proof is now split into two parts. The former shows that, for a generic sampling instant $t = z + k\delta$, if $b(k) \models_{\mathbb{Z}} \phi$ then $b'(t) \models_{\mathbb{R}} \phi'$, that is ϕ' holds in continuous-time at all sampling instants. Again, this involves reasoning on the intervals involved in the operators and the condition $\chi_\circ$ (or $\chi_\bullet^\phi$). Afterward, we show that ϕ' holds in between sampling instants: if it holds at some sampling instant t, then it is either always true in the future and past, or there exist definite "changing points" in the future and past where ϕ' changes its truth value to false. These changing points, however, can be shown to correspond to changes in the truth value of some condition ξ (because in normal form we have no nesting), and thus there can be at most one of them between any two consecutive sampling instants. A final analysis shows that indeed if any ϕ' holds at *all* sampling instants, then it also holds in *between* them, since it should otherwise change its value twice between two of them. □

8.3.2. Dense-valued Items

Let us now consider the case in which some time-dependent items in Ψ have dense co-domains, i.e., ψ_i has a dense co-domain D_i, for some $i = 1, \ldots, n$. Without loss of generality, we can even assume that *all* D_i's are dense sets, since we can handle separately the discrete sets as seen in the previous subsection (we do not discuss the details of that as it is straightforward). Similarly as for discrete-valued items, we assume that each D_i is a totally ordered set, with no practical loss of generality.

Conditions. Let Λ be a set of n-tuples of the form $\langle \langle^1 l_1, u_1\rangle^1, \ldots, \langle^n l_n, u_n\rangle^n \rangle$, with $l_i, u_i \in D_i$, $l_i \leq u_i$, $\langle^i \in \{(, [\}$, and $\rangle^i \in \{),]\}$, for all $i = 1, \ldots, n$. Moreover, for any $\lambda \in \Lambda$, we denote by $\lambda|_i$ its projection to the ith component, i.e., $\lambda|_i \equiv \langle^i l_i, u_i\rangle^i$. Then, if $\lambda \in \Lambda$, conditions ξ are defined simply as follows:

$$\xi \ ::= \ \langle \psi_1, \ldots, \psi_n \rangle \in \lambda \mid \neg\xi \mid \xi_1 \wedge \xi_2$$

For a behavior $b : \mathbb{T} \to D$, we define $\langle \psi_1, \ldots, \psi_n \rangle|_{b(t)} \equiv \forall i = 1, \ldots, n : \psi_i(t) \in \langle^i l_i, u_i\rangle^i$, i.e., all items are in their respective ranges at time t. Also in this case, it is simple to introduce abbreviations and notational conventions.

Constraint on Behaviors. The constraint on behaviors is now dependent on the actual conditions ξ that we have introduced in our specification formula ϕ. This is different than the constraint $\chi_\circ$ for discrete-valued items, which was independent of ϕ. Now, if we let $\widetilde{\Xi}$ be the set of conditions that appear in ϕ, we have to require that the value of every item in Ψ is such that changes with respect to the conditions in $\widetilde{\Xi}$ happen at most every δ time units. In other words, if any condition $\widetilde{\xi} \in \widetilde{\Xi}$ is true (resp. false) at some time, then it stays true (resp. false) in an interval of length δ (at least). This is expressed by the following "pseudo"-$^{\mathbb{R}}_{\mathbb{Z}}$TRIO formula, where the higher-order quantification $\forall \widetilde{\xi} \in \widetilde{\Xi}$ is to be meant as a shorthand for the explicit enumeration of all the conditions in (the finite set) $\widetilde{\Xi}$.

$$\chi^{\phi}_{\bullet} \triangleq \forall \widetilde{\xi} \in \widetilde{\Xi} : \ \mathrm{WithinP_{ii}}\Big(\mathrm{Lasts_{ii}}\Big(\widetilde{\xi}, \delta\Big), \delta\Big) \vee \mathrm{WithinP_{ii}}\Big(\mathrm{Lasts_{ii}}\Big(\neg\widetilde{\xi}, \delta\Big), \delta\Big)$$

Theorem 8.3.2 (Sampling Invariance for Dense-valued Items)**.** *Normal-form* $^{\mathbb{R}}_{\mathbb{Z}}$*TRIO is sampling invariant, for items mapping to a dense set D, with respect to the behavior constraint $\chi^{\phi}_{\bullet}$, the adaptation functions $\eta^{\mathbb{R}}_{\delta}\{\cdot\}$ and $\eta^{\mathbb{Z}}_{\delta}\{\cdot\}$, for any sampling period δ and origin z.*

Proof. Similarly as in Theorem 8.3.1, for dense-valued items the truth value of any condition ξ cannot change between any two consecutive sampling instants, exactly because of the behavior constraint $\chi_{\bullet}^{\phi}$. In fact, $\chi_{\bullet}^{\phi}$ requires that if the value of any item ψ_i is in (resp. out of) an interval $\lambda|_i$, it stays in (resp. out of) $\lambda|_i$ until the next sampling instant. Therefore, thanks to this observation, the proof is exactly as in Theorem 8.3.1 (see Appendix D.2 for details). □

8.4. An Example of Integration: The Controlled Reservoir

Let us now discuss an example that demonstrates our approach to integration. Let us consider a simple control system made of two modules: a reservoir and a controller. We model the former using continuous-time TRIO, and the latter using discrete-time TRIO. This mirrors the assumptions that the reservoir models a physical systems, where quantities vary continuously, while the controller is a digital device, that updates its state at periodic instants.

8.4.1. System Specification

The reservoir can nondeterministically leak; this is modeled by the Boolean item L which is true iff the reservoir leaks. Leaking can happen at any time. The reservoir can also be filled with new liquid: this happens whenever the Boolean item F is true. It is the controller which is responsible for setting F to true, thus replacing the fluid that has been split, whenever needed. Finally, the reservoir has a (nonnegative) real-valued time-dependent item l that represents the measure of the level of fluid in the reservoir at any instant. In our example, we model the Boolean items F and L as TRIO *states* (see Section 2.1.4).

When the reservoir is leaking, the level of fluid decreases at a constant rate of $\mathrm{r_l}$ units of fluid per unit of time. Conversely, when it is being filled, the level of fluid increases at a constant rate of $\mathrm{r_f}$, and we assume that $\mathrm{r_f} > \mathrm{r_l}$, i.e., the filling can contrast the leaking. The four possible resulting behaviors are described by the following TRIO axioms.

Axiom 35 (reservoir$^{\mathbb{R}}$.level_behavior_1).

$$\mathrm{Lasts_{ee}}(\mathsf{F} \wedge \mathsf{L}, t) \wedge \mathsf{l} = l \;\Rightarrow\; \mathrm{Futr}(\mathsf{l} = l + (\mathrm{r_f} - \mathrm{r_l})t, t)$$

Axiom 36 (**reservoir$^{\mathbb{R}}$.level_behavior_2**).

$$\mathrm{Lasts}_{\mathrm{ee}}(\mathsf{F} \wedge \neg \mathsf{L}, t) \wedge \mathsf{l} = l \;\Rightarrow\; \mathrm{Futr}(\mathsf{l} = l + \mathrm{r_f} t, t)$$

Axiom 37 (**reservoir$^{\mathbb{R}}$.level_behavior_3**).

$$\mathrm{Lasts}_{\mathrm{ee}}(\neg \mathsf{F} \wedge \mathsf{L}, t) \wedge \mathsf{l} = l \;\Rightarrow\; \mathrm{Futr}(\mathsf{l} = \max(l - \mathrm{r_l} t, 0), t)$$

Axiom 38 (**reservoir$^{\mathbb{R}}$.level_behavior_4**).

$$\mathrm{Lasted}_{\mathrm{ee}}(\neg \mathsf{F} \wedge \neg \mathsf{L}, t) \wedge \mathsf{l} = l \;\Rightarrow\; \mathrm{Futr}(\mathsf{l} = l, t)$$

For specifying the control action of the controller we define two constant values $\mathrm{l}, \mathrm{t} \in \mathbb{R}_{\geq 0}$ that represent, respectively, the minimum desired level of fluid in the reservoir and a threshold below which the control action is triggered. More precisely, t is such that $\mathrm{t} > \mathrm{l} + \mathrm{r_l}\delta$, i.e., a leaking cannot empty the difference $\mathrm{t} - \mathrm{l}$ in less than δ time units, for some $\delta \in \mathbb{R}_0^+$. Thus, we formalize very simply control action as follows (recall that this is interpreted over discrete time):

Axiom 39 (**controller$^{\mathbb{Z}}$.filling_behavior**).

$$\mathsf{l} < \mathrm{t} \;\Rightarrow\; \mathsf{F}$$

We also assume that the reservoir is initially filled properly.

Axiom 40 (**reservoir$^{\mathbb{R}}$.initialization**).

$$\mathrm{Som}(\mathrm{AlwP_e}(\mathsf{l} \geq \mathrm{t}))$$

8.4.2. Sampling Invariant Derived Specification

Axioms 35–36 above use free variables, so they are not $^{\mathbb{R}}_{\mathbb{Z}}$TRIO formulas. However, it is straightforward to derive some simpler formulas, using only constant values, that are full $^{\mathbb{R}}_{\mathbb{Z}}$TRIO formulas and can therefore be shown to be sampling invariant. They are listed below as four theorems, whose elementary proofs are shown in Appendix C for reference.

Theorem 41 (**reservoir$^{\mathbb{R}}$.abstract_behavior_1**).

$$\mathrm{Lasts}_{\mathrm{ee}}(\mathsf{F}, \delta) \wedge \mathsf{l} \geq \mathrm{l} \;\Rightarrow\; \mathrm{Futr}(\mathsf{l} \geq \mathrm{l}, \delta)$$

Theorem 42 (**reservoir$^{\mathbb{R}}$.abstract_behavior_2**).

$$\mathsf{l} \geq \mathrm{t} \;\Rightarrow\; \mathrm{Futr}(\mathsf{l} \geq \mathsf{l}, \delta)$$

Theorem 43 (**reservoir$^{\mathbb{R}}$.abstract_behavior_3**).

$$\mathrm{Lasts}_{\mathrm{ee}}(\neg\mathsf{F} \wedge \neg\mathsf{L}, \delta) \wedge \mathsf{l} \geq \mathrm{l} \;\Rightarrow\; \mathrm{Futr}(\mathsf{l} \geq \mathrm{l}, \delta)$$

Notice, instead, that the other axioms already qualify as $^{\mathbb{R}}_{\mathbb{Z}}$TRIO formulas.

8.4.3. System Requirement

The goal of the verification problem for this example is to prove the following $^{\mathbb{R}}_{\mathbb{Z}}$TRIO formula, which formalizes the requirement that the level of the reservoir never goes below the value l. Note that this formula can be equivalently interpreted over the reals or the integers.

Theorem 44 (**level_invariance**).

$$\mathrm{Alw}(\mathsf{l} \geq \mathrm{l})$$

8.4.4. Adapted Specification

Let us now put all the relevant formulas in normal form; note that we omit the parenthesis index in operators (such as the $\rangle$ in $\mathrm{Until}_{I\rangle}(\cdots)$), whenever it can be anything. For brevity, let us not describe the full transcription process.

Theorem 45 (**reservoir$^{\mathbb{R}}$.abstract_behavior_1-normal**).

$$\mathsf{l} < \mathrm{l} \;\vee\; \mathrm{Until}_{(0,\delta)}(\mathrm{true}, \neg\mathsf{F}) \;\vee\; \mathrm{Releases}_{[\delta,\delta]}(\mathrm{false}, \mathsf{l} \geq \mathrm{l})$$

Theorem 46 (**reservoir$^{\mathbb{R}}$.abstract_behavior_2-normal**).

$$\mathsf{l} < \mathrm{t} \;\vee\; \mathrm{Releases}_{[\delta,\delta]}(\mathrm{false}, \mathsf{l} \geq \mathrm{l})$$

Theorem 47 (**reservoir$^{\mathbb{R}}$.abstract_behavior_3-normal**).

$$\mathsf{l} < \mathrm{l} \;\vee\; \mathrm{Until}_{(0,\delta)}(\mathrm{true}, \mathsf{F} \vee \mathsf{L}) \;\vee\; \mathrm{Releases}_{[\delta,\delta]}(\mathrm{false}, \mathsf{l} \geq \mathrm{l})$$

In the following axiom, note that S is the auxiliary item introduced to eliminate nesting.

Axiom 48 (reservoir$^{\mathbb{R}}$.initialization-normal).

$$\big(\big(\mathsf{S} \wedge \mathrm{Released}_{(0,\infty)}(\mathrm{false}, \mathsf{I} \geq \mathrm{t})\big) \ \vee \ \big(\neg\mathsf{S} \wedge \mathrm{Since}_{(0,\infty)}(\mathrm{true}, \mathsf{I} < \mathrm{t})\big)\big) \\ \wedge \big(\mathrm{Since}_{(0,\infty)}(\mathrm{true}, \mathsf{S}) \vee \mathsf{S} \vee \mathrm{Until}_{(0,\infty)}(\mathrm{true}, \mathsf{S})\big)$$

Next, we adapt the continuous-time Theorems 45–47 and Axiom 48 into discrete-time formulas according to a sampling period δ. The resulting formulas, after putting them back into more readable form, are the following, where recall that $\mathrm{Lasts}_{\mathrm{ii}}(\phi, t) = \forall d \in [0, t] : \mathrm{Dist}(\phi, d)$.

Theorem 49 (reservoir$^{\mathbb{Z}}$.abstract_behavior_1).

$$\mathrm{Lasts}_{\mathrm{ii}}(\mathsf{F}, 1) \wedge \mathsf{I} \geq \mathrm{l} \ \Rightarrow \ \mathrm{NowOn}(\mathsf{I} \geq \mathrm{l})$$

Theorem 50 (reservoir$^{\mathbb{Z}}$.abstract_behavior_2).

$$\mathsf{I} \geq \mathrm{t} \ \Rightarrow \ \mathrm{NowOn}(\mathsf{I} \geq \mathrm{l})$$

Theorem 51 (reservoir$^{\mathbb{Z}}$.abstract_behavior_3).

$$\mathrm{Lasts}_{\mathrm{ii}}(\neg\mathsf{F} \wedge \neg\mathsf{L}, 1) \wedge \mathsf{I} \geq \mathrm{l} \ \Rightarrow \ \mathrm{NowOn}(\mathsf{I} \geq \mathrm{l})$$

Axiom 52 (reservoir$^{\mathbb{Z}}$.initialization).

$$\mathrm{Som}(\mathrm{AlwP_i}(\mathsf{I} \geq \mathrm{t}))$$

8.4.5. System Verification

Let us assume that the behavior constraint $\chi^{\phi}_{\bullet}(49-52)$ for Formulas 49–52 holds. In the following section we will discuss the actual impact of the constraint and how it can be removed.

Proof of Theorem 44 ($\mathrm{Alw}(\mathsf{I} \geq \mathrm{l})$*).* Thanks to Axiom 52, we have a time instant u at which $\mathrm{AlwP_i}(\mathsf{I} \geq \mathrm{t})$ holds. This instant serves as the base of the induction.

So, for all instants up to u the property $\mathsf{I} \geq \mathrm{l}$ holds trivially.

Let us now consider a generic time instant $v \geq u$ and assume that $\mathsf{I} \geq \mathrm{l}$ holds up to v, included. We show that $\mathsf{I} \geq \mathrm{l}$ holds at $v+1$ as well, by case discussion at v for the predicates F and L. More precisely, we have the following cases.

1. NowOn($\neg$F), that is F is false at $v+1$.
   Then, by Axiom **filling_behavior**, I $\geq$ t $>$ l at $v+1$.

2. NowOn(F), that is F is true at $v+1$. Then, we distinguish two more cases.

   a) F at v. Therefore, we have immediately I $\geq$ l at $v+1$ by Theorem 49.

   b) $\neg$F at v. Therefore, by Axiom **filling_behavior**, I $\geq$ t at v. Then, by Theorem 51, I $\geq$ l at $v+1$. □

It can be noticed that the proof is remarkably simple. In particular, thanks to the sampling invariance conditions it has been possible to abstract away from all the details on the continuous variability of the I item, while being guaranteed that nothing "unpleasant" happens between any two discrete-time instants. Thanks to the simplicity, the proof can also be completely automated. In particular, we proved it using model-checking techniques for TRIO and the SPIN model checker (see Section 2.4.1).

For a comparison, the reader may consider [FR04], where a similar example of a controlled reservoir was verified in continuous-time TRIO exploiting a compositional framework. The system description in [FR04] was less abstract than the present one, and the proof was consequently substantially more complicated even after the simplifications permitted by the compositional approach. In particular, it was not possible to completely automate the proof process, which could only be checked with the aid of PVS.

Finally, we note that the property, proved in the discrete-time setting, immediately translates through sampling invariance to the theorem **level_invariance**.

Operator	Definition
$\text{Releases}_{I\rangle}(\phi_1, \phi_2)$	$\neg\text{Until}_{I\rangle}(\neg\phi_1, \neg\phi_2)$
$\text{Released}_{I\langle}(\phi_1, \phi_2)$	$\neg\text{Since}_{I\langle}(\neg\phi_1, \neg\phi_2)$
$\exists t \in I = \langle l, u\rangle : \text{Dist}(\phi, t)$	$\begin{cases} \text{Until}_{\langle l,u\rangle\rangle}(\text{true}, \phi) & \text{if } u \geq l \geq 0 \\ \text{Since}_{\langle -l,-u\rangle\rangle}(\text{true}, \phi) & \text{if } u \leq l \leq 0 \end{cases}$
$\forall t \in I = \langle l, u\rangle : \text{Dist}(\phi, t)$	$\begin{cases} \text{Releases}_{\langle l,u\rangle\rangle}(\text{false}, \phi) & \text{if } u \geq l \geq 0 \\ \text{Released}_{\langle -l,-u\rangle\rangle}(\text{false}, \phi) & \text{if } u \leq l \leq 0 \end{cases}$
$\text{Dist}(\phi, d)$	$\forall t \in [d, d] : \text{Dist}(\phi, t)$
$\text{Futr}(\phi, d)$	$d \geq 0 \wedge \text{Dist}(\phi, d)$
$\text{Past}(\phi, d)$	$d \geq 0 \wedge \text{Dist}(\phi, -d)$
$\text{SomF}(\phi)$	$\exists t \in (0, +\infty) : \text{Dist}(\phi, t)$
$\text{SomP}(\phi)$	$\exists t \in (-\infty, 0) : \text{Dist}(\phi, t)$
$\text{Som}(\phi)$	$\text{SomF}(\phi) \vee \phi \vee \text{SomP}(\phi)$
$\text{AlwF}(\phi)$	$\forall t \in (0, +\infty) : \text{Dist}(\phi, t)$
$\text{AlwP}(\phi)$	$\forall t \in (-\infty, 0) : \text{Dist}(\phi, t)$
$\text{Alw}(\phi)$	$\text{AlwF}(\phi) \wedge \phi \wedge \text{AlwP}(\phi)$
$\text{WithinF}(\phi, \tau)$	$\exists t \in (0, \tau) : \text{Dist}(\phi, t)$
$\text{WithinP}(\phi, \tau)$	$\exists t \in (-\tau, 0) : \text{Dist}(\phi, t)$
$\text{Within}(\phi, \tau)$	$\text{WithinF}(\phi, \tau) \vee \phi \vee \text{WithinP}(\phi, \tau)$
$\text{Lasts}(\phi, \tau)$	$\forall t \in (0, \tau) : \text{Dist}(\phi, t)$
$\text{Lasted}(\phi, \tau)$	$\forall t \in (-\tau, 0) : \text{Dist}(\phi, t)$
$\text{Until}(\phi_1, \phi_2)$	$\text{Until}_{(0,+\infty))}(\phi_1, \phi_2)$
$\text{Since}(\phi_1, \phi_2)$	$\text{Since}_{(0,+\infty)(}(\phi_1, \phi_2)$
$\text{NowOn}(\phi)$	$\begin{cases} \text{Lasts}(\phi, \delta) & \text{if the time domain is } \mathbb{R} \\ \text{Futr}(\phi, 1) & \text{if the time domain is } \mathbb{Z} \end{cases}$
$\text{UpToNow}(\phi)$	$\begin{cases} \text{Lasted}(\phi, \delta) & \text{if the time domain is } \mathbb{R} \\ \text{Past}(\phi, 1) & \text{if the time domain is } \mathbb{Z} \end{cases}$
$\text{Becomes}(\phi)$	$\text{UpToNow}(\neg\phi) \wedge \text{NowOn}(\phi)$

Table 8.1.: TRIO derived temporal operators

9. Features of the Integration Framework

This chapter analyzes some important features of the integration framework presented in the previous Chapter 8.

More precisely, the following aspects are investigated.

- The adaptation function, that is part of the integration framework, prescribes to shrink intervals appearing in temporal operators; it may happen that a shrinking causes an interval to become empty. Section 9.1 shows conditions on the size of the intervals under which this does not happen.

- The expressiveness of $\frac{\mathbb{R}}{\mathbb{Z}}$TRIO is investigated in Section 9.2. In particular, we compare its expressiveness to that of the MTL temporal logic, showing that the two languages are in fact equally expressive.

- Section 9.3 studies properties of the behaviors obeying the constraint χ. Namely, we show conditions under which the "slow variability" enforced by χ can be lifted up from formulas, thus permitting to nest formulas while retaining the results about sampling invariance stated in the previous chapter for nesting-free formulas. Moreover, it is shown how to (partially) characterize the requirement of the behavior constraint $\chi^{\phi}_{\bullet}$ for dense-valued items through the mathematical notion of uniform continuity.

- In the literature, the most used notion of discretization is that based on the notion of *digitization*, introduced by Henzinger, Manna and Pnueli [HMP92]. This notion has similarities and differences with our notion of sampling invariance. Therefore, Section 9.4 formally compares the two notions.

- Although temporal logics are less expressive than automata (since the former cannot express "counting" properties), it may be useful to be able to characterize the behavior of an automaton through

a set of logic formulas. This allows one, among other things, to derive properties of the automata through logic inference, and therefore consists another approach to modeling and verification (called "dual-language" [FMMR07]). In Section 9.5 we provide an axiomatization of timed automata in $\frac{\mathbb{R}}{\mathbb{Z}}$TRIO.

9.1. How to Avoid Degenerate Intervals

In presenting the results about sampling invariance for $\frac{\mathbb{R}}{\mathbb{Z}}$TRIO, we made no assumptions on the size of the intervals involved in the formulas. In particular, it may happen that the adaptation function changes an interval so that it becomes degenerate, that is empty. Although the above proofs still hold, it may be objected that a degenerate formula makes little sense, since it is either trivially true or trivially false. In this section we present sufficient conditions on the size of the un-adapted intervals that guarantee that adaptation gives non-degenerate adapted intervals.

Table 9.1 lists the requirements on unadapted intervals that guarantee that the adapted intervals are non-empty (i.e., of size greater than 0, but allowing the endpoints to coincide for intervals of $\mathbb{R}$).

The sufficiency of these requirements is proved in Appendix D.3.

Operator (in $\mathbb{R}$)	Interval	Requirement
$\text{Until}_{I\rangle}(\cdot,\cdot)$	$I = \langle l, u\rangle$	$u - l > 0$
$\text{Since}_{I\langle}(\cdot,\cdot)$	$I = \langle l, u\rangle$	$u - l > 0$
$\text{Releases}_{I\rangle}(\cdot,\cdot)$	$I = \langle l, u\rangle \neq (l, u)$	$u - l \geq \delta$
$\text{Released}_{I\langle}(\cdot,\cdot)$	$I = \langle l, u\rangle \neq (l, u)$	$u - l \geq \delta$
$\text{Releases}_{I\rangle}(\cdot,\cdot)$	$I = (l, u)$	$u - l \geq 2\delta$
$\text{Released}_{I\langle}(\cdot,\cdot)$	$I = (l, u)$	$u - l \geq 2\delta$
Operator (in $\mathbb{Z}$)	Interval	Requirement
$\text{Until}_{I\rangle}(\cdot,\cdot)$	$I = [l, u]$	$u - l \geq 0$
$\text{Since}_{I\langle}(\cdot,\cdot)$	$I = [l, u]$	$u - l \geq 0$
$\text{Releases}_{I\rangle}(\cdot,\cdot)$	$I = [l, u]$	$u - l \geq 2$
$\text{Released}_{I\langle}(\cdot,\cdot)$	$I = [l, u]$	$u - l \geq 2$

Table 9.1.: Requirements for non-degenerate adapted intervals

9.2. On the Expressiveness of $^{\mathbb{R}}_{\mathbb{Z}}$TRIO

This section compares the expressiveness of $^{\mathbb{R}}_{\mathbb{Z}}$TRIO with those of other related formalisms, and discusses how the nesting-freeness requirement on formulas also influences the expressiveness of the resulting language.

9.2.1. Strict vs. Non-Strict Operators

$^{\mathbb{R}}_{\mathbb{Z}}$TRIO definition of the *until* and *since* operators is *non-strict* in the first argument, in that the first argument is required (in particular) to hold at the current instant as well (see Section 8.1). This choice is not very common, and it has been made necessary to achieve sampling invariance. On the contrary, the usual choice for dense-time domains is a *strict* one. This section discusses how the choice of strict and non-strict operators impacts the expressiveness of a language, by showing that non-strict metric operators are as expressive as strict ones. For brevity and ease of exposition, we moved most of the proofs to Appendix D.4, focusing here on presenting the results.

It is usually held that the expressive equivalence between strict and non-strict operators does not hold, in general, over dense time, where strict variants are believed to be strictly more expressive. This is a "folk theorem" which has never been explicitly proved — to the best of our knowledge — but whose validity is generally accepted. For instance, [BCM05] claims that the formula $\neg a \wedge \widetilde{\text{Until}}_{(0,+\infty))}(a, b)$ — where $\widetilde{\text{Until}}$ denotes a strict variation of the until operator that we are going to define shortly — cannot be expressed using only non-strict until.[1] [Hen98] claims in Footnote 5 that "in real time, strict until cannot be defined from weak until and next"[2] citing a personal communication with Raskin as a reference. [AFH96] claims that MITL's *bounded until* "cannot be defined in terms of an *until* operator that is not strict in its first argument".

However, these claim are not true for our continuous-time semantics, if we restrict ourselves to non-Zeno behaviors (as it is customary, and as it is implicit with models such as the interval-based sequence), as we are now going to show. Let us remark that we are focusing on the dense-time semantics, unless explicitly stated otherwise.

[1] Actually, it is not clear if [BCM05] refers the claim to qualitative and/or discrete-time temporal logic.

[2] Notice, however, that the reference is to a point-based semantics, while we consider interval-based models.

Strict semantics of operators. Let us define formally the semantics of strict *until* and *since*, denoted as $\widetilde{\text{Until}}$ and $\widetilde{\text{Since}}$, respectively.

$b(t) \models_{\mathbb{T}} \widetilde{\text{Until}}_{I\rangle}(\phi_1, \phi_2)$ iff there exists $d \in I$ such that $b(t+d) \models_{\mathbb{T}} \phi_2$ and, for all $u \in (0, d\rangle$ it is $b(t+u) \models_{\mathbb{T}} \phi_1$

$b(t) \models_{\mathbb{T}} \widetilde{\text{Since}}_{I\langle}(\phi_1, \phi_2)$ iff there exists $d \in I$ such that $b(t-d) \models_{\mathbb{T}} \phi_2$ and, for all $u \in \langle -d, 0)$ it is $b(t+u) \models_{\mathbb{T}} \phi_1$

We refer to the definitions in Section 8.1 for the semantics of the non-strict *until* and *since*, as well as the derived operators that we are going to use. In particular, let us remark that we always assume that the intervals I are non empty, i.e., the right end-point is at least as large as the left end-point. It is not difficult to see that this is without loss of generality.

Strict is at least as expressive as non-strict. It is quite clear that strict metric operators are at least as expressive as non-strict one. In fact, in general the following equivalences hold both over dense and over discrete time.

$$\text{Until}_{I\rangle}(\phi_1, \phi_2) \equiv \begin{cases} \phi_2 \vee (\phi_1 \wedge \widetilde{\text{Until}}_{(0,u\rangle)}(\phi_1, \phi_2)) & \text{if } I = [0, u\rangle \text{ and } \rangle \text{ is }) \\ \phi_1 \wedge \widetilde{\text{Until}}_{I\rangle}(\phi_1, \phi_2) & \text{otherwise } (0 \notin I \text{ or } \rangle =]) \end{cases}$$

The case for the *since* is immediately derivable.

Over discrete time, strict is exactly as expressive as non-strict. Over *discrete time*, it is also straightforward to notice that the converse holds, i.e., non-strict operators are at least as expressive as strict ones. In fact we have the following (recall that, in discrete time, all intervals can be expressed as closed ones).

$$\widetilde{\text{Until}}_{[l,u]\rangle}(\phi_1, \phi_2) \equiv^{\mathbb{Z}} \begin{cases} \text{NowOn}\big(\text{Until}_{[l-1,u-1]\rangle}(\phi_1, \phi_2)\big) & \text{if } l \geq 1 \\ \text{NowOn}\big(\text{Until}_{[l,u-1]\rangle}(\phi_1, \phi_2)\big) \vee \phi_2 & \text{if } l = 0 < u \\ \phi_2 & \text{if } l = u = 0 \end{cases}$$

The case for the *since* is immediately derivable.

Notice that, in the above equivalences, we used the NowOn operator, which in discrete time can be expressed as $\text{NowOn}(\phi) = \text{Futr}(\phi, 1)$, which is the same as the usual Linear Temporal Logic (LTL) *next* operator [GHR94]. Therefore, the equivalence in expressive power over discrete time does not

contradict the well-known result that *next* in LTL cannot be defined from non-strict *until* only [GHR94], as the result for LTL does not deal with the metric modality of *bounded until* — which is instead the basis of $^{\mathbb{R}}_{\mathbb{Z}}$TRIO — but with qualitative (unbounded) *until* only.

Strict and non-strict NowOn operators. In order to better comprehend the problem in the dense-time setting, let us strip it down to its core. Let us define two variations of the *nowon* and *uptonow* operators which are useful over dense time. $\epsilon\text{NowOn}(\phi)$ denotes that ϕ holds in a non-empty right-open left-closed interval on the right of the current instant; therefore $\epsilon\text{NowOn}(\phi) \equiv \text{Until}_{(0,+\infty))}(\phi, \text{true})$. Its past counterpart is $\epsilon\text{UpToNow}(\phi)$; it denotes that ϕ holds in a non-empty left-open right-closed interval on the left of the current instant, and it is defined as $\epsilon\text{UpToNow}(\phi) \equiv \text{Since}_{(0,+\infty)(}(\phi, \text{true})$. Their *strict* counterparts predicate over intervals that are open on both ends; their definitions are the following: $\epsilon\widetilde{\text{NowOn}}(\phi) \equiv \widetilde{\text{Until}}_{(0,+\infty))}(\phi, \text{true})$ and $\epsilon\widetilde{\text{UpToNow}}(\phi) \equiv \widetilde{\text{Since}}_{(0,+\infty)(}(\phi, \text{true})$.[3]

As we are going to show more explicitly shortly, the problem of expressiveness for the strict vs. non-strict *until* and *since* requires in particular to express the strict *nowon* and *uptonow* using their non-strict counterparts. In order to achieve this, we have to first recall a basic property of non-Zeno time-dependent predicates. As it is shown in [GM01], any non-Zeno primitive p item obeys the following formulas:

$$\text{Alw}\Big(\epsilon\widetilde{\text{NowOn}}(\mathsf{p}) \vee \epsilon\widetilde{\text{NowOn}}(\neg\mathsf{p})\Big)$$
$$\text{Alw}\Big(\epsilon\widetilde{\text{UpToNow}}(\mathsf{p}) \vee \epsilon\widetilde{\text{UpToNow}}(\neg\mathsf{p})\Big)$$

That is, there is always a non-empty interval in the future (and one in the past) where p has a definite constant value. It is not difficult to "lift" such property from basic items to arbitrary formulas, by showing that for any $^{\mathbb{R}}_{\mathbb{Z}}$TRIO formula ϕ, if all of the basic items on which it predicates are non-Zeno, then the truth value of ϕ as a function of time is also non-Zeno. In other words, (temporal) operators preserve non-Zenoness. The proof would go by structural induction. So we have, for any $^{\mathbb{R}}_{\mathbb{Z}}$TRIO formula ϕ,

[3] Recall that in standard TRIO the operators $\epsilon\widetilde{\text{NowOn}}$ and $\epsilon\widetilde{\text{UpToNow}}$ are denoted simply as NowOn and UpToNow [GM01]. Here, however, we adopt the notation and definitions of $^{\mathbb{R}}_{\mathbb{Z}}$TRIO (see Section 8.1).

that:

$$\mathrm{Alw}\Big(\epsilon\widetilde{\mathrm{NowOn}}(\phi) \vee \epsilon\widetilde{\mathrm{NowOn}}(\neg\phi)\Big) \tag{9.1}$$

$$\mathrm{Alw}\Big(\epsilon\widetilde{\mathrm{UpToNow}}(\phi) \vee \epsilon\widetilde{\mathrm{UpToNow}}(\neg\phi)\Big) \tag{9.2}$$

Therefore, we exploit this basic non-Zenoness property to express strict operators using non-strict ones. We have the following equivalences.

$$\epsilon\widetilde{\mathrm{NowOn}}(\phi) \;\Leftrightarrow\; \epsilon\mathrm{NowOn}(\phi) \vee (\neg\phi \wedge \neg\, \epsilon\mathrm{NowOn}(\neg\phi))$$

$$\epsilon\widetilde{\mathrm{UpToNow}}(\phi) \;\Leftrightarrow\; \epsilon\mathrm{UpToNow}(\phi) \vee (\neg\phi \wedge \neg\, \epsilon\mathrm{UpToNow}(\neg\phi))$$

The proof is simple, and relies on properties 9.1 and 9.2. Let us demonstrate the proof for the *nowon* case, the other being obviously all similar.

Proof. Let us start by proving the $\Rightarrow$ direction: assume that $\epsilon\widetilde{\mathrm{NowOn}}(\phi)$ holds at some instant t. Let us first consider the case: ϕ true at t; then also $\epsilon\mathrm{NowOn}(\phi)$, and we satisfy the first term of the disjunction. Otherwise, let us consider ϕ false at t. Then, $\epsilon\mathrm{NowOn}(\neg\phi)$ must also be false at t, otherwise $\epsilon\widetilde{\mathrm{NowOn}}(\phi)$ cannot be true. This concludes this branch, since we satisfy the second term of the disjunction.

For the $\Leftarrow$ direction, let us start by considering the case $\epsilon\mathrm{NowOn}(\phi)$ at t; then, *a fortiori*, $\epsilon\widetilde{\mathrm{NowOn}}(\phi)$, at t and we are done. Otherwise, let us assume that $\neg\phi \wedge \neg\, \epsilon\mathrm{NowOn}(\neg\phi)$ holds at the current instant t. By evaluating Formula 9.1 at time t, we immediately conclude that $\epsilon\widetilde{\mathrm{NowOn}}(\phi)$ at t, which concludes the whole proof. □

Strict until expressed with non-strict operators. Let us assume $a > 0$; then, the following equivalence hold (it is proved in Appendix D.4.1).

$$\widetilde{\mathrm{Until}}_{(a,b\rangle)}(\phi_1,\phi_2) \;\Leftrightarrow\; \mathrm{Until}_{(a,b\rangle)}(\mathrm{true},\phi_2) \wedge \mathrm{Lasts}_{\mathrm{ei}}\big(\mathrm{Until}_{(0,+\infty))}(\phi_1,\phi_2)\,,a\big) \tag{9.3}$$

Now, using Formula 9.3, it is not difficult to prove the following (still for $a > 0$, see Appendix D.4.2).

$$\widetilde{\mathrm{Until}}_{[a,b\rangle)}(\phi_1,\phi_2) \;\Leftrightarrow\; \widetilde{\mathrm{Until}}_{(a,b\rangle)}(\phi_1,\phi_2) \vee (\mathrm{Lasts}_{\mathrm{ee}}(\phi_1,a) \wedge \mathrm{Futr}(\phi_2,a)) \tag{9.4}$$

Variants with]. It is simple to express the] variants of the *until* using the) variants used above; in fact, we have the following.

$$\widetilde{\text{Until}}_{(a,b]}(\phi_1, \phi_2) \;\Leftrightarrow\; \widetilde{\text{Until}}_{(a,b)}(\phi_1, \phi_2 \wedge \phi_1) \tag{9.5}$$

$$\widetilde{\text{Until}}_{[a,b]}(\phi_1, \phi_2) \;\Leftrightarrow\; \widetilde{\text{Until}}_{[a,b)}(\phi_1, \phi_2 \wedge \phi_1) \tag{9.6}$$

No need for punctual intervals. Notice that Formula 9.4 uses a *punctual* interval, i.e., it specifies an exact time distance through the Futr operator. Nonetheless, this is not necessary, as far as the expression of the strict *until* is concerned. In fact, we provide alternative equivalent formulas that do not use punctual intervals (of course, provided the strict until itself is constrained by a non-punctual interval). They will be useful in defining the relation between $^{\mathbb{R}}_{\mathbb{Z}}$TRIO and MITL in Section 9.2.2. The proofs are provided in Appendix D.4.3.

This time, we start with the] variant of the *until* operator, and we first establish the following (for $a > 0$).

$$\widetilde{\text{Until}}_{[a,b]}(\phi_1, \phi_2) \;\Leftrightarrow\; \text{Lasts}_{\text{ei}}\big(\text{Until}_{[0,+\infty)]}(\phi_1, \phi_2), a\big) \wedge \text{Until}_{[a,b]}(\text{true}, \phi_2 \wedge \phi_1) \tag{9.7}$$

Finally, in the following we express the) variant through the] variant (as always, assume $a > 0$).

$$\widetilde{\text{Until}}_{[a,b)}(\phi_1, \phi_2) \;\Leftrightarrow\; \widetilde{\text{Until}}_{[a,b]}(\phi_1, \phi_2) \;\vee\; \big(\text{Lasts}_{\text{ee}}\big(\text{Until}_{(0,+\infty))}(\phi_1, \neg\phi_1 \wedge \phi_2), a\big) \wedge \text{Until}_{[a,b)}(\text{true}, \neg\phi_1 \wedge \phi_2)\big) \tag{9.8}$$

When the left bound is zero. Finally, we have to handle the $a = 0$ case for the intervals. Let us start with left-open intervals; the case for left-closed will be an immediate consequence.

Now, we finally need the above results about the expressibility of the $\epsilon\widetilde{\text{NowOn}}$ operator. In fact, we prove the following (see proof in Appendix D.4.3).

$$\widetilde{\text{Until}}_{(0,b)}(\phi_1, \phi_2) \;\Leftrightarrow\; \text{Until}_{(0,b)}(\text{true}, \phi_2) \wedge \epsilon\widetilde{\text{NowOn}}\big(\text{Until}_{(0,+\infty))}(\phi_1, \phi_2)\big) \tag{9.9}$$

Then, it is simple to express the case in which the left endpoint is included.

$$\widetilde{\text{Until}}_{[0,b\rangle)}(\phi_1, \phi_2) \;\Leftrightarrow\; \widetilde{\text{Until}}_{(0,b\rangle)}(\phi_1, \phi_2) \vee \phi_2 \tag{9.10}$$

Finally, notice that Formulas 9.5 and 9.6 are valid also when $a = 0 < b$. For $a = b = 0$ it is routine to verify the following, which concludes our set of equivalences.

$$\widetilde{\text{Until}}_{[0,0]\rangle}(\phi_1, \phi_2) \;\Leftrightarrow\; \phi_2 \tag{9.11}$$

Switching to other semantic models. Let us conclude this section with a remark. Notice that the above equivalence proofs still hold if we make some variations on the semantic model according to which we interpret $^{\mathbb{R}}_{\mathbb{Z}}$TRIO formulas. In particular, the equivalence between strict and non-strict operators holds even for *mono-infinite time domains* (namely, the nonnegative reals $\mathbb{R}_{\geq 0}$), and for the *future-only* fragment of $^{\mathbb{R}}_{\mathbb{Z}}$TRIO (that is, the fragment which does not use the Since operator). This remark will be useful in comparing the expressiveness of $^{\mathbb{R}}_{\mathbb{Z}}$TRIO to that of other metric temporal logics.

9.2.2. $^{\mathbb{R}}_{\mathbb{Z}}$TRIO, MTL and MITL

This section compares $^{\mathbb{R}}_{\mathbb{Z}}$TRIO with the metric temporal logic MTL [Koy90, AH93], and its syntactic fragment MITL [AFH96].

Syntax and Semantics of MTL and MITL

Let us start by introducing formally the syntax and semantics of MTL (and MITL).

Koymans's and Alur and Henzinger's MTL. First of all, let us remark a fact that is usually left implicit (or only briefly hinted at) in the literature. There are actually *two* metric temporal logics that are referred to as "MTL".

The original one is that first defined by Koymans in [Koy90]. Notice that Koymans's MTL permits full quantification on variables, including time variables, as well as the expression of arithmetic relations between time bounds (or other variables). Therefore, as it was originally defined, MTL is a very expressive (and fundamentally undecidable) language which may be compared to full TRIO for several of its features.

Afterward, Alur and Henzinger published several in-depth analyses of the expressiveness of metric temporal logics, including MTL. In particular

[AH93] shows that a suitable subset of Koymans's MTL, interpreted over the naturals, can be regarded as an expressively complete (elementarily decidable) fragment of a monadic first-order language over the time sort. Even if [AH93] simply calls "MTL" the proposed logic, it is indeed a proper subset of Koymans's "full" MTL, in that it is purely propositional, and only intervals with constant endpoints are allowed. Much subsequent literature has also used simply the term "MTL" to refer to Alur and Henzinger's MTL *fragment*.

In the same vein, we warn the reader that in this paper the name "MTL" will refer to MTL *à la* Alur and Henzinger, not Koymans's. Moreover, we consider the MTL variant with past operators, as also done in [AH93].

MTL syntax. MTL syntax is defined by the following grammar:

$$\phi \quad ::= \quad \xi \mid \phi_1 \mathsf{U}_I \phi_2 \mid \phi_1 \mathsf{S}_I \phi_2 \mid \neg\phi \mid \phi_1 \wedge \phi_2$$

where I is an interval of $\mathbb{R}_{\geq 0}$[4], whose endpoints are constants.

It is customary to define some derived operators in MTL (as it is done with $^{\mathbb{R}}_{\mathbb{Z}}$TRIO). In Table 9.2 we list the most common ones. Also notice that

Operator	Definition	Name
$\Diamond_I \phi$	$\mathsf{true}\mathsf{U}_I \phi$	time-constrained *eventually*
$\Box_I \phi$	$\neg\Diamond_I \neg\phi$	time-constrained *always*
$\phi_1 \mathsf{W}_I \phi_2$	$\neg\left(\neg\phi_2 \mathsf{U}_I \neg\phi_1\right)$	time-constrained *unless*
$\overleftarrow{\Diamond}_I \phi$	$\mathsf{true}\mathsf{S}_I \phi$	time-constrained *eventually* (past)
$\overleftarrow{\Box}_I \phi$	$\neg\overleftarrow{\Diamond}_I \neg\phi$	time-constrained *always* (past)
$\phi_1 \mathsf{T}_I \phi_2$	$\neg\left(\neg\phi_2 \mathsf{S}_I \neg\phi_1\right)$	time-constrained *unless* (past)

Table 9.2.: MTL derived temporal operators

we freely introduce abbreviations to denote intervals. Namely, we denote as $\leq u$, $< u$, $\geq l$, $> l$, and $= l$ the intervals $[0, u]$, $[0, u)$, $[l, +\infty)$, $(l, +\infty)$, and $[l, l]$, respectively.

[4] Another difference with respect to most papers dealing with MTL is that they usually consider intervals of the rationals or integers only. These restrictions are however adopted solely to achieve decidability properties, which we do not deal with here; hence, we do not adopt such restrictions — that are however orthogonal to the notion of sampling invariance and to the related issues — in our presentation of MTL.

MTL semantics. In defining MTL semantics, we consider operators that are strict in their first arguments (as it is more common in MTL-related literature), and we adopt bi-infinite temporal domains, in conformance with $^{\mathbb{R}}_{\mathbb{Z}}$TRIO. Moreover, we consider interpretations over behaviors, adapting the standard interval-based sequence interpretation of MTL (and MITL, see e.g. [AFH96]). Finally, we remark that standard MTL semantics does not use the] variation of the *until* (and *since*) operators, where the first and the second argument are required "to meet", which is instead introduced in $^{\mathbb{R}}_{\mathbb{Z}}$TRIO.

$b(t) \models_{\mathbb{T}} \xi$	iff	$\xi\|_{b(t)}$
$b(t) \models_{\mathbb{T}} \phi_1 \mathsf{U}_I \phi_2$	iff	there exists $d \in I$ such that $b(t+d) \models_{\mathbb{T}} \phi_2$ and, for all $u \in (0,d)$ it is $b(t+u) \models_{\mathbb{T}} \phi_1$
$b(t) \models_{\mathbb{T}} \phi_1 \mathsf{S}_I \phi_2$	iff	there exists $d \in I$ such that $b(t-d) \models_{\mathbb{T}} \phi_2$ and, for all $u \in (0,d)$ it is $b(t-u) \models_{\mathbb{T}} \phi_1$
$b(t) \models_{\mathbb{T}} \neg\phi$	iff	$b(t) \not\models_{\mathbb{T}} \phi$
$b(t) \models_{\mathbb{T}} \phi_1 \wedge \phi_2$	iff	$b(t) \models_{\mathbb{T}} \phi_1$ and $b(t) \models_{\mathbb{T}} \phi_2$
$b \models_{\mathbb{T}} \phi$	iff	for all $t \in \mathbb{T}$: $b(t) \models_{\mathbb{T}} \phi$

MITL. MITL [AFH96] is simply defined as a syntactic subset of MTL, where all intervals I in formulas are required to be *non-singular*, i.e., not in the form $[l,l]$ for any $l \in \mathbb{R}_{\geq 0}$.

Equivalence between MTL and $^{\mathbb{R}}_{\mathbb{Z}}$TRIO

In order to show that MTL and $^{\mathbb{R}}_{\mathbb{Z}}$TRIO are equally expressive languages, we provide two translations — one from MTL formulas to $^{\mathbb{R}}_{\mathbb{Z}}$TRIO and the other from $^{\mathbb{R}}_{\mathbb{Z}}$TRIO formulas to MTL — that preserve truth of formulas. Notice that these translations will make use of the equivalence between strict and non-strict operators shown in Section 9.2.1; therefore, we will use strict versions of $^{\mathbb{R}}_{\mathbb{Z}}$TRIO operators whenever needed. For brevity, we omit the proofs that the translations preserve truth of formulas, as they are straightforward by case discussion.

Translating MTL to $^{\mathbb{R}}_{\mathbb{Z}}$TRIO. The function $\sharp\{\cdot\}$ provides a translation from MTL formulas to $^{\mathbb{R}}_{\mathbb{Z}}$TRIO ones that preserves truth values.

$$
\begin{array}{lcl}
\sharp\{\xi\} & \equiv & \xi \\
\sharp\{\neg\phi\} & \equiv & \neg\, \sharp\{\phi\} \\
\sharp\{\phi_1 \wedge \phi_2\} & \equiv & \sharp\{\phi_1\} \wedge \sharp\{\phi_2\} \\
\sharp\{\phi_1 \mathsf{U}_I \phi_2\} & \equiv & \widetilde{\mathrm{Until}}_{I)}(\sharp\{\phi_1\}, \sharp\{\phi_2\}) \\
\sharp\{\phi_1 \mathsf{S}_I \phi_2\} & \equiv & \widetilde{\mathrm{Since}}_{I)}(\sharp\{\phi_1\}, \sharp\{\phi_2\})
\end{array}
$$

Translating $^{\mathbb{R}}_{\mathbb{Z}}$TRIO to MTL. The function $\flat\{\cdot\}$ provides a translation from $^{\mathbb{R}}_{\mathbb{Z}}$TRIO formulas to MTL ones that preserves truth values. In this case, we have to take care of $^{\mathbb{R}}_{\mathbb{Z}}$TRIO's non-strict operators (which are simply expressible using MTL's strict ones), and to deal with the variation of $^{\mathbb{R}}_{\mathbb{Z}}$TRIO's operators with a] subscript (which is also rather simply reducible to the standard variation).

$$
\begin{array}{lcl}
\flat\{\xi\} & \equiv & \xi \\
\flat\{\neg\phi\} & \equiv & \neg\, \flat\{\phi\} \\
\flat\{\phi_1 \wedge \phi_2\} & \equiv & \flat\{\phi_1\} \wedge \flat\{\phi_2\} \\
\flat\big\{\mathrm{Until}_{I\rangle}(\phi_1, \phi_2)\big\} & \equiv & \begin{cases}
\flat\{\phi_1\} \wedge (\flat\{\phi_1\}\, \mathsf{U}_I \flat\{\phi_2\}) \\
\qquad \text{if } 0 \notin I \text{ and } \rangle \text{ is)} \\
\flat\{\phi_2\} \vee \big(\flat\{\phi_1\} \wedge \big(\flat\{\phi_1\}\, \mathsf{U}_{(0,u\rangle} \flat\{\phi_2\}\big)\big) \\
\qquad \text{if } 0 \in I = [0, u\rangle \text{ and } \rangle \text{ is)} \\
\flat\{\phi_1\} \wedge (\flat\{\phi_1\}\, \mathsf{U}_I\, \flat\{\phi_2 \wedge \phi_1\}) \\
\qquad \text{if } 0 \notin I \text{ and } \rangle \text{ is]} \\
\flat\{\phi_1 \wedge \phi_2\} \vee \big(\flat\{\phi_1\} \wedge \big(\flat\{\phi_1\}\, \mathsf{U}_{(0,u\rangle}\, \flat\{\phi_2 \wedge \phi_1\}\big)\big) \\
\qquad \text{if } 0 \in I = [0, u\rangle \text{ and } \rangle \text{ is]}
\end{cases} \\
\flat\big\{\mathrm{Since}_{I\rangle}(\phi_1, \phi_2)\big\} & \equiv & \begin{cases}
\flat\{\phi_1\} \wedge (\flat\{\phi_1\}\, \mathsf{S}_I \flat\{\phi_2\}) \\
\qquad \text{if } 0 \notin I \text{ and } \rangle \text{ is)} \\
\flat\{\phi_2\} \vee \big(\flat\{\phi_1\} \wedge \big(\flat\{\phi_1\}\, \mathsf{S}_{(0,u\rangle} \flat\{\phi_2\}\big)\big) \\
\qquad \text{if } 0 \in I = [0, u\rangle \text{ and } \rangle \text{ is)} \\
\flat\{\phi_1\} \wedge (\flat\{\phi_1\}\, \mathsf{S}_I\, \flat\{\phi_2 \wedge \phi_1\}) \\
\qquad \text{if } 0 \notin I \text{ and } \rangle \text{ is]} \\
\flat\{\phi_1 \wedge \phi_2\} \vee \big(\flat\{\phi_1\} \wedge \big(\flat\{\phi_1\}\, \mathsf{S}_{(0,u\rangle}\, \flat\{\phi_2 \wedge \phi_1\}\big)\big) \\
\qquad \text{if } 0 \in I = [0, u\rangle \text{ and } \rangle \text{ is]}
\end{cases}
\end{array}
$$

$^{\mathbb{R}}_{\mathbb{Z}}$TRIO and MITL

As we recalled above, MITL is the syntactic subset of MTL where intervals are all non-singular. We have shown that MTL is as expressive as $^{\mathbb{R}}_{\mathbb{Z}}$TRIO. Moreover, as we noticed in Section 9.2.1, the expression of strict operators using non-strict ones does not require to change the non-singularity of the intervals that are involved. In other words, any MTL formula involving non-singular intervals can be expressed as an $^{\mathbb{R}}_{\mathbb{Z}}$TRIO formula also involving non-singular intervals only. The translation $\flat\{\cdot\}$ shows that the converse also holds: any $^{\mathbb{R}}_{\mathbb{Z}}$TRIO formula involving non-singular intervals only can be rendered as a MTL formula without singular intervals.

All in all, the syntactic restriction on $^{\mathbb{R}}_{\mathbb{Z}}$TRIO formulas that requires that all intervals are non-singular yields a language whose expressiveness coincide with that of MITL's. Therefore, all general results about MITL decidability and expressiveness are retained when dealing with $^{\mathbb{R}}_{\mathbb{Z}}$TRIO formulas without singular intervals — of course, provided the same semantic assumptions (e.g., point-based vs. interval-based, mono-infinite vs. bi-infinite, etc.) are made).

9.2.3. Expressiveness and Decidability Issues

Now, thanks to the equivalence between $^{\mathbb{R}}_{\mathbb{Z}}$TRIO and MTL (or MITL) we are able to collocate $^{\mathbb{R}}_{\mathbb{Z}}$TRIO's relative expressiveness, by drawing from the several works about the expressiveness of MTL and variants thereof that are available in the literature. To this end, Appendix E reviews several relevant works about such an important topic.

Notice, however, that most — if not all — works in the literature consider a semantics for MTL which is different than the one we introduced above. Nonetheless, according to the remark outlined at the end of Section 9.2.1, our equivalence results still hold for two common semantics, and namely:

- the use of a mono-infinite time domain, namely the nonnegative reals $\mathbb{R}_{\geq 0}$;
- the use of future-only operators (i.e., only Until and not Since as temporal operator).

Therefore, we are able to collocate adequately $^{\mathbb{R}}_{\mathbb{Z}}$TRIO among other logics interpreted in the interval-based semantics.

An issue that we do not deal with here, for brevity, is how our results can be adapted when dealing with another popular semantic model, namely

the point-based semantics (a.k.a. the timed trace). This is an interesting direction which belongs to future work. We notice, however, that the point-based semantics carries several peculiar features that sometimes may make it somewhat unwieldy to model naturally some features of real-time systems (see [HR04] and its summary in Appendix E). We believe that this defends our choice of focusing on the interval-based semantics *first*.

Let us conclude this section by adding a couple of notes about the expressiveness of $^{\mathbb{R}}_{\mathbb{Z}}$TRIO with respect to that of full TRIO, and that of nesting-free $^{\mathbb{R}}_{\mathbb{Z}}$TRIO formulas. Notice that we deliberately stay on an informal level: rather than proving the stated expressiveness results, we just give some evidence, and refer to other works for further details. In this case, we prefer this approach for the sake of brevity and clarity, as a detailed and completely formal analysis of these issues would likely distract us from our main focus.

$^{\mathbb{R}}_{\mathbb{Z}}$TRIO and TRIO. It is rather obvious that $^{\mathbb{R}}_{\mathbb{Z}}$TRIO is strictly less expressive than full TRIO. The latter is a very expressive language, which even includes all arithmetic and full first-order quantification. $^{\mathbb{R}}_{\mathbb{Z}}$TRIO is instead purely propositional, and considers only constant bounds for time intervals. This is enough to separate the two expressive powers.

Nesting operators and expressiveness. An issue which has not been investigated often for metric temporal logics over dense time is how the nesting depth of temporal operators impacts the expressiveness of the resulting language. It is however to be expected that the nesting depth of temporal operators defines an expressively strict hierarchy of formulas, that is each level of nesting introduces a larger class of expressible properties. In particular, it should be possible to prove that nesting-free $^{\mathbb{R}}_{\mathbb{Z}}$TRIO formulas define a class of properties which is strictly contained in that defined by $^{\mathbb{R}}_{\mathbb{Z}}$TRIO formulas that nest temporal operators at any depth.

Moreover, notice that the results about the expressively strict nesting hierarchy also hold in the discrete time setting. Therefore, it is likely that the same holds for continuous-time behaviors which are restricted by the constraint χ, i.e., of bounded variability, since they can be basically described as discrete histories.

We refer the reader to Appendix E where we summarize some related works about the problem of the expressiveness of nesting with metric temporal logics over dense (and discrete) time. In particular, we mention here

the works [AH92a,KS05] for their results, and [BCM05,PD06] for the proof techniques they use, which rely on characterizing the expressiveness of a temporal logic formula given the values of its interval bounds and the nesting depth of its operators.

9.3. Berkeley and Non-Berkeley Behaviors

This section considers behaviors subject to the regularity constraint χ — introduced in Section 8.2.2 — shows some operators that preserve the regularity (Section 9.3.1), and studies how such a regularity can be characterized when dealing with items varying over dense domains (Section 9.3.2).

Let us recall that Abadi and Lamport introduced the term *Zeno* [AL94] to identify those behaviors where time converges to a finite value, and thus, in a sense, "stops". The name is after the ancient Greek philosopher Zeno of Elea[5] and his paradoxes on time advancement. In the same vein, we propose to call "*Berkeley* behaviors" those behaviors failing to satisfy a regularity constraint such as χ, from the name of the famous Irish philosopher George Berkeley[6] who criticized the use of the notion of *infinitesimal* among the founding principles of calculus. In fact, in Berkeley behaviors the distance between any two transitions is infinitesimal, that is indefinitely small (or with infimum equal to zero), even if the infinitesimal times may not accumulate (otherwise, the behavior is also Zeno). Therefore, in the remainder of this report we will refer to behaviors satisfying the constraint χ as *non-Berkeley behaviors*.

9.3.1. Shiftable Operators

The results about the sampling invariance of $^{\mathbb{R}}_{\mathbb{Z}}$TRIO we presented in Chapter 8 consider only nesting-free formulas. In general, as we have argued in Section 9.2.3, this restricts the expressiveness of our formal language. In the previous chapter we have also shown how to put any $^{\mathbb{R}}_{\mathbb{Z}}$TRIO formula into a nesting-free one *by introducing auxiliary items*. This does not contradict the results on the expressiveness: in fact the auxiliary items are then subject to the constraint χ, and therefore we end up with a nesting-free formula which is, in general, stronger than the original one, as any subformula of the original formula is required to hold over intervals at least as

[5] Circa 490–430 B.C.
[6] 1685–1753 A.D.

long as δ.

In this respect, we now investigate what kind of $\frac{\mathbb{R}}{\mathbb{Z}}$TRIO formulas preserve the χ constraint, or, in other words, what formulas can be nested without need for introducing additional constraints. The basic idea is the following: if, for any behavior b, a formula holding at some instant t is shown to be true over a full δ-length interval that contains t, then the non-Berkeley variability of the basic items can be "lifted" up to the truth value of the formula itself over time, thus allowing one to nest the formula without introducing additional constraints.

Under this respect, we introduce the following definitions, parametric with respect to a parameter $\epsilon > 0$.

Definition 9.3.1 (ϵ-Shiftable Operator)**.** An n-argument operator Op interpreted over the basic items $\xi_1, \ldots, \xi_n$ is:

- *positively ϵ-shiftable* iff, for all behaviors $b \in [\![\chi_\circ]\!]_{\mathbb{R}}$, whenever $b(t) \models_{\mathbb{R}} \mathrm{Op}(\xi_1, \ldots, \xi_n)$ for some t, there exists an interval $I = [u, u + \epsilon]$ such that $t \in I$ and, for all $v \in I$ it is $b(v) \models_{\mathbb{R}} \mathrm{Op}(\xi_1, \ldots, \xi_n)$;
- *negatively ϵ-shiftable* iff, for all behaviors $b \in [\![\chi_\circ]\!]_{\mathbb{R}}$, whenever $b(t) \models_{\mathbb{R}} \neg\mathrm{Op}(\xi_1, \ldots, \xi_n)$ for some t, there exists an interval $I = [u, u + \epsilon]$ such that $t \in I$ and, for all $v \in I$ it is $b(v) \models_{\mathbb{R}} \neg\mathrm{Op}(\xi_1, \ldots, \xi_n)$;
- *ϵ-shiftable* iff it is both positively and negatively ϵ-shiftable, and its "change points" for some behavior b, that is the instants at which it switches its truth value with respect to b, coincide with some change points of b.[7] In other words, the change points of the operator are a subset (possibly equal to) of the changing points of b.

Therefore, if an operator is δ-shiftable, then its truth value over time respects the constraint χ if its arguments do. Therefore, it can be nested without impacting sampling invariance.

Non-Shiftable Operators

Let us start by showing that — unsurprisingly — not all operators are δ-shiftable. In particular, let us show that the Until operator is not, at

[7] Notice that this additional requirement on the "change points" is not redundant. For instance the "rigid shift" of a basic item ξ (i.e., $\mathrm{Futr}(\xi, \tau)$) does change its value at instants other than those of the item itself, even if it is both positively and negatively δ-shiftable. We also note that this requirement is similar to that of *stability* introduced elsewhere [Rab03].

least in its most general application. Let ξ_1, ξ_2 be two basic Boolean time-dependent items. Let us consider the behavior b in Figure 9.1, such that ξ_1 and ξ_2 both hold over $[t, t+\delta]$, and are both false everywhere else. Clearly

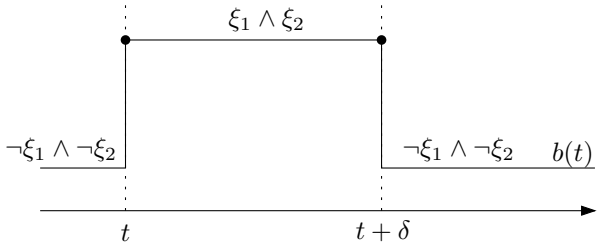

Figure 9.1.: Until is non-shiftable.

$b \in [\![\chi_\circ]\!]_{\mathbb{R}}$, and $b(t) \models_{\mathbb{R}} \text{Until}_{[\delta,+\infty)]}(\xi_1, \xi_2)$, but $\text{Until}_{[\delta,+\infty)]}(\xi_1, \xi_2)$ is false throughout $I' = (-\infty, t)$ because ξ_1 is false in any right-neighborhood of a point in I', and it is false through $I'' = (t, +\infty)$, because ξ_2 is always false in the interval $(t+\delta, +\infty)$. Therefore, the Until operator is in general non shiftable.

In the remainder of this section, we consider some much more restricted applications of the Until (and Since) operator, in particular by avoiding metric constraints and thus expressing only qualitative properties; we are able to show that such restrictions are δ-shiftable.

Qualitative Formulas

Although it is simple to see that "existential" operators — such as WithinF — are positively δ-shiftable under simple restrictions of the bound (e.g., for $\text{WithinF}(\xi, u)$ such that it is $u > \delta$), and dually "universal" operators — such as Lasts — are negatively δ-shiftable under the same restrictions, the two facts are in contrast, so that existential operators that are positively δ-shiftable are not negatively δ-shiftable, and vice versa for the universal operators. More explicitly, it is relatively easy to check that $\text{WithinF}(\xi, 2\epsilon)$ is positively ϵ-shiftable but not negatively so, and its negation $\text{Lasts}(\neg\xi, 2\epsilon)$ is negatively ϵ-shiftable (and not positively so). Therefore, positive and negative shiftability are orthogonal notions, as one does not imply the other.

These considerations suggest that it is the use of *metric* itself that hinders the shiftability of temporal operators. Therefore, we try to consider

operators that define *qualitative* properties, and we show that these are indeed fully shiftable.

Definition 9.3.2 (Qualitative Formulas)**.** A formula ϕ is qualitative iff one of the following applies:

- $\phi = \xi$ for some basic item ξ;
- $\phi = \text{Until}_{I\rangle}(\phi_1, \phi_2)$ with $I = [0, +\infty)$ and ϕ_1, ϕ_2 qualitative formulas;
- $\phi = \text{Since}_{I\rangle}(\phi_1, \phi_2)$ with $I = [0, +\infty)$ and ϕ_1, ϕ_2 qualitative formulas;
- ϕ is a Boolean combination of qualitative formulas

Establishing that qualitative formulas are δ-shiftable amounts to tediously considering several cases: we sketch the proof for the most important ones.

Qualitative Until is δ-shiftable. Let ξ_1, ξ_2 be two basic Boolean time-dependent items, and let us show that the formula $\phi = \text{Until}_{[0,+\infty)\rangle}(\xi_1, \xi_2)$ is δ-shiftable.

Positively shiftable. Let $b \in [\![\chi_\circ]\!]_{\mathbb{R}}$ be any non-Berkeley behavior, and let t be an instant such that $b(t) \models_{\mathbb{R}} \phi$. Thus, there exists a $d \in [t, +\infty)$ such that $b(d) \models_{\mathbb{R}} \xi_2$, and for all $u \in [t, d\rangle$ it is $b(u) \models_{\mathbb{R}} \xi_1$. Let us consider several cases:

- if $d \geq t + \delta$, ϕ can be shifted forward over the interval $[t, t + \delta]$;
- if $d < t + \delta$, then there exist t', d' such that $t' \leq t$, $d' \geq d$, $d' - t' = \delta$ and for all $v' \in [t', d']$ it is $b(v') \models_{\mathbb{R}} \xi_1$. Therefore ϕ can be surely shifted over the interval $[t', d)$. Let us further distinguish two cases:
  - if $t - t' \geq \delta$ we are done, as ϕ is positively shiftable over $[t', t]$;
  - otherwise, it must be $d' > d$ and $t - t' < \delta$; therefore notice that ξ_2 cannot switch its value to false before or at d', otherwise there would be two switches within δ time units (against the hypothesis $b \in [\![\chi_\circ]\!]_{\mathbb{R}}$). Hence, ϕ can be shifted over the interval $[t', d']$, and we are done.

This proves that qualitative Until is positively δ-shiftable. For brevity, we omitted some finer-grain details in the above proof sketch, but they can be easily reconstructed by the observant reader.

Negatively shiftable. Let us now show that Until is negatively δ-shiftable. This is the same as proving that $\phi = \text{Releases}_{[0,+\infty)\rangle}(\xi_1, \xi_2)$ is positively δ-shiftable. Note that we drop the negations over ξ_1, ξ_2 as they are inessential in the proof (i.e., the proof has a symmetry with respect to complementing the truth value of primitive conditions). Thus, let $b \in [\![\chi_\circ]\!]_{\mathbb{R}}$ be any non-Berkeley behavior, and let t be an instant such that $b(t) \models_{\mathbb{R}} \phi$: for all $d \in [t, +\infty)$ either $b(d) \models_{\mathbb{R}} \xi_2$ or there exists a $u \in [t, d\rangle$ such that $b(u) \models_{\mathbb{R}} \xi_1$.

Proceeding by contradiction simplifies a bit the proof. Thus let us assume that ϕ is not positively δ-shiftable. In particular, this is the case if there exists $r \in [t, t + \delta]$ such that $\text{Until}_{[0,+\infty)\rangle}(\neg\xi_1, \neg\xi_2)$ holds at r. Therefore, there exists a $d_r \in [r, +\infty)$ such that $b(d_r) \models_{\mathbb{R}} \neg\xi_2$ and for all $u_r \in [r, d_r\rangle$ it is $b(u_r) \models_{\mathbb{R}} \neg\xi_1$. Then, from the definition of the Releases operator, there must be a $u \in [t, d_r\rangle$ such that $b(u) \models_{\mathbb{R}} \xi_1$. To avoid contradictions, it must be $u \in [t, r)$. Moreover, let u' be the largest instant in $[t, r]$ such that $\epsilon\widetilde{\text{UpToNow}}(\xi_1)$ holds at u' (it may be that $u' = u$). Then, in compliance with the regularity constraint χ, for all $v' \in [u' - \delta, u')$ it must be $b(v') \models_{\mathbb{R}} \xi_1$. Notice that $t \in [u' - \delta, u')$. But then, ϕ is shiftable over $[u' - \delta, u')$, and a little reasoning (still based on the properties implied by χ) lets us conclude that we can even extend the interval to a right-closed one (either by including u' or by shifting its left endpoint to the left a bit before $u' - \delta$), so that ϕ is indeed δ-shiftable, against the assumption.

Shiftable. We need a key observation to conclude that Until is δ-shiftable; it is not unlike some considerations we did when proving closure under inverse sampling (see proofs in Appendix D).. A δ-shiftable quantitative Until or Since has indeed the additional property of changing its truth value in correspondence with some basic item changing its value. For instance, it is easy to see that $\phi = \text{Until}_{[0,+\infty)\rangle}(\xi_1, \xi_2)$ changes its value to false exactly when either ξ_2 becomes false, or ξ_1 does. To this end, consider an instant at which $b(t) \models_{\mathbb{R}} \phi$; then there exists a $u \in [t, +\infty)$ such that $b(u) \models_{\mathbb{R}} \xi_2$ and for all $v \in [t, u\rangle$ it is $b(v) \models_{\mathbb{R}} \xi_1$. If we can shift ϕ to the left, then ξ_1 must be true also on some interval of the form $\langle t', t)$ for some $t' < t$; ϕ can only become false when ξ_1 also does, and with the same "edge" (i.e., right-continuously or not). On the other hand, if $u > t$ we can surely shift ϕ to the right, at least until instant u; afterward, ϕ switches to false only if ξ_2 does so, or ξ_1 does, or both. All in all, qualitative Until (and Since) is δ-shiftable. Therefore, the application of a qualitative temporal operator yields a formula whose truth value over time respects the constraint χ

together with the other basic items.

Boolean combinations are shiftable. That δ-shiftable formulas are closed under complement is apparent from Definition 9.3.1. Thus, let us just prove that conjunction of formulas preserves δ-shiftability. For a formula $\phi = \phi_1 \wedge \phi_2$ such that ϕ_1 and ϕ_2 are both δ-shiftable, not only do ϕ_1 and ϕ_2 each hold over a δ-length interval, but they also hold over a *common* δ-length interval. This is due to the fact that they both shift values together with some basic items, and these have "change points" that are at least δ time units apart, due to χ. In particular, notice that situations such as the one in Figure 9.2 cannot happen as the value of b changes from (T, F) to (F, F) and then to (F, T) in less than δ time units. This means that any conjunction ϕ also holds over a δ-length interval. Moreover, note that the same argument about the instants where the truth value change can be lifted the the conjunction formula ϕ itself, as any of ϕ_1 or ϕ_2 becoming false implies that ϕ becomes false there as well. All in all, conjunction is δ-shiftable.

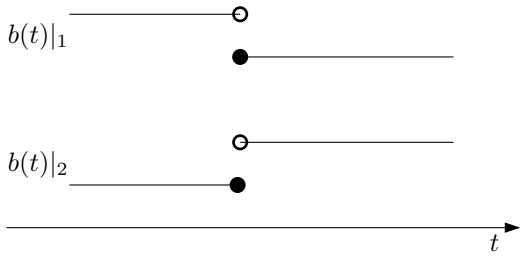

Figure 9.2.: A bi-dimensional behavior not complying with χ.

Left-open intervals are not δ-shiftable. Let us also remark that the adoption of a left-closed interval in the Until is necessary to have δ-shiftable formulas. In fact, Figure 9.1 serves as a counterexample: $\mathrm{Until}_{(0,\infty))}(\xi_1, \xi_2)$ only holds in the right-open interval $[t, t+\delta)$, while it is false precisely at $t+\delta$, since ξ_2 is false everywhere in the interval $(t+\delta, +\infty)$.

Sufficiency and (Non) Necessity of Shiftability

Let us now provide the straightforward proofs that having a formula that nests shiftable operators *only* is a sufficient but not necessary condition for

the formula to be closed under sampling.

Shiftability is sufficient for closure under sampling. This comes straightforwardly from the definition of δ-shiftability. In fact, let us consider a formula ϕ that nests some operators. Let ϕ' be the formula obtained by putting ϕ in normal form (according to the definition given in Section 8.2.3), and in particular by "unnesting" all operators by introducing additional items $\xi'_1, \xi'_2, \ldots$.

Then, let us consider any of these additional items ξ'_i. In the normal form, we have a constraint of the form $\xi'_i \Leftrightarrow \mathrm{Op}(\cdots)$, where Op is a δ-shiftable operator by hypothesis. Therefore, the constraint χ simply evaluates to true for the item ξ'_i. In other words, the item ξ'_i satisfies all requirements for closure under sampling; in particular, the formulas in which it is mentioned are closed under sampling. Since the same applies to any additional item ξ'_i, all in all the original formula ϕ is closed under sampling, with the constraint χ applying to basic items only.

Shiftability is not necessary for closure under sampling. Let us consider the formula $\zeta = \mathrm{Lasts}_{\mathrm{ee}}(\xi, \tau)$, for any $\tau > \delta$. Clearly, ζ is not δ-shiftable according to the definition, as the item ξ is not constrained to stay true out of the τ-length interval introduced by the $\mathrm{Lasts}_{\mathrm{ee}}$ operator. Nonetheless, it is easy to see that $b \models_{\mathbb{R}} \mathrm{Alw}(\zeta)$ is the same as $b \models_{\mathbb{R}} \mathrm{Alw}(\xi)$. The formula $\mathrm{Alw}(\xi)$ is obviously sampling invariant. Therefore, so is the formula $\mathrm{Alw}(\zeta)$, even if it nests a non-δ-shiftable operator.[8]

Shiftability and closure under inverse sampling. Notice that δ-shiftability is a requirement on formulas interpreted over continuous time, whereas the notion of closure under inverse sampling applies to formulas interpreted over discrete time. Nonetheless, the above results about shiftability have consequences also to closure under inverse sampling. In fact, it is easy to see that whenever a formula ϕ nests only subformulas $\varphi_1, \varphi_2, \ldots$ all of whose adaptations $\eta_\delta^{\mathbb{Z}}\{\varphi_1\}, \eta_\delta^{\mathbb{Z}}\{\varphi_2\}, \ldots$ are δ-shiftable, then ϕ is closed under inverse sampling. Thus, shiftability of the discrete-to-continuous adaptation

[8] As an aside, let us point out that the definition of sampling invariance implies that whenever two formulas ϕ_1, ϕ_2 are equivalent both in dense and in discrete time, and so are their adaptations (i.e., $\eta_\delta^{\mathbb{R}}\{\phi_1\} \equiv \eta_\delta^{\mathbb{R}}\{\phi_2\}$ in discrete time, and $\eta_\delta^{\mathbb{Z}}\{\phi_1\} \equiv \eta_\delta^{\mathbb{Z}}\{\phi_2\}$ in dense time), then ϕ_1 is sampling invariant if and only if ϕ_2 is also sampling invariant.

of operators is a sufficient condition for a formula to be closed under inverse sampling. It is also straightforward to realize that the condition is not necessary (we could provide examples along the same lines as the one above about closure under sampling).

Finally, let us point out that this requirement cannot be moved to the non-adapted formula itself, for qualitative formulas. In other words, there are qualitative formulas whose discrete-to-continuous adaptations are not δ-shiftable (thus, in particular, are not qualitative formulas). Therefore, in general nesting qualitative formulas in discrete-time formulas cannot be done "for free" — that is without introducing additional constraints on the variability of the adapted formula — for the sake of closure under inverse sampling. As a proof of this fact, let us consider the formula $\phi = \mathrm{Until}_{[0,+\infty)]}(\xi_1, \xi_2)$ and its discrete-to-continuous adaptation $\phi' = \eta_\delta^{\mathbb{Z}}\{\mathrm{Until}_{[0,+\infty)]}(\xi_1, \xi_2)\}$. It is easy to see that ϕ' is equivalent to $\mathrm{WithinP}_{\mathrm{ie}}(\xi_2, \delta) \vee \mathrm{Until}_{[0,+\infty)]}(\xi_1, \xi_2)$; let us show that ϕ' is not δ-shiftable. To this end, let us consider the behavior b represented by the interval-based sequence $\langle(-\infty, 0.7], \xi_2 \wedge \neg\xi_1\rangle, \langle(0.7, +\infty), \neg\xi_2 \wedge \xi_1\rangle$. Clearly, ϕ' is positively and negatively δ-shiftable over b, since it is true in the interval $(-\infty, \delta + 0.7]$ and false in the complement interval $(\delta + 0.7, +\infty)$. However, its truth value changes at $\delta + 0.7$, which is not a change point for either ξ_1 or ξ_2 (as $\delta > 0$). Therefore, ϕ' is not (fully) δ-shiftable.

A Formula Not Closed Under Sampling

Let us now exhibit a formula ϕ^{ns} which nests non-shiftable operators and, in fact, is not closed under sampling.

$$\phi^{\mathrm{ns}} \quad \equiv \quad \mathrm{Som}(\mathrm{Lasts}_{\mathrm{ii}}(\xi, \delta))$$

Let us consider the behavior $b \in [\![\chi_\circ]\!]_{\mathbb{R}}$ of Figure 9.3 — where we assume that ξ is false everywhere except that in some interval, larger than δ and internal to $(t, t + 2\delta)$ — as a proof (by counterexample). Remember that the definition of sampling invariance requires invariance for any choice of the origin z; hence, let us choose adversarially the sampling as in the figure. Clearly $b \models_{\mathbb{R}} \phi^{\mathrm{ns}}$; nonetheless, the adaptation $\eta_\delta^{\mathbb{R}}\{\phi^{\mathrm{ns}}\}$ of ϕ^{ns} is:

$$\eta_\delta^{\mathbb{R}}\{\phi^{\mathrm{ns}}\} \quad = \quad \mathrm{Som}(\mathrm{Lasts}_{\mathrm{ii}}(\xi, 1))$$

It is easy to realize that $\mathrm{Lasts}_{\mathrm{ii}}(\xi, 1) = \xi_1 \wedge \mathrm{NowOn}(\xi_1)$ holds at no discrete sampling point of $\sigma_{\delta,z}[b]$, so $\sigma_{\delta,z}[b] \not\models_{\mathbb{Z}} \eta_\delta^{\mathbb{R}}\{\phi^{\mathrm{ns}}\}$. That is, ϕ^{ns} is not closed under sampling, and *a fortiori* nor sampling invariant.

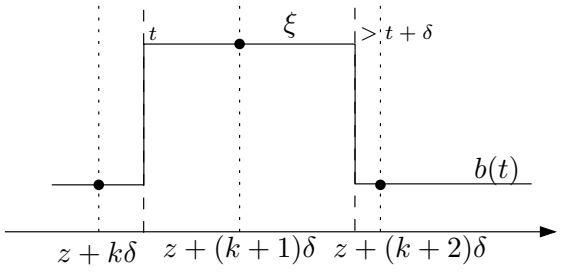

Figure 9.3.: Proof of non sampling invariance.

With a very similar proof, it can be seen that the regularity constraint χ itself is not a sampling-invariant formula (in particular, it is not closed under sampling).

9.3.2. Towards Characterizing Berkeley Behaviors of Dense-Valued Items

This section aims at characterizing non-Berkeley behaviors for basic items that map to a dense (or, more specifically, continuous) domain. More precisely, we aim at giving a mathematical characterization of the behaviors that satisfy the constraint $\chi^{\phi}_{\bullet}$ for dense-valued items.

As we have already discussed in Section 8.3.2, the behavior constraint $\chi^{\phi}_{\bullet}$ for dense-valued items depends, in general, on the particular formula ϕ that constitutes our specification; the notation $\chi^{\phi}_{\bullet}$ stresses this fact. In general, the specification formula ϕ can be very complex, and so can be the conditions in Ξ. Therefore, the actual "physical" impact of the constraint $\chi^{\phi}_{\bullet}$ may be very difficult to predict and characterize in a clear-cut manner.

To focus our discussion, let us choose a particular — yet significant and rather general — form for the conditions in Ξ. Namely, we consider conditions of the form $\mathsf{x} \in \langle \mathrm{l}, \mathrm{u} \rangle$, where x is a basic item (for simplicity, taking values to the set $\mathbb{R}$), $\mathrm{l}, \mathrm{u} \in \mathbb{R} : \mathrm{l} < \mathrm{u}$ are two constant bound values, and $\langle \in \{(, [\}$ (resp. $\rangle \in \{),]\}$) denotes whether the left (resp. right) endpoint in excluded or included. We point out that the generalization to $n > 1$ items is straightforward. In this basic case, the condition $\chi^{\phi}_{\bullet}$ becomes:

$$\chi^{\phi}_{\bullet} = \mathrm{WithinP_{ii}}(\mathrm{Lasts_{ii}}(\mathsf{x} \in \langle \mathrm{l}, \mathrm{u} \rangle, \delta), \delta) \vee \mathrm{WithinP_{ii}}(\mathrm{Lasts_{ii}}(\mathsf{x} \notin \langle \mathrm{l}, \mathrm{u} \rangle, \delta), \delta)$$

Let us now consider a generic behavior $b \in \mathcal{B}_{\mathbb{R}}$ for x, that is a generic

function $b : \mathbb{R} \to \mathbb{R}$. We now give an "approximate" mathematical characterization of the subset $[\![\chi^{\phi}_{\bullet}]\!]_{\mathbb{R}} \subset \mathcal{B}_{\mathbb{R}}$ for any ϕ where Ξ is in the form we have outlined.

Notice that we will not give a completely equivalent characterization since, as we will show, there are some aspects implied by $\chi^{\phi}_{\bullet}$ which are hard to characterize in a simple way. Nonetheless, we are going to provide a mathematical notion that can be reasonably regarded as a formal description of the behaviors satisfying $\chi^{\phi}_{\bullet}$, *within some tolerance*.

Uniformly Continuous Functions

Let us start by defining formally the notion of *uniform continuity*. As we will show, it will constitute our characterization of non-Berkeley behaviors.

Continuity. First, let us recall the well-known definition of continuous function. A function $b : \mathbb{R} \to \mathbb{R}$ is *continuous at some point* t iff: for any $\epsilon > 0$ there exists a $\delta_{\epsilon,t} > 0$ such that for all x such that $|x - t| < \delta_{\epsilon,t}$ it is: $|f(x) - f(t)| < \epsilon$. Notice that in the definition $\delta_{\epsilon,t}$ is in general a function not only of ϵ, but also of t, and the notation stresses this fact.

Then, a function b is continuous *over an interval* $I \subseteq \mathbb{R}$ iff it is continuous at all points in I.

Uniform continuity. A function $b : \mathbb{R} \to \mathbb{R}$ is *uniformly continuous over an interval* $I \subseteq \mathbb{R}$ iff: for any $\epsilon > 0$ there exists a $\delta_{\epsilon} > 0$ such that, for all $x, t \in I$ such that $|x - t| < \delta_{\epsilon}$ it is: $|f(x) - f(t)| < \epsilon$. Notice that the key difference with respect to the definition of continuity is that now δ_{ϵ} does not depend on the chosen t, but it must be unique for all points in I.

Notice that uniform continuity is a notion close to, but more general, than having bounded derivative. In fact, one can show that any function with bounded derivative (i.e., such that $\sup_{x \in I} |b'(x)| < K$ for some K) is uniformly continuous. But the converse is in general not true: for instance, the function $\sqrt{x}$ has derivative $1/(2\sqrt{x})$ which is unbounded over the open interval $(0, 1)$ as it goes to ∞ as x goes to 0. Nonetheless, it is not difficult to see that $\sqrt{x}$ is uniformly continuous over $(0, 1)$.

Sampling a Uniformly Continuous Behavior

Let us now show what is the link between uniformly continuous functions and the slow-variability requirement expressed by $\chi^{\phi}_{\bullet}$.

Uniformly continuous behaviors. Let us consider a condition Ξ of the form $\mathsf{x} \in \langle \mathrm{l}, \mathrm{u} \rangle = I$; if b is uniformly continuous over $\mathbb{R}$, then it is always possible to find a δ such that $|b(t') - b(t)| < |I| = \mathrm{u} - \mathrm{l}$ for all instants t, t' such that $|t' - t| < \delta$. Therefore, let such δ be our sampling period. Then, let us consider any instant at t which b "enters" the interval I such that b is monotonic non-decreasing over I; then b will remain in I for at least δ time units before crossing it and exiting above. Thus, for monotonic behaviors, the condition $\chi^{\phi}_{\bullet}$ is always satisfied.

More generally, however, b is non monotonic. In this case, let us pick a $2\epsilon < \mathrm{u} - \mathrm{l}$ and require that every behavior we consider does not do the following: enter I, reach at least $\mathrm{l} + \epsilon$, exit I again, all this in less than a sampling period. Then, if b is uniformly continuous, we always have a δ_ϵ which depends on 2ϵ, such that, if we pick δ_ϵ as sampling period, b satisfies the requirement.

In general, ϵ can be as small as desired: uniform continuity ensures that a suitable δ_ϵ can always be found. Obviously, this can be generalized to several intervals partitioning the domain of x, by considering the smallest of such intervals in place of the unique I. We assume that the various intervals are derived from a specification formula ϕ which is finite in size; therefore the number of such intervals will always be finite as well (and thus the notion of "smallest" is well-defined).

Undesired behaviors about extrema. Although what discussed above allows one to find behaviors that comply with the regularity constraint $\chi^{\phi}_{\bullet}$ to within some tolerance (given by the choice of ϵ), it is also clear that, however small ϵ is, one can always find behaviors that do not comply with $\chi^{\phi}_{\bullet}$ but nonetheless are uniformly continuous. Figure 9.4 gives a graphical representation of this fact: if b enters the interval I "just a little bit", reaches a maximum at $\hat{t} < \epsilon$, and exits it afterward, it can do so within less than any fixed δ while still having bounded derivative. So, uniform continuity is not precisely a characterization of the requirement expressed by $\chi^{\phi}_{\bullet}$, but only within some arbitrarily small tolerance, given by the choice of ϵ.

Sets of behaviors. One more thing should be said about the use of the notion of uniform continuity to characterize behaviors satisfying $\chi^{\phi}_{\bullet}$. So far, we have only considered the uniform continuity of a *single* behavior b. However, one would like to derive similar properties about a whole *set of behaviors* $B = \{b_i\}$. In general, even if each $b_i \in B$ is uniformly continuous,

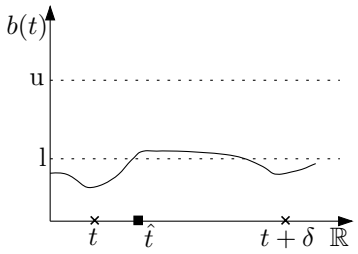

Figure 9.4.: A behavior violating $\chi_{\bullet}^{\phi}$ about an extremum.

the δ_ϵ for b_i may also depend on i, so it is actually a $\delta_{i,\epsilon}$. If, for a given ϵ, $\inf_i \delta_{i,\epsilon}$ is zero, then it is not possible to find a unique sampling period δ such that all samplings of any behavior in B with period δ satisfy $\chi_{\bullet}^{\phi}$ (up to the chosen tolerance ϵ). Therefore, we are going to include explicitly that such an infimum is greater than zero, to rule out undesirable cases.

Quasi-Non-Berkeley behaviors. All in all, according to what we have discussed above and generalizing, we introduce the following terminology. Let m be a basic time-dependent item mapping to the set $D \equiv D_1 \times \cdots \times D_n$, such that every D_i is a continuous set. Let b be a behavior mapping $\mathbb{R}$ to the set D. If b is uniformly continuous in $\mathbb{R}$, then we say that b is *quasi-non-Berkeley*.

A set $B = \{b_i\}$ of behaviors is quasi-non-Berkeley if each $b_i \in B$ is uniformly continuous, and, for all $\epsilon > 0$, the infimum $\inf_i \delta_i(\epsilon)$ of the δ_i's (given by the uniform continuity requirement for each b_i), is strictly greater than zero.

Clearly, a non-Berkeley behavior is *a fortiori* a quasi-non-Berkeley one. However, we have outlined above behaviors that are compatible with being quasi-non-Berkeley but are not fully non-Berkeley (i.e., they fail to satisfy $\chi_{\bullet}^{\phi}$).

Also notice that we are unable to express the quasi-non-Berkeley requirement in $^{\mathbb{R}}_{\mathbb{Z}}$TRIO as we did for the non-Berkeley requirement. In the former case we were interested in the orthogonal concern of giving a mathematical characterization of the behaviors, rather than one easily expressible in temporal logic.

Continuity, Uniform Continuity, and Analyticity

Gargantini and Morzenti discussed in [GM01] how the non-Zeno requirement for dense-valued items can be mathematically characterized by the notion of analyticity. We know that, for discrete-valued items, the quasi-non-Berkeley requirement is strictly stronger than the non-Zeno requirement: while the former requires that the lower bound on the constancy intervals of the items is strictly positive, the latter just requires that intervals of infinitesimal length do not accumulate.

In this vein, Appendix D.5 makes a brief mathematical detour to show that uniform continuity and analyticity are *orthogonal notions*.

Uniform continuity over finite time. Let us also recall a classic result of mathematical analysis which goes under the name of *Heine-Cantor theorem*. The theorem states that every continuous function defined over a compact (metric) domain is also uniformly continuous over the same domain. In particular, over the real line every closed interval is a compact set, thus every continuous function is also uniformly continuous over a closed interval. This means that if we restrict ourselves to behavior over *finite time*, and we ensure that the chosen finite interval is also closed, then it is sufficient to require that our behaviors are continuous functions to meet the quasi-non-Berkeley requirement.

9.4. A Comparison With Digitization

Henzinger, Manna and Pnueli [HMP92] were the first ones — to the best of our knowledge — to study explicitly the continuous- and discrete-time semantics of timed systems in general, and temporal logic in particular. Their results are based on the notion of *digitization*. This section briefly reports the main results of [HMP92] and relate them to our results about sampling invariance.

Informal comparison. Let us start with an informal comparison between the notions of digitization and of sampling invariance. Digitizability is similar to our sampling invariance in that they both define a notion of invariance between discrete- and dense-time interpretations of the same formula.

However, they differ in other aspects. First, [HMP92] compares analog- and digital-clock models, that is behaviors consisting of a *discrete* sequences of events, each carrying a timestamp: in analog-clock behaviors the timestamp is a real value, whereas in digital-clock ones it is an integer value. On the contrary, our work considers truly continuous-time behaviors and encompasses the description of dense-valued items; these features are both needed to faithfully model continuous physical quantities. Clearly, the introduction of the behavior constraint χ does limit in practice the dynamics of the items, in order to render them ultimately discretizable and thus equivalent to a discrete sequence; nonetheless, the possibility of modeling dense-valued items is unaffected, and we believe that the explicit (syntactic) introduction of the constraint χ — required to achieve invariance — permits a clearer understanding of what we "lose" exactly by discretization, and makes it possible, whenever needed, a comparison with the unconstrained continuous-time model.

The other major difference between our approach and the one presented in [HMP92] concerns the physical motivation for the notion of invariance. In fact, sampling invariance models — albeit in an abstract and idealized way — the sampling process which really occurs in systems composed by a digital controller interacting with a physical environment. On the contrary, digitization is based on the idea of "shifting" the timestamps of the analog model to make them coincide with integer values; informally, we may say that behaviors are "stretched", without changing the relative order of the events. This notion allows for a neater statement of the results about digitization; however it is more oriented towards mathematical and verification results, and less to physical reasons.

Technical differences. There are two main technical differences between our approach and the one in [HMP92].

- First, while the framework in [HMP92] adopts a point-based semantics based on (timed) trace (called *timed state sequence* in [HMP92]), ours refers to total functions from time to the state domain (i.e., behaviors). In practice, since it is customary to restrict to non-Zeno behaviors, we can equivalently say that our framework has interval-based sequences as its semantic structure [GM01, HR04].

- Second, while sampling invariance is defined for a *formula*, digitization is defined for a *property*, that is a set of timed traces. In other

words, digitization is defined independently of any language, whereas sampling invariance is defined with respect to the metric temporal logic $\frac{\mathbb{R}}{\mathbb{Z}}$TRIO.

Under this respect, the next Section 9.4.1 introduces the framework in [HMP92], while Section 9.4.2 suggests how it can be compared to ours. Afterward, Section 9.4.3 carries out the comparison under those lines, and shows that the two notions of sampling invariance and digitization are *orthogonal*. Finally, Section 9.4.4 sketches other aspects that have not been touched upon in the comparison, and may belong to future work.

9.4.1. Timed Traces and Digitization

Let us consider a generic timed trace; while timed traces are usually defined as mono-infinite sequences, our framework refers to bi-infinite time domains. Therefore, we change the definitions of [HMP92] to refer them to bi-infinite sequences. It is routine to check that all the relevant results are retained in the new setting. Moreover, we could have equivalently rephrased all our framework in terms of mono-infinite sequences, obtaining basically the same results in the comparison.

Timed traces. So, let us consider the generic timed trace:

$$\rho: \quad \cdots(\sigma_{-2},\tau_{-2})(\sigma_{-1},\tau_{-1})(\sigma_0,\tau_0)(\sigma_1,\tau_1)(\sigma_2,\tau_2)\cdots$$

where $\tau_i \in \mathbb{R}$ for all $i \in \mathbb{Z}$. Following [HMP92], we assume *weak monotonicity* of the timestamps (i.e., $\tau_i \leq \tau_{i+1}$ for all $i \in \mathbb{Z}$), and *progress* (i.e., for all $t \in \mathbb{R}$ there exists a $i \in \mathbb{Z}$ such that $\tau_i \geq t$).

ϵ-digitizations. For any $0 \leq \epsilon < 1$, and any timed trace ρ, the *ϵ-digitization* $[\rho]_\epsilon$ is the integer-timed trace that results from rounding, with respect to ϵ, every timestamp in ρ to an integer value. Namely:

$$[\rho]_\epsilon: \quad \cdots(\sigma_{-2},[\tau_{-2}]_\epsilon)(\sigma_{-1},[\tau_{-1}]_\epsilon)(\sigma_0,[\tau_0]_\epsilon)(\sigma_1,[\tau_1]_\epsilon)(\sigma_2,[\tau_2]_\epsilon)\cdots$$

where:

$$[x]_\epsilon \quad = \quad \begin{cases} \lfloor x \rfloor & \text{if } x \leq \lfloor x \rfloor + \epsilon \\ \lceil x \rceil & \text{otherwise} \end{cases}$$

Digitization of a property. Given a property π, that is a set of timed traces, its *digitization* $[\pi]$ is a property containing all and only integer-timed traces resulting from digitizing all traces in π with respect to all fractional values $0 \leq \epsilon < 1$:

$$[\pi] \quad = \quad \{[\rho]_\epsilon \mid \rho \in \pi \text{ and } 0 \leq \epsilon < 1\}$$

Digitizable properties. Given a property π, we say that:

- π is *closed under digitization* iff $[\pi] \subseteq \pi$ (or, equivalently, for all timed traces ρ: $\rho \in \pi$ implies $[\rho] \subseteq \pi$);
- π is *closed under inverse digitization* iff, for all timed traces ρ, $[\rho] \subseteq \pi$ implies $\rho \in \pi$;
- π is *digitizable* iff it is closed under both digitization and inverse digitization (or, equivalently, for all timed traces ρ: $\rho \in \pi$ iff $[\rho] \subseteq \pi$).

9.4.2. Digitizable Formulas

Let us now suggest a way to compare the two frameworks of sampling invariance and digitizability. First of all, let us define an $^{\mathbb{R}}_{\mathbb{Z}}$TRIO semantics over timed traces, that mirrors as closely as possible that over behaviors. Second, let us give a definition of what it means for an $^{\mathbb{R}}_{\mathbb{Z}}$TRIO *formula* to be digitizable.

Timed trace semantics of $^{\mathbb{R}}_{\mathbb{Z}}$TRIO. Let us define the satisfiability relation for $^{\mathbb{R}}_{\mathbb{Z}}$TRIO formulas and a generic timed trace ρ. Let $i \in \mathbb{Z}$ be a generic position in the trace. Then:[9]

$\rho_i \models \xi$	iff	$\xi \in \sigma_i$
$\rho_i \models \text{Until}_{I\rangle}(\phi_1, \phi_2)$	iff	there exists $j \geq i$ such that: $\tau_j \in \tau_i + I$, $\rho_j \models \phi_2$, and for all integers $k \in [i, j\rangle$: $\rho_k \models \phi_1$
$\rho_i \models \text{Since}_{I\langle}(\phi_1, \phi_2)$	iff	there exists $j \leq i$ such that: $\tau_j \in \tau_i - I$, $\rho_j \models \phi_2$, and for all integers $k \in \langle j, i]$: $\rho_k \models \phi_1$
$\rho_i \models \neg\phi$	iff	$\rho_i \not\models \phi$
$\rho_i \models \phi_1 \wedge \phi_2$	iff	$\rho_i \models \phi_1$ and $\rho_i \models \phi_2$
$\rho \models \phi$	iff	for all $i \in \mathbb{Z}$: $\rho_i \models \phi$

[9] Sums between a real value and an interval are meant to be Minkovsky sums, with the familiar meaning.

Digitizable formulas. Given an $^{\mathbb{R}}_{\mathbb{Z}}$TRIO formula ϕ, we say that:

- ϕ is *closed under digitization* iff the property $\{\rho \mid \rho \models \phi\}$ is closed under digitization;
- ϕ is *closed under inverse digitization* iff the property $\{\rho \mid \rho \models \phi\}$ is closed under inverse digitization;
- ϕ is *digitizable* iff the property $\{\rho \mid \rho \models \phi\}$ is digitizable.

9.4.3. Digitization and Sampling Invariance Are Orthogonal Notions

This section shows that the two notions of digitization and sampling invariance define incomparable classes of formulas, when the comparison is carried out along the lines introduced in the previous sections. To this end, first of all we exhibit a formula which is sampling invariant but not digitizable; then, we provide a formula which is digitizable but not sampling invariant.

We remark that, throughout this section, we always refer to the notion of sampling invariance according to the adaptation functions $\eta_\delta^{\mathbb{R}}\{\}, \eta_\delta^{\mathbb{Z}}\{\}$ as we defined them in Section 8.2.3. Exploring the consequences of adopting different definitions of adaptation functions in the comparison with digitization is out of the scope of the present work.

Sampling Invariant Non-Digitizable Formulas

It is simple to find sampling invariant non-digitizable formulas, as they can be build directly out of some examples of non-digitizable formulas given in [HMP92]. In particular, let us consider the formula:

$$\Theta \quad \equiv \quad \mathrm{Som}(\mathrm{WithinF}_{\mathrm{ie}}(\xi, 1))$$

It is simple to check that Θ is sampling invariant, since the outer existential quantification on time (i.e., the Som operator) does not affect sampling invariance.

On the other hand, Θ is not closed under digitization. The timed trace:

$$\rho' \quad = \quad \cdots(\neg\xi, k)(\xi, k+\mu)(\neg\xi, \tau)\cdots$$

with $k \in \mathbb{Z}$, $0 < \mu < 1$, $\tau > k+1$, and such that ξ is false at any other position in the trace, provides a proof of this (in the form of a counterexample). In fact, ρ' satisfies Θ, but any of its ϵ-digitizations with respect to

any $\epsilon < \mu$ does not, as they all correspond to $\cdots(\neg\xi, k)(\xi, k+1)(\neg\xi, \tau')\cdots$ with $\tau' \geq k+1$.

Digitizable Non-Sampling Invariant Formulas

Let us consider the following $^{\mathbb{R}}_{\mathbb{Z}}$TRIO formula (also put in normal form).

$$\begin{aligned}\Upsilon \quad \equiv \quad & \mathrm{Som}\big(\xi \wedge \neg \mathrm{Until}_{(0,+\infty))}(\xi, \mathrm{true})\big) \\ & \equiv \mathrm{Som}\big(\xi \wedge \mathrm{Releases}_{(0,+\infty))}(\neg\xi, \mathrm{false})\big)\end{aligned}$$

Let us also build the the formula:

$$\Omega \quad \equiv \quad \Upsilon \wedge \phi^{\mathrm{ns}}$$

where ϕ^{ns} was defined in Section 9.3.1. We now show that Ω is not sampling invariant, but it is digitizable.

Ω is digitizable. Let us write out explicitly the semantics of Υ, when interpreted over timed traces according to the weak-time semantics defined beforehand. For a timed trace $\rho = (\sigma, \tau)$: $\rho \models \Upsilon$ iff there exists a $i \in \mathbb{Z}$ such that:

1. $\sigma_i \models \xi$, and, for all $j \geq i$:
2. either $\sigma_j \not\models \mathrm{true}$ and $\tau_j > \tau_i$, which however never holds,
3. or $\sigma_k \not\models \xi$, for some $i \leq k < j$.

Now, since condition 3 must hold for any $j \geq i$, in particular (for $j = i+1$) it requires that $\sigma_i \not\models \xi$. This is in contradiction with condition 1, therefore $\Upsilon \equiv \mathrm{false}$ in the timed trace semantics.

Thus also $\Omega \equiv \Upsilon \wedge \phi^{\mathrm{ns}} \equiv \mathrm{false} \wedge \phi^{\mathrm{ns}} \equiv \mathrm{false}$. The formula false is trivially closed under digitization, since so is the empty set of timed traces $\emptyset$. Moreover, it is also trivially closed under inverse digitization, therefore Ω is digitizable.

Ω is not sampling invariant. Let us now consider the adaptation of Υ:

$$\eta^{\mathbb{R}}_{\delta}\{\Upsilon\} \quad \equiv \quad \mathrm{Som}\big(\xi \wedge \mathrm{Releases}_{(0,+\infty)]}(\neg\xi, \mathrm{false})\big)$$

It is not difficult to realize that, thanks to the adaptation of the Releases subformula to the] bracket, $\eta_\delta^{\mathbb{R}}\{\Upsilon\}$ is satisfiable in discrete time.[10]

Then, let us consider the behavior b in Figure 9.5, which is derived from the one in Figure 9.3 by additionally assuming that ξ holds at t and at $t+\mu$ as well. (In the figure, crosses now denote sampling instants). Now,

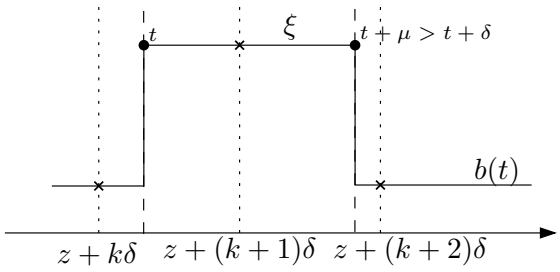

Figure 9.5.: Proof of non sampling invariance.

clearly $b(t+\mu) \models_{\mathbb{R}} \xi \wedge \text{Releases}_{(0,+\infty))}(\neg\xi, \text{false})$, and $b(t) \models_{\mathbb{R}} \text{Lasts}_{\text{ii}}(\xi, \delta)$, therefore $b \models_{\mathbb{R}} \Omega \wedge \chi_\circ$. However, while $\sigma_{\delta,z}[b] \models_{\mathbb{Z}} \eta_\delta^{\mathbb{R}}\{\Upsilon\}$, we have shown in Section 9.3.1 that $\sigma_{\delta,z}[b] \not\models_{\mathbb{Z}} \eta_\delta^{\mathbb{R}}\{\phi^{\text{ns}}\}$. Therefore $\sigma_{\delta,z}[b] \not\models_{\mathbb{Z}} \Omega$, and thus Ω is not closed under sampling, and thus *a fortiori* it is not sampling invariant.

9.4.4. Other Aspects for Comparisons

Let us now briefly hint at other aspects that may pertain to a comparison between the notions of sampling invariance and digitizability, but which have not been addressed here. We present them in a rather informal way.

- A feature that distinguishes [HMP92]'s semantic model is that it is *weakly monotonic*. This introduces some peculiarities, such as a predicate which can be both true and false for some timestamp value τ, but still at different positions in the timed trace. In other words, weak monotonicity really exposes the existence of "two times", one being the position in the timed word, the other being the value of the timestamp.

[10] Therefore, the particular definition of $\eta_\delta^{\mathbb{R}}\{\}$ plays an important role here. As usual, we do not discuss (for the sake of brevity) the impact of choosing different definitions for $\eta_\delta^{\mathbb{R}}\{\}$.

An analysis of digitization may therefore consider what are the features which are peculiar to weak monotonicity, and what would change by switching to a strongly monotonic time.

- More generally, we already discussed how digitization is an intrinsically *semantic* notion, while sampling invariance is defined for formulas. Indeed, [HMP92]'s analysis of digitizable formulas is limited to two simple classes of formulas, namely bounded response and bounded invariance properties, which basically represent the notion of upper and lower bound on a response, respectively.

  Therefore, we may extend the analysis of digitizability to seek a wider characterization of digitizable formulas. Conversely, we may also provide a semantic counterpart to sampling invariance, and carry out the comparison in a semantic setting.

- A feature of our framework is that it changes interval bounds in formulas, in particular by *weakening formulas* when passing from discrete to continuous in order to preserve entailment. [HMP92] also introduces a notion of weakening (and strengthening) of formulas, although it uses it for different purposes (namely to outline verification techniques).

  We may see if [HMP92]'s notion of weakening can be used in a similar manner as our adaptation, and compare the resulting framework.

- Notice that digitization always rounds timestamp values to the nearest *integer value*; in other words, it approximates to within a precision of one. It would be straightforward, though, to generalize the definition to other fixed values, with a finer grain than one unit. This has been done in other works (see Appendix E). It seems that such changes would not affect our comparison in any "qualitative" way, as changing this just amount to a rescaling of timing informations.

- More generally, we notice that the timed trace semantics has *several peculiar* (and, somewhat, unintuitive [HR04]) *features* that make a comparison with other semantic models more difficult and problematic. Among other things (and see Appendix E for references):

  – metric temporal logics over timed trace models are strictly *less expressive* than over interval-based sequences; therefore, one

may suggest to restrict comparisons to properties expressible in both models;

- a straightforward way to describe a timed trace as an interval-based sequence is to introduce *isolated events* at any timestamp, separated by "no-action" intervals where all predicates are false; notice that such sequences would all fail to satisfy the regularity constraint χ and thus would be trivially sampling invariant;
- otherwise, one could represent interval-based sequences as timed traces with "samplings" taken to within some precision (and, possibly, according to the property at hand). Notice in particular that the granularity of such "samplings" would depend in general on the constants used in the formulas, if one seeks to preserve entailment. Possibly, the granularity should also be related to the regularity of the interval-based sequence (as mandated by χ).

9.5. Formalizing Timed Automata in $^{\mathbb{R}}_{\mathbb{Z}}$TRIO

It is well-known from the literature that automata languages cannot be defined using "standard" propositional temporal logics [Wol83], as the latter lack the ability to "count", which is instead a feature of finite-state automata [MP72]. In particular, more recent work has shown that there are languages accepted by timed automata [AD94] that cannot be defined by any metric temporal logic formula. See Appendix E for more extensive references about these facts.

This expressiveness gap notwithstanding, it is still possible to pursue a *dual language* approach [FMMR07, FMM94] for automata and temporal logic. This consists basically in describing a system through a combination of an operational formalism and a descriptive formalism: in our case, timed automata are the operational formalism and metric temporal logics (and $^{\mathbb{R}}_{\mathbb{Z}}$TRIO in particular) are its descriptive counterpart. The dual language approach then requires to formalize the behavior of the automata through logic formulas, so that one can prove properties about the automaton by logic deduction (rather than, e.g., algorithmic methods). Moreover, if we have a formalization of timed automata in $^{\mathbb{R}}_{\mathbb{Z}}$TRIO, it is possible to exploit a notion of discretization on the description of the automata, by applying the results about sampling invariance to the formalization. This section shows how to formalize the behavior of timed automata in $^{\mathbb{R}}_{\mathbb{Z}}$TRIO.

We stress the fact that the aforementioned expressiveness results about "counting" do not prevent one from performing such a formalization: even if, in general, one cannot define the language accepted by a timed automaton through $\frac{\mathbb{R}}{\mathbb{Z}}$TRIO formulas, it is still possible to characterize the *runs* of the automaton through logic formulas. That is, one includes in the formalization the state of the automaton, and how it evolves in response to inputs.[11]

9.5.1. Timed Automata Definition

Let us recall the definition of timed automata; all of them are from the original paper by Alur and Dill [AD94].

Syntax of Timed Automata

First, we consider the definition of *clock constraints*, then we define timed automata themselves.

Clock constraints. For a set X of clock variables, the set $\Phi(X)$ of *clock constraints* ξ is defined inductively by

$$\xi ::= x \leq c \mid c \leq x \mid \neg\xi \mid \xi_1 \wedge \xi_2$$

where x is a clock in X and c is a constant in $\mathbb{Q}$.

Timed automaton. A *timed automaton* A is a tuple $\langle \Sigma, S, S_0, C, E \rangle$, where:

- Σ is a finite alphabet,
- $S = \{s_0, s_1, \ldots, s_{N_s-1}\}$ is a finite set of N_s states,
- $S_0 \subseteq S$ is a finite set of start states,
- C is a finite set of clocks, and

[11] We remark the fact that one could then switch to the formalization of the accepted language itself if the logic had a *projection* (a.k.a. hiding or existential quantification) operator (see Appendix E), which is equivalent to saying that "a logic with projection can count". $\frac{\mathbb{R}}{\mathbb{Z}}$TRIO is not endowed with projection, and therefore we will limit ourselves to the formalization of automata runs.

- $E \subseteq S \times S \times \Sigma \times 2^C \times \Phi(C)$ gives the set of transitions.
  An edge $\langle s, s', a, \Lambda, \xi \rangle$ represents a transition from state s to state s' on input symbol a. The set $\Lambda \subseteq C$ gives the clocks to be reset with this transition, and ξ is a clock constraint over C.

Simplifications. For simplicity, let us restrict all the automata we consider to avoid self-loops among the transitions; i.e., $\langle s, s, a, \Lambda, \xi \rangle \notin E$ for all $s \in S$. For our purposes, this restriction is without loss of generality, as it can be shown that any timed automaton with self-loops can be transformed into one that recognizes the same language but does not use any self-loops. Moreover, let us avoid the use of ϵ-moves, which can however be rendered by augmenting the input alphabet by a special symbol.

Semantics of Timed Automata

In order to define the semantics of timed automata, we recall the definition of timed words and timed languages. Notice that we have some differences with respect to the definition in Section 9.4.1; in the remainder we will reference to the following definitions only.

Timed words. A *timed word* over an alphabet Σ is a pair of sequences (σ, τ). σ is an infinite word $\sigma = \sigma_1\sigma_2\sigma_3\cdots$ over Σ. τ is a *time sequence*, that is an infinite sequence $\tau = \tau_1\tau_2\tau_3\cdots$ of time values $\tau_i \in \mathbb{R}_{\geq 0}$; the sequence must be strictly monotonic (that is $\tau_i < \tau_{i+1}$ for all $i \geq 1$) and divergent (that is, for all $r \in \mathbb{R}_{\geq 0}$ there exists a $i \geq 1$ such that $\tau_i > r$).

Timed languages. A *timed language* is defined as a set of timed words.

Run of a timed automaton. To define the semantics of timed automata as acceptors of timed languages, we introduce the notion of *run* of a timed automaton. A *run* of a timed automaton $\langle \Sigma, S, S_0, C, E \rangle$ over a timed word (σ, τ) is an infinite sequence of the form:

$$(s_0, \nu_0) \xrightarrow{\sigma_1,\tau_1} (s_1, \nu_1) \xrightarrow{\sigma_2,\tau_2} (s_2, \nu_2) \xrightarrow{\sigma_3,\tau_3} \cdots$$

where $s_i \in S$ and $\nu_i \in [C \to \mathbb{R}]$, for all $i \geq 0$, satisfying the following requirements:

- *Initiation*: $s_0 \in S_0$ and $\nu_0(x) = 0$ for all $x \in C$.

- *Consecution*: for all $i \geq 1$, there is an edge in E of the form $(s_{i-1}, s_i, \sigma_1, \Lambda_i, \xi_i)$ such that $\nu_{i-1} + \tau_i - \tau_{i-1}$ satisfies ξ_i and ν_i equals $[\Lambda_i \mapsto 0](\nu_{i-1} + \tau_i - \tau_{i-1})$.

Language accepted by a timed automaton. Consequently, given a timed automaton A, and a set $F \subseteq S$ of accepting states, we say that A accepts the timed word (σ, τ) iff the timed word has a run that visits some state in F infinitely often (Büchi acceptance condition). Finally, a timed automaton A accepts the language identified by all and only timed words accepted by A.

Berkeley Behaviors for Timed Automata

Since the ultimate goal of this section is to provide a formalization that allows us to harness the discretization results about sampling invariance over languages defined by timed automata, we would like not only to formalize timed automata in $\frac{\mathbb{R}}{\mathbb{Z}}$TRIO, but also to formalize them in a way which complies with the requirements of sampling invariance. Namely, we seek a formalization over slowly-variable behaviors (according to χ), and with nesting-free formulas.

Under this respect, the present subsection extends the notion of non-Berkeley behaviors to timed automata. This allows us to identify exactly those timed words that are representable by non-Berkeley behaviors.

Zeno behaviors and divergence. First of all, let us recall that Zeno behaviors are those where the time values in τ, although ever increasing, converge to a finite value $\overline{\tau}$: time "stops" at the value $\overline{\tau}$. The requirement that timestamps sequences in timed words be *divergent* is assumed precisely to rule out Zeno timed words.

Berkeley behaviors and finite difference time. As shown in [AD94, Example 3.16], and first explicitly discussed in [CHR02], in timed automata "infinitely fast" Berkeley behaviors can manifest themselves even if we rule out Zeno phenomena. Let us discuss an example of this fact with some detail.

An example of Berkeley timed automaton. Let us consider the timed automaton of Figure 9.6, where $k \in \mathbb{R}_{>0}$ is some positive constant. Let

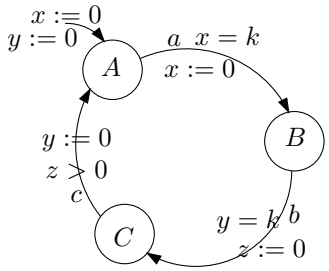

Figure 9.6.: A timed automaton with Berkeley behavior.

$x_0 = 0$ and $y_0 = 0$ be the initial values of the clocks x and y. Then, for all $i \geq 1$, let x_i be the values of the same clocks when entering the state A after the ith iteration of the loop. Let $\alpha_i, \beta_i, \gamma_i$ be the times spent sitting in the states A, B, C, respectively, during the ith iteration of the loop. This notation corresponds to the timed word:

$$(a, \alpha_1)\,(b, \alpha_1 + \beta_1)\,(c, \alpha_1 + \beta_1 + \gamma_1)\,\cdots$$
$$(a, \alpha_n + \sum_{i=1}^{n-1}(\alpha_i + \beta_i + \gamma_i))\,(b, \alpha_n + \beta_n + \sum_{i=1}^{n-1}(\alpha_i + \beta_i + \gamma_i))$$
$$(c, \alpha_n + \beta_n + \gamma_n + \sum_{i=1}^{n-1}(\alpha_i + \beta_i + \gamma_i))\,\cdots$$

Let us notice immediately that it must be $x_i + \alpha_{i+1} = k$, because of the guard on transition a, that is $k - \alpha_{i+1} = x_i$. Now, because of the guard on transition b, it must also be $\alpha_{i+1} + \beta_{i+1} = k$ (notice that y is reset when entering A). Finally, we have $x_{i+1} = \beta_{i+1} + \gamma_{i+1}$, since the clock x is last reset when entering B.

Thus, from the last equation, we get $x_{i+1} = \beta_{i+1} + \gamma_{i+1} = (k - \alpha_{i+1}) + \gamma_{i+1} = x_i + \gamma_{i+1}$. This holds for all i, thus we can iteratively substitute and get: $x_{i+1} = x_i + \gamma_{i+1} = (x_{i-1} + \gamma_i) + \gamma_{i+1} = \cdots = x_0 + \sum_{j=1}^{i+1} \gamma_j = \sum_{j=1}^{i+1} \gamma_j$.

Finally, let us notice that it must be $x_i \leq k$ for all i, otherwise the a transition cannot be taken. So we must have $\sum_{j=1}^{\infty} \gamma_j \leq k$, that is the residence times γ_i must get arbitrarily close to zero, while not being zero, in order to satisfy the constraint on the z clock.

Berkeley timed words and languages. Such kind of behaviors correspond to timed words where the difference between adjacent timestamps get ar-

bitrarily small, even if time does not accumulate and stop. More precisely, we call *Berkeley* a timed word $(\sigma, \tau) = (\sigma_1, \tau_1)(\sigma_2, \tau_2) \cdots$ where the lower bound of the differences between adjacent timestamps $\lambda = \inf_{i \geq 1}(\tau_{i+1} - \tau_i)$ is zero, and *non-Berkeley* if it equals instead some $\theta \in \mathbb{R}_{>0}$. Similarly, a set of timed words $(\sigma, \tau)_{i \in I}$ (where $I \subseteq \mathbb{N}$) is Berkeley iff the lower bound $\inf_{i \in I} \lambda_i$ of the lower bounds of the timed words is zero. Consequently, a timed automaton is Berkeley iff the timed language it accepts is Berkeley.

In the rest of this section, when discussing the correspondence between timed words and timed behaviors in $^{\mathbb{R}}_{\mathbb{Z}}$TRIO, we assume that all timed languages are non-Berkeley, unless otherwise explicitly stated. We leave to future work the analysis of how ruling out Berkeley behaviors impacts the expressiveness of timed automata.

9.5.2. Axiomatization of Timed Automata with $^{\mathbb{R}}_{\mathbb{Z}}$TRIO

Let us now consider how to describe in $^{\mathbb{R}}_{\mathbb{Z}}$TRIO all the runs of a given timed automaton. For simplicity, let us not consider acceptance conditions, that is let us assume that all states are accepting. Note, however, that introducing acceptance conditions (e.g., Büchi, Muller, etc) in the formalization would be routine.

First of all, since timed automata are interpreted over timed words, whereas $^{\mathbb{R}}_{\mathbb{Z}}$TRIO formulas are interpreted over behaviors $b : \mathbb{T} \to D$, we have to relate the two semantic models. Remember that we want to be able to consider *constrained* behaviors, i.e., regular according to the constraint $\chi_\circ$ for some fixed sampling period δ. Therefore, let us pick $\delta < \theta/2$, where θ is the lower bound of the transition times for the automaton. The reason for taking θ halved will be clear in the remainder.

Timed Words and Timed Behaviors

Let us consider a timed automaton $A = \langle \Sigma, S, S_0, C, E \rangle$; it is interpreted over timed words of the form $\langle \sigma, \tau \rangle$, where $\sigma \subseteq \Sigma^\omega$.

To map runs (over timed words) to behaviors, we have first to choose a set of basic *time-dependent items* for the $^{\mathbb{R}}_{\mathbb{Z}}$TRIO description. More precisely, we consider the following items.

- a time-dependent item st taking values to the set of states S;
- a time-dependent item in taking values to the alphabet set $\Sigma \cup \{\epsilon\}$;[12]

[12]Notice that ϵ does not represent ϵ-moves but simply absence of input.

- a time-dependent predicate $\mathsf{rs}()$ having the set of clocks C as domain;
- a time-dependent (Boolean) time-dependent predicate start.

Here it is the intuitive meaning of these items.

- $\mathsf{st} = s_i$ means that the automaton is in state $s_i \in S$;
- $\mathsf{in} = \sigma$ means that the symbol $\sigma \in \Sigma$ is currently being inputted (or no current input, in the case of ϵ);
- $\mathsf{rs}(x)$ means that the clock $x \in C$ is currently being reset;
- start represents the initialization event.

Note that this choice of items prevents multiple input symbols from arriving at the same time, whereas multiple resets (for different clocks) can happen at the same instant.

Mapping a timed word onto a timed behavior. Let us consider a run of the automaton A:

$$(s_0, \nu_0) \xrightarrow{\sigma_1,\tau_1} (s_1, \nu_1) \xrightarrow{\sigma_2,\tau_2} (s_2, \nu_2) \xrightarrow{\sigma_3,\tau_3} \cdots$$

Given a sampling period δ, we associate to the run the behavior b for the above items as follows.

- for every transition $\xrightarrow{\sigma_i,\tau_i} (s_i, \nu_i) \xrightarrow{\sigma_{i+1},\tau_{i+1}}$, the item st takes the value s_i exactly from time τ_i to time τ_{i+1}, left-continuously;
- for every input pair (σ_i, τ_i), the item in takes the value σ_i exactly from time τ_i (included) to time $\tau_i + \delta$ (included); otherwise (i.e., at any other time) in takes the value ϵ;
- for every clock $x \in C$ that is reset when taking the transition corresponding to the input pair (σ_i, τ_i), the predicate $\mathsf{rs}(x)$ is true exactly from time τ_i (included) to time $\tau_i + \delta$ (included); moreover, $\mathsf{rs}(x)$ holds from time 0 until δ (included) for every clock $x \in C$;
- start is true from time $-\infty$ to time δ (included), and false everywhere else.

Mapping a timed behavior onto a timed word. The inverse mapping can be defined, for the present purposes, as follows. Simply, given a timed behavior b, if there exists a timed word (σ, τ) such that the latter maps to the former according to the previously defined mapping, then we say that b maps to (σ, τ), otherwise the mapping of b is undefined. Notice that, when it is defined, this inverse mapping is also uniquely defined.

Axiomatization of a Timed Automaton

Let us finally consider a timed automaton $A = \langle \Sigma, S, S_0, C, E \rangle$. We now give a set of $^{\mathbb{R}}_{\mathbb{Z}}$TRIO axioms that formally describe its runs as behaviors, according to the mapping we outlined above. First, we present the axioms; afterward we justify their correctness.

In the remainder, we assume that there are no edges going from a state to itself (self-loops); this does not impacts on the expressiveness of timed automata, as it can be shown.

Translating clock constraints into formulas. We associate a $^{\mathbb{R}}_{\mathbb{Z}}$TRIO formula Ξ to every clock constraint ξ. In order to simplify this step of the formalization, we assume that all constants in the clock constraints are strictly greater than δ. It can be shown that this can always be achieved by properly scaling all constants by some suitable factor [AD94]; this corresponds to a change in the time scale. This restriction is inessential in the formalization, but we introduce it for the sake of simplicity, leaving to future work the full discussion of all the possible cases.

Let us define inductively Ξ on the structure of ξ as follows:

$$\begin{array}{lcl} x \leq c & \longrightarrow & \mathrm{WithinP}_{\mathrm{ei}}(\mathsf{rs}(x), c - \delta) \\ c \leq x & \longrightarrow & \mathrm{Lasted}_{\mathrm{ee}}(\neg \mathsf{rs}(x), c - \delta) \\ \neg \xi & \longrightarrow & \neg \Xi \\ \xi_1 \wedge \xi_2 & \longrightarrow & \Xi_1 \wedge \Xi_2 \end{array}$$

Basically, Ξ translates the guard ξ by comparing the current time to the last time a reset for the clock x happened. The offset δ is introduced to take into account the fact that each reset item holds continuously for δ time units from the moment when the reset is actually triggered (to comply with the slow variability requirement $\chi_\circ$).[13]

[13] It can be shown that derived properties of composite clock constraints hold for the corresponding temporal logic formulas. For instance $x \leq x \wedge c \leq x$ corresponds to

Necessary conditions for state change. Let us state the necessary conditions that characterize a state change. For any pair of states $s_i, s_j \in S$ such that $\langle s_i, s_j, \sigma, \Lambda, \xi \rangle \in E$ for some σ, $\Lambda = c_1, \ldots, c_r$, ξ, we introduce the axiom:[14]

$$\begin{aligned} &\mathrm{UpToNow}(\mathsf{st} = s_i) \wedge \mathrm{NowOn}(\mathsf{st} = s_j) \\ &\qquad \Rightarrow \quad \mathrm{NowOn}(\mathsf{in} = \sigma) \wedge \mathrm{NowOn}(\mathsf{rs}(c_1) \wedge \cdots \wedge \mathsf{rs}(c_r)) \wedge \Xi \end{aligned} \tag{9.12}$$

Sufficient conditions for state change. We have multiple sufficient condition for state changes; basically, they account for reactions to reading input symbols and resetting clocks. Let us consider input first: for each input symbol $\sigma \in \Sigma$, let us consider all the relations of the form $\langle s_i^k, s_j^k, \sigma, \Lambda^k, \xi^k \rangle \in E$. Then, we introduce the axiom:

$$\begin{aligned} &\mathrm{NowOn}(\mathsf{in} = \sigma) \Rightarrow \\ &\qquad \bigvee_k \left(\mathrm{UpToNow}\Big(\mathsf{st} = s_i^k\Big) \wedge \mathrm{NowOn}\Big(\mathsf{st} = s_j^k\Big) \wedge \right. \\ &\qquad\qquad \left. \mathrm{NowOn}\left(\bigwedge_{\lambda \in \Lambda^k} \mathsf{rs}(\lambda) \right) \wedge \Xi^k \right) \end{aligned} \tag{9.13}$$

This postulates the fact that, whenever a symbol σ is inputted, it arrives under conditions that are compatible with one of the transitions specified by the relation E.

Similarly, for each reset of a clock $c \in C$, let us consider all the relations of the form $\langle s_i^k, s_j^k, \sigma^k, \Lambda^k, \xi^k \rangle \in E$, such that $c \in \Lambda^k$ for all k. Then, we

$\phi = \mathrm{WithinP}_{\mathrm{ei}}(\mathsf{rs}(x), c - \delta) \wedge \mathrm{Lasted}_{\mathrm{ee}}(\neg\mathsf{rs}(x), c - \delta)$; it can be shown that if ϕ holds, then $\mathsf{rs}(x)$ last held exactly $c - \delta$ time units in the past. Therefore, the last reset has occurred exactly $c - \delta + \delta = c$ time units in the past, which corresponds to the condition $x = c$ being true, as expected.

[14] Throughout the formalization, we assume $\mathrm{NowOn}(p) = \mathrm{Lasts}_{\mathrm{ee}}(p, \delta)$, and $\mathrm{UpToNow}(p) = \mathrm{Lasted}_{\mathrm{ee}}(p, \delta)$.

introduce the axiom:

$$\begin{aligned}&\mathrm{NowOn}(\mathsf{rs}(c)) \Rightarrow \\ &\bigvee_k \Bigg(\mathrm{UpToNow}\Big(\mathsf{st} = s_i^k\Big) \wedge \mathrm{NowOn}\Big(\mathsf{st} = s_j^k\Big) \wedge \mathrm{NowOn}\Big(\mathsf{in} = \sigma^k\Big) \\ &\qquad\qquad \wedge \mathrm{NowOn}\left(\bigwedge_{\lambda \in \Lambda^k} \mathsf{rs}(\lambda)\right) \wedge \Xi^k \Bigg) \vee \mathrm{NowOn}(\mathsf{start}) \end{aligned} \tag{9.14}$$

Initialization and liveness condition. We complete our axiomatization by first describing the system initialization: at the beginning the predicate start is true for δ time units, the system is in an initial state and all the clock are reset.

$$\begin{aligned}&\mathrm{UpToNow}(\mathsf{start}) \wedge \mathrm{NowOn}(\neg\mathsf{start}) \\ &\qquad \Rightarrow \quad \mathrm{AlwP}(\mathsf{in} = \epsilon) \wedge (\exists s \in S_0 : \mathrm{AlwP}(\mathsf{st} = s)) \\ &\qquad\qquad\qquad\qquad \wedge (\forall \lambda \in C : \mathrm{AlwP}(\mathsf{rs}(\lambda))) \end{aligned} \tag{9.15}$$

Then, we have to say that start actually becomes true, sometimes, and it has been true always in the past and will be false in the future.

$$\begin{aligned}&\mathrm{Som}(\mathrm{AlwP}(\mathsf{start})) \wedge \mathrm{Som}(\neg\mathsf{start}) \\ &\qquad\qquad \wedge (\mathrm{AlwP}(\mathsf{start}) \wedge \mathrm{NowOn}(\neg\mathsf{start}) \Rightarrow \mathrm{AlwF}(\neg\mathsf{start})) \end{aligned} \tag{9.16}$$

Recall that the nesting of the temporal operator Som and AlwP does not affect sampling invariance of the resulting formula.

Finally, a "liveness" condition states that we eventually have to move out of every state (this corresponds to the fact that timed traces are made of infinite words). Thus, for every state $s_i \in S$, let $S_i \subset S$ be the set of states that are directly reachable from s_i through a single transition; then we consider the axiom:

$$\mathrm{UpToNow}(\mathsf{st} = s_i) \Rightarrow \mathrm{SomF}\left(\bigvee_{s \in S_i} \mathsf{st} = s\right) \tag{9.17}$$

Correctness and Completeness of the Axiomatization

We refer the reader to Appendix D.6 for a proof of the correctness and completeness of the above axiomatization of timed automata.

Axiomatization of Berkeley Timed Automata

Let us also discuss how it is possible to modify the above axiomatization in order to describe the behavior of any non-Zeno timed automaton, including those with Berkeley behaviors. Notice that now we forget about the requirements of the integration framework (such as the non-Berkeleyness constraint $\chi_\circ$), and we only focus to formalizing the behavior of a *generic* timed automaton with $^{\mathbb{R}}_{\mathbb{Z}}$TRIO. In this case, we give an extension of the dual language approach to deal with any timed automaton, giving up the possibility of applying our notion of discretization through sampling invariance.

To this end, it is convenient to declare the item st as a *state*, whereas all the other items are *events*, according to the definitions given in Section 2.1.4. Note that the input item would now simply be false for all $\sigma \in \Sigma$ when no character is inputted. Then, these items would be true exactly at those times when an input is received, a reset is triggered, or the system is started.

The translation of clock constraints into formulas would take into account these changes, so that the bounds of the WithinP and of the Lasted would simply be the constant c against which a clock is compared, without the δ offset.

Finally, the overall structure of the axioms would be unchanged, except that all the references to event items would now get rid of the UpToNow or NowOn operators. On the contrary, UpToNow and NowOn remains when their arguments are states; however we now have to use the $\epsilon\widetilde{\text{UpToNow}}$ and $\epsilon\widetilde{\text{NowOn}}$ versions (defined in Section 9.2.1), as states are no more guaranteed to hold over δ. For clarity, we replicate here the new versions of the axioms.

$$\begin{aligned} &\text{UpToNow}(\mathsf{st} = s_i) \wedge \text{NowOn}(\mathsf{st} = s_j) \\ &\qquad\qquad \Rightarrow \quad \mathsf{in} = \sigma \wedge (\mathsf{rs}(c_1) \wedge \cdots \wedge \mathsf{rs}(c_r)) \wedge \Xi \end{aligned} \tag{9.18}$$

$$\mathsf{in} = \sigma \Rightarrow \bigvee_k \left(\mathrm{UpToNow}\Big(\mathsf{st} = s_i^k\Big) \wedge \mathrm{NowOn}\Big(\mathsf{st} = s_j^k\Big) \wedge \left(\bigwedge_{\lambda \in \Lambda^k} \mathsf{rs}(\lambda) \right) \wedge \Xi^k \right) \tag{9.19}$$

$$\mathsf{rs}(c) \Rightarrow \bigvee_k \left(\mathrm{UpToNow}\Big(\mathsf{st} = s_i^k\Big) \wedge \mathrm{NowOn}\Big(\mathsf{st} = s_j^k\Big) \wedge \mathsf{in} = \sigma^k \wedge \left(\bigwedge_{\lambda \in \Lambda^k} \mathsf{rs}(\lambda) \right) \wedge \Xi^k \right) \vee \mathsf{start} \tag{9.20}$$

$$\mathsf{start} \Rightarrow \quad \mathrm{AlwP}(\forall \sigma \in \Sigma : \mathsf{in} \neq \sigma) \wedge (\exists s \in S_0 : \mathrm{AlwP}(\mathsf{st} = s)) \wedge (\forall \lambda \in C : \mathsf{rs}(\lambda)) \tag{9.21}$$

$$\mathrm{Som}(\mathsf{start}) \wedge (\mathsf{start} \Rightarrow \mathrm{AlwP}(\neg \mathsf{start}) \wedge \mathrm{AlwF}(\neg \mathsf{start}))$$

$$\mathrm{UpToNow}(\mathsf{st} = s_i) \Rightarrow \mathrm{SomF}\left(\bigvee_{s \in S_i} \mathsf{st} = s \right) \tag{9.22}$$

Now, note that the same correctness and completeness proofs for the non-Berkeley case hold, with some technical modifications, for the unconstrained case as well. For brevity, we omit the details.

9.5.3. An Example

Let us apply the above axiomatization on a simple example: the automaton of Figure 9.7, modeling a train. In the model, the input symbols a, i, o, e represent, respectively, the train approaching a crossing area, entering the area, exiting it, and notifying its exit. The example automaton is taken from [AD94].

It is straightforward to realize that the automaton accepts the language:

$$\{((aioe)^\omega, \tau) \mid \forall i (i \equiv 1 \bmod 4 \rightarrow \tau_{i+1} \geq \tau_i + 2 \wedge \tau_{i+3} \leq \tau_i + 5)\}$$

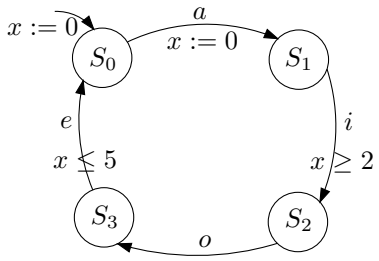

Figure 9.7.: A timed automaton modeling a train.

Let us now write out all the axioms. First of all, axiom 9.12 is implemented as the four following axioms.

$$\begin{aligned}
\mathrm{UpToNow}(\mathsf{st} = S_0) \wedge \mathrm{NowOn}(\mathsf{st} = S_1) &\Rightarrow \mathrm{NowOn}(\mathsf{in} = a) \wedge \mathrm{NowOn}(\mathsf{rs}(x)) \\
\mathrm{UpToNow}(\mathsf{st} = S_1) \wedge \mathrm{NowOn}(\mathsf{st} = S_2) & \\
\Rightarrow \mathrm{NowOn}(\mathsf{in} = i) \wedge \mathrm{Lasted}_{\mathrm{ee}}(\neg\mathsf{rs}(x), 2 - \delta) & \\
\mathrm{UpToNow}(\mathsf{st} = S_2) \wedge \mathrm{NowOn}(\mathsf{st} = S_3) &\Rightarrow \mathrm{NowOn}(\mathsf{in} = o) \\
\mathrm{UpToNow}(\mathsf{st} = S_3) \wedge \mathrm{NowOn}(\mathsf{st} = S_0) & \\
\Rightarrow \mathrm{NowOn}(\mathsf{in} = e) \wedge \mathrm{WithinP}_{\mathrm{ii}}(\mathsf{rs}(x), 5 - \delta) &
\end{aligned}$$

Let us now consider axiom 9.13, which gets to:

$$\begin{aligned}
\mathrm{NowOn}(\mathsf{in} = a) \Rightarrow \mathrm{UpToNow}(\mathsf{st} = S_0) \wedge \mathrm{NowOn}(\mathsf{st} = S_1) \wedge \mathrm{NowOn}(\mathsf{rs}(x)) & \\
\mathrm{NowOn}(\mathsf{in} = i) & \\
\Rightarrow \mathrm{UpToNow}(\mathsf{st} = S_1) \wedge \mathrm{NowOn}(\mathsf{st} = S_2) \wedge \mathrm{Lasted}_{\mathrm{ee}}(\neg\mathsf{rs}(x), 2 - \delta) & \\
\mathrm{NowOn}(\mathsf{in} = o) \Rightarrow \mathrm{UpToNow}(\mathsf{st} = S_2) \wedge \mathrm{NowOn}(\mathsf{st} = S_3) & \\
\mathrm{NowOn}(\mathsf{in} = e) & \\
\Rightarrow \mathrm{UpToNow}(\mathsf{st} = S_3) \wedge \mathrm{NowOn}(\mathsf{st} = S_0) \wedge \mathrm{WithinP}_{\mathrm{ii}}(\mathsf{rs}(x), 5 - \delta) &
\end{aligned}$$

Axiom 9.14 gets simply into:

$$\begin{aligned}
\mathrm{NowOn}(\mathsf{rs}(x)) \Rightarrow (\mathrm{UpToNow}(\mathsf{st} = S_0) \wedge \mathrm{NowOn}(\mathsf{st} = S_1) & \\
\wedge \mathrm{NowOn}(\mathsf{in} = a)) \ \vee \ \mathrm{NowOn}(\mathsf{start}) &
\end{aligned}$$

We do not write out explicitly the initialization conditions of axioms 9.15–9.16, as they do not depend on the actual automaton (except for the

simple condition on start states). Thus, let us just consider the liveness condition of axiom 9.17; this is rendered by the following four axioms.

$$\begin{aligned}
&\mathrm{UpToNow}(\mathsf{st} = S_0) \Rightarrow \mathrm{SomF}(S_1)\\
&\mathrm{UpToNow}(\mathsf{st} = S_1) \Rightarrow \mathrm{SomF}(S_2)\\
&\mathrm{UpToNow}(\mathsf{st} = S_2) \Rightarrow \mathrm{SomF}(S_3)\\
&\mathrm{UpToNow}(\mathsf{st} = S_3) \Rightarrow \mathrm{SomF}(S_0)
\end{aligned}$$

This concludes the axiomatization for the present example.

10. Conclusions

This thesis focused on the problem of improving the scalability in the use of formal methods towards large and heterogeneous systems. In particular, emphasis has been put on real-time systems, where time is a fundamental dimension of analysis, and a further source of complexity. Our reference language was the metric temporal logic TRIO, which we presented in Chapter 2. We pursued the descriptive approach, where both the specification and the requirements of a system are formalized as a set of temporal logic formulas.

Compositionality. Our solution was grounded in the well-known software engineering principles of modularization and abstraction. When these techniques are applied to formal methods, they usually come under the name *compositionality*. Therefore, compositionality was the topic of Part I of this thesis. In Chapter 3, we defined precisely the notion of compositionality. In particular, we discussed how it consists of both techniques and methods to specify and verify modular systems.

Compositional *techniques* usually boil down to the availability of compositional inference rules. A compositional inference rule allows one to infer facts that are global to a composite systems from the composition of facts that are local to the various modules out of which the system is built. In Chapter 3 we also provided the generic components of a "typical" compositional inference rule, and we discussed various features that a rule may have or not have. In particular, we linked the features of a rule to the features of a system, whose correctness one verifies through application of the rule.

On the other hand, compositional *methods* provide frameworks within which a compositional inference rule can be applied. Moreover, they give general guidelines according to which one can manage modularization both when writing the specification and when building the verification of a system.

Chapter 4 reviewed the most significant literature about compositionality in formal methods. The review showed that most frameworks provided

in the literature are focused mostly on the technical aspects of compositionality, and in particular on the study of compositional inference rules. We claimed that this is impractical, and therefore we advocated a more broad and versatile approach to compositionality.

This approach was developed in Chapter 5. There we introduced an array of techniques, each suited for verifying certain classes of systems better than others. Our array of inference rules developed along the dimensions presented in Chapter 3. In particular, we provided both circular and non-circular rules: although the literature deals mostly with the technical complications introduced by the former, we maintained that most actual systems do not exhibit circular dependencies among modules, and therefore non-circular rules are often sufficient. Another technical issue which is often given great importance in the literature is the *completeness* of inference rule. We provided a completeness analysis for each rule we have introduced, but we also rediscussed the significance of completeness for the practical use of compositionality, claiming that it is not always an indispensable feature.

Together with the compositional inference rules, Chapter 5 also introduced a *general methodology* to formalize and verify modular systems. The methodology exploits TRIO's modular features at the specification level, and it combines them with compositional and general deductive verification techniques to provide support at the system verification level. Our methodology can be regarded as a *lightweight* approach to compositionality, where it is the system under development that guides and suggests the techniques and methods to be used, rather than the opposite.

Chapter 6 demonstrated our approach on two illustrative examples: a real-time version of the dining philosophers problem, and a peer-to-peer communication protocol. Although not industrial-strength case studies, we believe that the two examples are sufficiently large and complicated to assess, to some extent, the feasibility of our approach, and the advantages of using such an approach to compositionality in reducing the burden of verification.

Integration. The natural generalization of the compositional problem occurs when we consider composite systems whose modules are specified by means of different formalisms, and according to different semantic models. For such systems, the key problem is to integrate the various modules within a global semantics, so that it is possible to verify the overall system.

Under this respect, Part II of the thesis dealt with the problem of *integration*, that is composing modules written using different languages and semantics.

The focus of this thesis being on real-time systems, we directed our analysis on a particular, but very common, case of integration, namely the integration of continuous-time modules with discrete-time modules. As we discussed in Chapter 7, systems where continuous-time and discrete-time dynamics coexist are called *hybrid*; they are of great interest as most control systems are naturally described as hybrid systems.

We developed our solution to the integration problem in Chapter 8. The basic ingredient of our framework was a formal relation between discrete-time and continuous-time semantics that abstracts what happens in a system where the discrete-time component has access to an ideal sampling of the visible variables of the continuous-time components. Our relation was called *sampling*, and temporal logic formulas consistent with respect to sampling were said to be *sampling invariant*. The basic result of Chapter 8 was then the identification of a subset of the TRIO language whose formulas are sampling invariance. This subset was named $^{\mathbb{R}}_{\mathbb{Z}}$TRIO and formally defined in the same chapter. Moreover, sampling invariance also required to restrict the interpretations of $^{\mathbb{R}}_{\mathbb{Z}}$TRIO formulas to suitably regular behaviors; more precisely, in order not to lose any "information" through sampling, we required that the variability of a behavior is sufficiently slow.

The following Chapter 9 provided a precise analysis of several aspects of our integration framework. In particular, we dealt with three fundamental problems. First, we investigated the expressiveness of $^{\mathbb{R}}_{\mathbb{Z}}$TRIO, relating it to the MTL temporal logic. Second, we gave alternative characterizations of the variability restriction on behaviors introduced in the framework; in particular, we showed how it could be related to the mathematical notion of uniform continuity. Third, we compared our approach to integration with another widespread notion of discretization, introduced by Henzinger, Manna, and Pnueli [HMP92], called digitization. Finally, Chapter 9 also showed how $^{\mathbb{R}}_{\mathbb{Z}}$TRIO could be used in practice to formalize the behavior of another popular formalism for timed systems, namely timed automata.

The expressiveness analysis and the comparison with digitization performed in Chapter 9 showed several subtleties that influence the expressiveness of formalisms close to $^{\mathbb{R}}_{\mathbb{Z}}$TRIO in expressive power. Correspondingly, Appendix E reviewed related work about the expressiveness of metric temporal logic languages and automata. This collocated more precisely $^{\mathbb{R}}_{\mathbb{Z}}$TRIO

within several other formalisms with similar features, and it allowed us to see to what extent our results about integration can be moved to other formal languages.

We believe that the results on compositionality and integration provided in this thesis improve the feasibility in — some aspects of — the use of formal methods on large and heterogeneous systems, in a way which is coherent with the well-known and effective practices of modularization and separation of concerns of "classical" software-engineering.

Future work. Since the overall goal of this thesis was to apply formal methods to large and heterogeneous systems, an important topic for future work is to concretely use the solutions we provided on some industrial-strength applications. Indeed, the "technology transfer" from research results to industrial practice is something that is always advocated but very rarely — if ever — done in practice. From this point of view, this thesis was no exception, thus we stress the importance of this further development for the future.

Another important aspect to ameliorate the practical use of formal methods is the availability of tools. Tool development was not the focus of this thesis; nonetheless, all our techniques should be integrated in the TRIO IDE environment, currently under full development (see Section 2.4). In particular, not only our compositional techniques, but also our compositional methods should be given adequate tool support from IDE tools.

As we discussed in the incipit of Part II, integration encompasses a very broad number of problems. Although we focused on timing aspects in this thesis, other aspects of integration deserve investigation. Among many, let us mention the problem of integrating static complex data within timed systems.

More specifically with reference to the integration framework we provided, there are at least two major directions for future work. First, our notion of discretization allows for the discrete-time verification of continuous-time properties. This was not the primary goal of our integration effort, but it is an interesting by-product. On the other hand, discrete-time verification was the focus of the digitization approach, which we reviewed in this thesis and compared to ours. Therefore, it would be interesting to see what practical advantages our notion of integration brings to automated verification through the use of discretization techniques.

Refinement is a second application area for our notion of integration, and in particular the refinement of abstract specifications into implementations. In this sense, one would start from a high-level specification that deals with continuous time, and then he/she would refine it to refer to a more concrete view of implementable time, namely a discrete one; our integration framework may help in this respect. We already developed some of the aspects of this problem elsewhere [SFR+06, FRS+06]; future work should bring the results about integration we provided in this thesis to within a more general approach to the development and refinement of specifications.

A. The Dining Philosophers

A.1. TRIO Specification of the philosopher Class

class philosopher (**const** t_e, **const** T_e, **const** T_t)

signature:

visible: start, eating, holding, available, thinking;

temporal domain: $\mathbb{R}$;

domains: $\mathrm{S} : \{\mathrm{l}, \mathrm{r}\}$;

items:

event start, take(S), release(S);

state eating, holding(S), available(S), thinking, hungry;

formulae:

vars: $s : \mathrm{S}, t : \mathbb{R}$;

axioms:

holding_synch: $\mathsf{holding}(\mathrm{l}) \Leftrightarrow \mathsf{holding}(\mathrm{r})$;

hungry: $\mathrm{Lasted}(\neg\mathsf{eating}, T_t) \Leftrightarrow \mathsf{hungry}$;

acquire: $\mathsf{hungry} \wedge \mathrm{UpToNow}(\mathsf{available}(\mathrm{l}) \wedge \mathsf{available}(\mathrm{r}))$
$\Rightarrow (\mathsf{take}(\mathrm{l}) \wedge \mathsf{take}(\mathrm{r})) \vee \mathrm{NowOn}(\neg\mathsf{available}(\mathrm{l}) \vee \neg\mathsf{available}(\mathrm{r}))$;

eat_till_release: $\mathrm{Becomes}(\mathsf{holding}(\mathrm{l}) \wedge \mathsf{holding}(\mathrm{r}))$
$\Rightarrow (\exists t > t_e : \mathrm{Lasts}(\mathsf{eating}, t) \wedge \mathrm{Futr}(\mathsf{release}(\mathrm{l}) \wedge \mathsf{release}(\mathrm{r}), t))$;

eating_duration: $\mathrm{Lasted}(\mathsf{eating}, t) \Rightarrow t < T_e$;

eating_def: $\mathsf{holding}(\mathrm{l}) \wedge \mathsf{holding}(\mathrm{r}) \Leftrightarrow \mathsf{eating}$;

thinking_def: $\mathsf{thinking} \Leftrightarrow \neg\mathsf{eating}$;

thinking_duration: $\mathrm{Becomes}(\mathsf{thinking}) \Rightarrow \mathrm{Lasts}(\mathsf{thinking}, T_t)$;

taking: $\mathsf{take}(s) \wedge \mathrm{UpToNow}(\neg\mathsf{holding}(s)) \wedge \mathrm{UpToNow}(\mathsf{available}(s))$
$\Rightarrow \mathrm{Until}(\mathsf{holding}(s), \mathsf{release}(s))$;

putting: $\mathsf{release}(s) \wedge \mathrm{UpToNow}(\mathsf{holding}(s))$
$\Rightarrow \mathrm{Until}(\neg\mathsf{holding}(s), \mathsf{take}(s))$;

assumptions:

TCCs: $T_e > t_e > 0 \ \wedge \ T_t > 2T_e$;

availability: $(\exists t \geq T_t : \text{Lasts}(\mathsf{available}(s), t))$
$\vee \ \text{WithinF}_{\text{ei}}(\text{Becomes}(\mathsf{available}(s)), T_t + T_e)$;

availability_2: $\text{WithinF}(\text{UpToNow}(\mathsf{available}(s)), T_e)$;

lasting_availability: $\text{Becomes}(\mathsf{available}(s))$
$\Rightarrow \text{Lasts}(\mathsf{available}(s), T_t)$;

theorems:

taking_turns: $(\exists t \geq T_t : \text{Lasts}(\neg\mathsf{holding}(s), t))$
$\vee \ \text{WithinF}_{\text{ei}}(\text{Becomes}(\neg\mathsf{holding}(s)), T_t + T_e)$;

taking_turns_2: $\text{WithinF}(\text{UpToNow}(\neg\mathsf{holding}(s)), T_e)$;

fork_availability:
$\text{WithinF}_{\text{ei}}(\text{UpToNow}(\mathsf{available}(\text{l}) \wedge \mathsf{available}(\text{r})), T_t + 2T_e)$;

always_eating_or_not: hungry
$\vee \ \text{WithinP}_{\text{ii}}((\exists t > t_e : \text{Lasted}(\mathsf{eating}, t))$
$\vee \ \text{Becomes}(\mathsf{eating}), T_t + T_e)$;

regular_eating_rg:
$\text{WithinF}_{\text{ei}}(\text{UpToNow}(\mathsf{available}(\text{l}) \wedge \mathsf{available}(\text{r})), T_t + 2T_e)$
$\gg$
$(\text{SomP}_{\text{i}}(\mathsf{start}) \Rightarrow (\exists t > t_e : \text{Within}_{\text{ii}}(\text{Lasts}(\mathsf{eating}, t), T_t + 2T_e)))$;

end

A.2. TRIO Specification of the dining_N Class

class dining_N (**const** t_e, **const** T_e, **const** T_t, **const** N)

import: philosopher;

signature:

visible: start;

temporal domain: $\mathbb{R}$;

items: **event** start;

modules:
Philosophers: **array** $[0..N-1]$ **of** philosopher
$[t_e$ **is const** t_e, T_e **is const** T_e, T_t **is const** $T_t]$;
connections:
vars: $i : [0..N-1]$;
(**direct** Philosophers$[i]$.available(l) ,
$\neg$Philosophers$[(i-1) \mod N]$.holding(r)),
(**direct** Philosophers$[i]$.available(r) ,
$\neg$Philosophers$[(i+1) \mod N]$.holding(l)),
(**broadcast** start, Philosophers.start);
formulae:
vars: $k : [0..N-1], t : \mathbb{R}$;
assumptions:
TCCs: $T_e > t_e > 0 \ \wedge \ T_t > 2T_e \ \wedge \ N \geq 2$;
theorems:
liveness_rg: $\forall k \in [0..N-1] : (\ \mathrm{SomP_i}(\mathrm{Philosophers}[k].\mathsf{start})$
$\Rightarrow \ (\exists t > t_e : \mathrm{Within_{ii}}(\mathrm{Lasts}(\mathrm{Philosophers}[k].\mathsf{eating}, t)\ , T_t + 2T_e)))$;
end

A.3. Proofs of the philosopher Class

In this section we provide proofs for all the theorems of the **philosopher** class not proved in the main body of this thesis.

Theorem 53 (**philosopher.taking_turns**).
$(\exists t \geq \mathrm{T_t} : \mathrm{Lasts}(\neg\mathsf{holding}(s)\ , t))$
$\vee \ \mathrm{WithinF_{ei}}(\mathrm{Becomes}(\neg\mathsf{holding}(s))\ , \mathrm{T_t} + \mathrm{T_e})$

Proof. Let us take $t = \mathrm{T_t}$ and show that:

$$\mathrm{Lasts}(\neg\mathsf{holding}(s)\ , \mathrm{T_t}) \vee \mathrm{WithinF_{ei}}(\mathrm{Becomes}(\neg\mathsf{holding}(s))\ , \mathrm{T_t} + \mathrm{T_e})$$

for a generic s. The proof proceeds by contradiction: assume the negation of the goal, that is, after some simple rewriting, $\mathrm{WithinF}(\mathsf{holding}(s)\ , \mathrm{T_t})$ and $\mathrm{Lasts_{ei}}(\neg\mathrm{Becomes}(\neg\mathsf{holding}(s))\ , \mathrm{T_t} + \mathrm{T_e})$.

Let us consider the first term $\mathrm{WithinF}(\mathsf{holding}(s)\ , \mathrm{T_t})$; by the definition of the WithinF operator, this is the same as saying that there is a time instant $0 < u < \mathrm{T_t}$ in the future in which $\mathsf{holding}(s)$ is true. It is simple to infer, by axioms **holding_synch** and **eating_def**, that eating is true at the same instant u.

We have briefly highlighted the fact that eating holds on right-continuous intervals. It can be shown that, for any right-continuous state S and for any time distance d, if $S \wedge \text{Lasts}(\neg\text{Becomes}(\neg S), d)$ is true at a certain time, then it follows that $\text{Lasts}(S, d)$ at the same time (intuitively, this is simple to understand). Therefore, consider the state eating and the distance T_e. We have just shown that at time u eating is true. Furthermore, our initial assumption of the proof by contradiction $\text{Lasts}_{\text{ei}}(\neg\text{Becomes}(\neg\text{holding}(s)), \text{T}_\text{t} + \text{T}_\text{e})$ implies that $\text{Lasts}(\neg\text{Becomes}(\neg\text{holding}(s)), \text{T}_\text{e})$ at time u, being $\text{T}_\text{e} + u < \text{T}_\text{e} + \text{T}_\text{t}$. Again, by combining axioms **holding_synch** and **eating_def**, this last statement implies $\text{Lasts}(\neg\text{Becomes}(\neg\text{eating}), \text{T}_\text{e})$. Hence, we deduce that $\text{Lasts}(\text{eating}, \text{T}_\text{e})$ holds at time u.

Equivalently, this last fact can be stated at time $u + \text{T}_\text{e}$ as $\text{Lasted}(\text{eating}, \text{T}_\text{e})$. It is immediate to derive a contradiction with the statement of axiom **eating_duration**. □

Theorem 54 (**philosopher.taking_turns_2**).
$\text{WithinF}(\text{UpToNow}(\neg\text{holding}(s)), \text{T}_\text{e})$

Proof. By axiom **eating_duration**, we can assume that it is false that $\text{Lasted}(\text{eating}, \text{T}_\text{e})$ at T_e time instants in the future or, equivalently, that $\text{Lasts}(\text{eating}, \text{T}_\text{e})$ is false at the current time. By the definition of the Lasts operator, this is equivalent to assuming that $\text{WithinF}(\neg\text{eating}, \text{T}_\text{e})$ is *true* at the current time.

Let $0 < u < \text{T}_\text{e}$ be the time instant in the future at which eating is false. Since eating is a right-continuous state, it follows that $\text{NowOn}(\neg\text{eating})$ is true at u. More explicitly, these facts can be stated as $\text{Lasts}_{\text{ie}}(\neg\text{eating}, v)$ at u, for some $v > 0$.

Now, consider any time instant after u, before $u + v$ and before T_e. For example, take the time instant $w = u + \min(\text{T}_\text{e} - u, v)/2$ (clearly, $w < u + v$ and $w < \text{T}_\text{e}$ by construction). We realize that $\text{UpToNow}(\neg\text{eating})$ holds at w, since it is true that $\text{Lasted}(\neg\text{eating}, \min(\text{T}_\text{e} - u, v)/2)$ holds at w.

Finally, by considering axioms **holding_synch** and **eating_def**, we deduce the statement $\text{UpToNow}(\neg\text{holding}(s))$ at $w < \text{T}_\text{e}$, which concludes the proof. □

Theorem 55 (**philosopher.fork_availability**).
$\text{WithinF}_{\text{ei}}(\text{UpToNow}(\text{available(l)} \wedge \text{available(r)}), \text{T}_\text{t} + 2\text{T}_\text{e})$

Proof. Let us consider two instances of assumption **availability**, one for the left fork and one for the right fork, and fix $t = \text{T}_\text{t}$. According to the disjunction, the proof can be split into four cases, whether:

1. $\mathrm{Lasts}(\mathsf{available}(l), \mathrm{T_t})$ and $\mathrm{Lasts}(\mathsf{available}(r), \mathrm{T_t})$
2. $\mathrm{Lasts}(\mathsf{available}(l), \mathrm{T_t})$ and $\mathrm{WithinF_{ei}}(\mathrm{Becomes}(\mathsf{available}(r)), \mathrm{T_t} + \mathrm{T_e})$
3. $\mathrm{Lasts}(\mathsf{available}(r), \mathrm{T_t})$ and $\mathrm{WithinF_{ei}}(\mathrm{Becomes}(\mathsf{available}(l)), \mathrm{T_t} + \mathrm{T_e})$
4. $\mathrm{WithinF_{ei}}(\mathrm{Becomes}(\mathsf{available}(l)), \mathrm{T_t} + \mathrm{T_e})$ and $\mathrm{WithinF_{ei}}(\mathrm{Becomes}(\mathsf{available}(r)), \mathrm{T_t} + \mathrm{T_e})$,

all formulas holding at the current time.

Let us first consider the simpler case 1. Let us consider the time instant $u = \mathrm{T_t}/2$ in the future. We claim that $\mathrm{UpToNow}(\mathsf{available}(l) \wedge \mathsf{available}(r))$ holds at u; since $u < \mathrm{T_t} + 2\mathrm{T_e}$ this would conclude this part of the proof. In fact, obviously $\mathrm{Lasts}(\mathsf{available}(l) \wedge \mathsf{available}(r), \mathrm{T_t}/2)$ holds, being an immediate consequence of the hypotheses. Equivalently, this can be stated as $\mathrm{Lasted}(\mathsf{available}(l) \wedge \mathsf{available}(r), \mathrm{T_t}/2)$ at time u. By considering the definition of the UpToNow operator, this implies $\mathrm{UpToNow}(\mathsf{available}(l) \wedge \mathsf{available}(r))$ holds at u, what we just claimed.

Let us now pass to case 2 and just assume $\mathrm{WithinF_{ei}}(\mathrm{Becomes}(\mathsf{available}(r)), \mathrm{T_t} + \mathrm{T_e})$ at the current time. Now, let $0 < u < \mathrm{T_t} + \mathrm{T_e}$ be the time instant at which $\mathrm{Becomes}(\mathsf{available}(r))$ holds. By assumption **lasting_availability**, it follows that $\mathrm{Lasts}(\mathsf{available}(r), \mathrm{T_t})$ also holds at u. Now, let us consider the other assumption **availability_2**, taking u as the base time and l as fork: therefore we assume that $\mathrm{WithinF}(\mathrm{UpToNow}(\mathsf{available}(l)), \mathrm{T_e})$ holds at u. Let $u+v < u+\mathrm{T_e}$ be the time instant at which $\mathrm{UpToNow}(\mathsf{available}(l))$ holds. Since $\mathrm{T_e} < \mathrm{T_t}/2$, it follows that $u + v < u + \mathrm{T_t}$ and therefore *a fortiori* $\mathrm{Lasts}(\mathsf{available}(r), v)$ holds at u. This last formula implies that $\mathrm{UpToNow}(\mathsf{available}(r))$ holds at $u + v$. All in all, $\mathrm{UpToNow}(\mathsf{available}(l) \wedge \mathsf{available}(r))$ holds at $u+v$. Now, since $u+v < (\mathrm{T_t}+\mathrm{T_e})+\mathrm{T_e} = \mathrm{T_t}+2\mathrm{T_e}$, this branch of the proof is successfully concluded.

Case 3 has a proof which is all similar to case 2, because of the symmetry between the left and right forks.

Finally, case 4 can be reduced to case 2 or 3: in fact in both of them we did not use the hypothesis $\mathrm{Lasts}(\mathsf{available}(s), \mathrm{T_t})$ in the proof. □

Theorem 56 (**philosopher.always_eating_or_not**). $\mathsf{hungry} \vee \mathrm{WithinP_{ii}}((\exists t > \mathrm{t_e} : \mathrm{Lasted}(\mathsf{eating}, t)) \vee \mathrm{Becomes}(\mathsf{eating}), \mathrm{T_t} + \mathrm{T_e})$

Proof. In order to prove the disjunction, we assume the negation of the first term as hypothesis. Therefore, we assume $\neg\mathrm{Lasted}(\neg\mathsf{eating}, \mathrm{T_t})$ at the current time or, equivalently, $\mathrm{WithinP}(\mathsf{eating}, \mathrm{T_t})$. Writing the WithinP

out explicitly, this means that there exists a time distance $0 < u < \mathrm{T_t}$ such that eating holds at $-u$.

Being eating a right-continuous state, we also have that NowOn(eating) at $-u$, that is Lasts(eating, v) at $-u$, for some positive time v. Now, consider the time instant $-w = -u + \min(u, v)/2$. Clearly $-u < -w < 0$ and $-w < -u + v$, by construction. Therefore, we can characterize the behavior of eating at $-w$ by noticing that the state is true before and after $-w$, that is UpToNow(eating) and NowOn(eating) at $-w$.

Now, for any right-continuous state predicate S such that NowOn(S), it can be proved[1] that either S stays true always in the future (i.e., AlwF(S)), or there is a future time instant p at which S becomes false, and before that time it stays true (i.e., $\exists p > 0 : \mathrm{Lasts}(S, p) \wedge \mathrm{Futr}(\mathrm{NowOn}(\neg S), p)$). A similar result holds for any right-continuous state predicate S such that UpToNow(S): either S stays true always in the past (i.e., AlwP(S)), or there is a past time instant $-p$ at which S becomes false, and before that time it stays true (i.e., $\exists p > 0 : \mathrm{Lasted}(S, p) \wedge \mathrm{Past}(\mathrm{NowOn}(\neg S) \vee \mathrm{UpToNow}(\neg S), p)$).[2] Therefore, for eating, we can combine the two properties in the past and in the future to split the proof into four cases.

1. AlwF(eating) and AlwP(eating);
2. Lasts(eating, p)$\wedge$Futr(NowOn($\neg$eating), p) and AlwP(eating), for some $p > 0$;
3. AlwF(eating) and Lasted(eating, p) $\wedge$ Past(NowOn($\neg$eating) $\vee$ UpToNow($\neg$eating), p), for some $p > 0$;
4. Lasts(eating, p_f) $\wedge$ Futr(NowOn($\neg$eating), p_f) and Lasted(eating, p_p)$\wedge$Past(NowOn($\neg$eating) $\vee$ UpToNow($\neg$eating), p_p), for some $p_f, p_p > 0$;

all of them evaluated at time $-w$.

Branch 1 is simple, since we have that eating always holds (note that eating is true exactly at $-w$ as well, since Lasts(eating, v) holds at $-u$, and $-w < -u + v$).

Let us consider branch 2. From AlwP(eating) at $-w$, *a fortiori* Lasted(eating, $\mathrm{T_e}$) at $-w$. Since $\mathrm{T_e} > t_e$ and $w < u < \mathrm{T_t} < \mathrm{T_t} + \mathrm{T_e}$, clearly

[1] A weaker but similar result holds for generic time-dependent predicates, but we do not need it here. We do not discuss neither proof.

[2] The asymmetry with the future case is due to the fact that S is *right*-continuous.

$\text{WithinP}_{\text{ii}}(\exists t > t_e : \text{Lasted}(\mathsf{eating}, t)\,, \text{T}_\text{t} + \text{T}_\text{e})$ is satisfied, thus concluding this branch.

Now for branch 3. Let us first try the case in which UpToNow($\neg$eating) at $-w-p$ and Lasted(eating, p) at $-w$. If $p > \text{T}_\text{e}$, it is true that Lasted(eating, T_e) at $-w > -(\text{T}_\text{t} + \text{T}_\text{e})$, so the goal $\text{WithinP}_{\text{ii}}(\exists t > t_e : \text{Lasted}(\mathsf{eating}, t)\,, \text{T}_\text{t} + \text{T}_\text{e})$ is satisfied. On the contrary, if $p \leq \text{T}_\text{e}$, we can show that Becomes(eating) holds at $-w-p$. In fact, we know that UpToNow($\neg$eating) at $-w-p$. Moreover, it is not hard to realize that Lasted(eating, T_e) at $-w$ implies that NowOn(eating) at $-w-p$, since $p \leq \text{T}_\text{e}$. Therefore UpToNow($\neg$eating)$\wedge$NowOn(eating) at $-w-p > -(u+\text{T}_\text{e}) > -(\text{T}_\text{t}+\text{T}_\text{e})$, which is the definition of Becomes(eating).

We are now considering the case in which NowOn($\neg$eating) at $-w-p$ and Lasted(eating, p) at $-w$. This case derives a contradiction, since it is both NowOn($\neg$eating) and NowOn(eating) at $-w-p$. In fact, the latter follows from Lasted(eating, p) at $-w$, which is equivalent to Lasts(eating, p) at $-w-p$. So, branch 3 is concluded.

Finally, branch 4 can be reduced to step 3, where we did not use the additional hypothesis AlwF(eating). □

B. The BitTorrent Example Verification

B.1. Proof of Proposition 6.3.1

Proposition B.1.1. *If, for some fixed duration* $\mathrm{T_B} > 0$*:*

1. *for all* $i \in \mathcal{I}_N$*:* $E_i \Rightarrow \mathrm{WithinF}(M_i, \mathrm{T_B})$
2. *for all* $i \in \mathcal{I}_N$*:* $E \wedge M_i \Rightarrow E_{\texttt{if } i<N \texttt{ then } i+1 \texttt{ else } 1}$
3. $\bigwedge_{i \in \mathcal{I}_N} \mathrm{WithinF}(M_i, N\mathrm{T_B}) \Rightarrow M$
4. $\mathrm{WithinF}(E_i, \mathrm{T_B})$*, for some* $i \in \mathcal{I}_N$

then $\mathrm{Lasts}(E, N\mathrm{T_B}) \Rightarrow M$*.*

Proof. Let us assume that $\mathrm{Lasts}(E, N\mathrm{T_B})$ holds at the current time t, that is E holds throughout the interval $(t, t + N\mathrm{T_B})$. By (4), there exists a $t < t_1 < t + \mathrm{T_B}$ such that E_i holds at t' for some i. Without loss of generality, let $i = N$: this amounts just to a re-numbering of the modules.

Next, let us show that there exists N instants $t_1, t_2, \ldots, t_N$ such that M_i holds at t_i, and $t < t_i < t + i\mathrm{T_B}$, for all $i \in \mathcal{I}_N$. The proof goes by (finite) induction. The base case $i = 1$ is obvious: since E_N at t_1, then also M_1 at t_1 by (2), and $t < t_1 < t + \mathrm{T_B}$ by assumption.

For the inductive step, assume that M_{i-1} holds at $t < t_{i-1} < t+(i-1)\mathrm{T_B}$; notice that E holds at t_{i-1} as well. Therefore, E_i holds at t_{i-1} from (2). Let us evaluate (1) at t_{i-1}; it follows that there exists a $t_{i-1} < t_i < t_{i-1} + \mathrm{T_B}$ such that M_i holds at t_i. Given the bounds on t_{i-1}, it is clear that $t < t_{i-1} < t_i < t_{i-1} + \mathrm{T_B} < t + (i-1)\mathrm{T_B} + \mathrm{T_B} = t + i\mathrm{T_B}$. This concludes the induction.

All in all we have shown that $\bigwedge_{i \in \mathcal{I}_N} \mathrm{WithinF}(M_i, N\mathrm{T_B})$ holds at t. (3) lets us conclude that M holds at t. □

B.2. Proof of Lemmas 6.3.2–6.3.4

Notice that, in proving each lemma, we can safely assume that we are in a time interval where $\mathrm{Lasts}_{\mathrm{ie}}(E^k, kT_s)$ holds, that is where the connections are constant. In fact, the conclusion of the whole compositional proof holds under the condition $\mathrm{Lasts}_{\mathrm{ie}}(E^k)\, kT_s$. Now, for the proofs.

Proof of Lemma 6.3.2. For every i in a cluster of size k, this lemma is equivalent to theorem **peer.bounded_send_recv**. □

Proof of Lemma 6.3.3. Trivial, since the consequent is immediately subsumed by the antecedent. □

Modular distance and properties. The proof of Lemma 6.3.4 is more complicated than the others, and requires an additional definition for a *distance* between two peers in the cluster, together with a property of this new item. The property is not hard to prove, but it requires a somewhat involved case discussion.

Let us first define the distance between two peers p_1, p_2 in a cluster as: $\mathrm{dist}_N(p_1, p_2) \equiv (p_2 - p_1) \bmod N$. Then, we have the following property.

Lemma B.2.1.

$$\mathrm{dist}_N(p_1, \mathrm{next}_N(p_2)) = \begin{cases} 0 & \textit{if } \mathrm{next}_N(p_2) = p_1 \\ \mathrm{dist}_N(p_1, p_2) + 1 & \textit{otherwise} \end{cases}$$

Proof. Let us first consider the simple case $\mathrm{next}_N(p_2) = p_1$. We then conclude, by the definition of distance, $\mathrm{dist}_N(p_1, \mathrm{next}_N(p_2)) = \mathrm{dist}_N(p_1, p_1) = 0$.

Let us now take the other case $\mathrm{next}_N(p_2) \neq p_1$. Considering the definitions of distance and next element, we have to prove

$$(((p_2 + 1) \bmod N) - p_1) \bmod N = ((p_2 - p_1) \bmod N) + 1. \tag{B.1}$$

The proof is split into two branches, whether $p_2 + 1 < N$ or not.

1. $p_2 + 1 < N$. In this case, (B.1) rewrites to

$$((p_2 + 1) - p_1) \bmod N = ((p_2 - p_1) \bmod N) + 1 \tag{B.2}$$

Let us distinguish whether $p_2 - p_1 + 1 \geq 0$ or not.

a) $p_2 - p_1 \geq -1$. Notice that $p_2 - p_1 = -1$ iff $\text{next}_N(p_2) = p_1$, so we have $p_2 - p_1 \geq 0$ in this branch of the proof. Hence, (B.2) is established

$$p_2 - p_1 = (p_2 - p_1) \bmod N = p_2 - p_1$$

b) $p_2 - p_1 < -1$. Notice that, for any natural number $k \in [1..N-1]$, we have $-k \bmod N = N - k$. So, we can rewrite (B.2) as

$$N - (-p_2 + p_1 - 1) = (N - (-p_2 + p_1)) + 1$$

which concludes this branch of the proof.

2. $p_2 + 1 \geq N$. It is simple to realize that it must be $p_2 + 1 = N$, since $p_2 < N$. Hence, (B.1) becomes

$$(0 - p_1) \bmod N = N - p_1 = ((N - 1 - p_1) \bmod N) + 1$$

It is also obvious that $N - 1 - p_1 \geq 0$, so we finally conclude

$$N - p_1 = (N - 1 - p_1) + 1 \qquad \square$$

To simplify the notation, we also introduce a notion of distance among peers in a cluster that refers to the permutation of the indexes induced by the connections. Namely, the distance between two peers with respect to a permutation is given by the distance between their indexes in the permutation and denoted as: $\text{dist}_{\pi_N}(p_1, p_2) \equiv \text{dist}_N(\pi_N^{-1}(p_1), \pi_N^{-1}(p_2))$. Consequently, a similar notion for the next peer is introduced. That is, the next of a peer with respect to a permutation is given by the element next to the peer in the permutation, and it is denoted as: $\text{next}_{\pi_N}(p) \equiv \pi_N(\text{next}_N(\pi_N^{-1}(p)))$. Notice that Lemma B.2.1 can be shown to hold also for the functions $\text{dist}_{\pi_N}(p_1, p_2)$ and $\text{next}_{\pi_N}(p)$ we have just defined; we omit the straightforward proof of this fact.

Proof of Lemma 6.3.4. We still need to prove an intermediate lemma, before actually performing the proof of Lemma 6.3.4. Notice that, for the sake of brevity, we now write simply $\text{dist}(p_1, p_2)$ and $\text{next}(p)$ instead of $\text{dist}_{\widetilde{\pi}_k}(p_1, p_2)$ and $\text{next}_{\widetilde{\pi}_k}(p)$ respectively, whenever the permutation $\widetilde{\pi}_k$ is understood to be the one defined by axiom **clustering** at a given time instant.

Lemma B.2.2 (futr_recv). $\mathsf{Pe}[p_1].\mathsf{recv}(i) \Rightarrow \forall 0 < d < k_i : \forall p_2 : (\mathrm{dist}(p_1, p_2) = d \Rightarrow \exists j : \mathrm{WithinF}(\mathsf{Pe}[p_2].\mathsf{recv}(j)\,, dT_s))$

Proof. The proof goes by induction on the distance $d > 0$.

The *base case* $d = 1$ requires us to show that, for all p_2 such that $\mathrm{dist}(p_1, p_2) = 1$, $\mathsf{Pe}[p_1].\mathsf{recv}(i) \Rightarrow \exists i : \mathrm{WithinF}(\mathsf{Pe}[p_2].\mathsf{recv}(i)\,, T_s)$. Now, as a consequence of axiom **peer.recv_to_send**, it follows from $\mathsf{Pe}[p_1].\mathsf{recv}(i)$ that
$\mathrm{WithinF}(\mathsf{Pe}[p_1].\mathsf{send}(j)\,, T_s)$ for some packet j. It is not hard to realize that, if $\mathrm{dist}(p_1, p_2) = 1$, then $p_2 = \mathrm{next}(p_1)$. Moreover, we are considering a time interval in the future where the connections are stable, so in particular we have that $\mathsf{connected_p}(p_1, p_2)$, that is $\mathsf{Pe}[p_1].\mathsf{send}(j)$ is equivalent to $\mathsf{Pe}[p_2].\mathsf{recv}(j)$. In other words, we have shown that $\mathrm{WithinF}(\mathsf{Pe}[p_2].\mathsf{recv}(j)\,, T_s)$, concluding the base step.

Now, let us consider a $d > 1$. The *inductive step* requires us to assume that the property holds for d and to show that it holds for $d + 1$ as a consequence. So, let us consider two generic peers such that $\mathrm{dist}(p_1, p_2) = d+1$, and assume that $\mathsf{Pe}[p_1].\mathsf{recv}(i)$ for some packet i. Let us also consider the peer p_3 such that $\mathrm{next}(p_3) = p_2$, that is the peer immediately preceding p_2 in the ring of connections. We can apply Lemma B.2.1 to show that either $\mathrm{dist}(p_1, \mathrm{next}(p_3)) = \mathrm{dist}(p_1, p_2) = 0$ or $\mathrm{dist}(p_1, p_2) = \mathrm{dist}(p_1, p_3)+1$. However, it is not the case that $p_2 = p_1$, otherwise it would be $\mathrm{dist}(p_1, p_2) = 0 < 1$. Hence $\mathrm{dist}(p_1, p_3) = d$.

The inductive step lets us imply that $\exists j : \mathrm{WithinF}(\mathsf{Pe}[p_3].\mathsf{recv}(j)\,, dT_s)$, assuming $\mathsf{Pe}[p_1].\mathsf{recv}(i)$. Considering the definition of the WithinF operator, and assuming 0 as the current time, this means that there exists a time instant $0 < t < dT_s$ time units in the future,when $\mathsf{Pe}[p_3].\mathsf{recv}(j)$ holds. With a proof similar to that of the base case, considering the fact that $\mathsf{connected_p}(p_3, p_2)$ we can show that there exists a time instant q, occurring no later than T_s time units after t, when $\mathsf{Pe}[p_2].\mathsf{recv}(l)$ for some packet l. Since $q - t < T_s$, then $0 < q < (d + 1)T_s$ and therefore we can write $\mathrm{WithinF}(\mathsf{Pe}[p_3].\mathsf{recv}(l)\,, (d + 1)T_s)$, thus establishing the property for $d + 1$ and concluding the inductive proof. □

We are now ready for the final proof.

Proof of Lemma 6.3.4. We are actually going to prove $\exists i$: WithinF($\mathsf{Pe}[j].\mathsf{recv}(i)\,, kT_s)$ for all j in the cluster, which subsumes the statement of the lemma.

Let us start by noting that axiom **clustering** implies that there exist a peer p and a seed s such that $\mathsf{connected_s}(s, p)$, and the connections stays stable over the next kT_s time units, because of the usual assumption. In the remainder, let us assume 0 as the current instant. As a consequence of axiom **seed.sending** for the seed s, it is true that $\mathrm{WithinF}(\mathsf{Se}[s].\mathsf{send}(i), T_s)$ for some packet i, that is there exists a time instant $0 < t < T_s$ in the future when $\mathsf{Se}[s].\mathsf{send}(i)$ holds. Therefore, because of the connection, $\mathsf{Pe}[p].\mathsf{recv}(i)$ at the same time t.

Let us consider any generic peer j in the cluster. We can assume that $\mathrm{dist}(p, j) > 0$ without loss of generality. In fact, if $\mathrm{dist}(p, j) = 0$, then it is simple to realize that $p = j$ and therefore the proof is concluded. Otherwise, we can apply lemma **futr_recv**, substituting p, j and $\mathrm{dist}(p, j)$ for p_1, p_2 and d respectively and considering t as the base time instant. We infer that $\mathrm{Futr}(\mathrm{WithinF}(\mathsf{Pe}[j].\mathsf{recv}(l), \mathrm{dist}(p, j)T_s), t)$ for some packet l. Considering the definitions of the corresponding TRIO operators, this means that there exists a time instant $t < q < \mathrm{dist}(p, j)T_s + t$ in the future when $\mathsf{Pe}[j].\mathsf{recv}(l)$ happens. It is simple to realize that $\mathrm{dist}(p, j) < k$, since this is true of any pair of peers in a cluster of size k. So $c = \mathrm{dist}(p, j) + 1 \leq k$, $t < q < cT_s$ and $q < kT_s$. This is the same as saying that $\mathrm{WithinF}(\mathsf{Pe}[j].\mathsf{recv}(i), kT_s)$ for the generic peer j in the cluster, what we had to prove. □

C. The Reservoir Integration Example

C.1. Proofs of Theorems 41–43

Theorem 57 (**reservoir$^{\mathbb{R}}$.abstract_behavior_1**).

$$\mathrm{Lasts}_{\mathrm{ee}}(\mathsf{F}, \delta) \wedge \mathsf{l} \geq \mathrm{l} \;\Rightarrow\; \mathrm{Futr}(\mathsf{l} \geq \mathrm{l}, \delta)$$

Proof. Since L is a state item, it changes its value a finite number of times in the interval of length δ from the current instant. In each of the resulting constancy intervals, we have that either $\mathsf{F} \wedge \mathsf{L}$ or $\mathsf{F} \wedge \neg\mathsf{L}$ holds; in both cases the level of fluid must increase in the interval, by Axioms **level_behavior_1** and **level_behavior_2**, respectively. Thus, after a finite number of increases, it must be $\mathsf{l} > \mathrm{l} \geq \mathrm{l}$ after δ time instants. □

Theorem 58 (**reservoir$^{\mathbb{R}}$.abstract_behavior_2**).

$$\mathsf{l} \geq \mathrm{t} \;\Rightarrow\; \mathrm{Futr}(\mathsf{l} \geq \mathsf{l}, \delta)$$

Proof. Let l' be the value of l at δ time instants in the future, and l the current value. By Axioms 35 through 38, it is simple to understand that the level cannot go below the value $\max(\mathsf{l} - \mathrm{r_l}\delta, 0) = \mathsf{l} - \mathrm{r_l}\delta \geq \mathrm{l}$, since we are assuming that $\mathrm{t} - \mathrm{r_l}\delta > \mathrm{l}$. □

Theorem 59 (**reservoir$^{\mathbb{R}}$.abstract_behavior_3**).

$$\mathrm{Lasts}_{\mathrm{ee}}(\neg\mathsf{F} \wedge \neg\mathsf{L}, \delta) \wedge \mathsf{l} \geq \mathrm{l} \;\Rightarrow\; \mathrm{Futr}(\mathsf{l} \geq \mathrm{l}, \delta)$$

Proof. Let l' be the value of l at δ time instants in the future, and l the current value. Then, $\mathsf{l}' = \mathsf{l} \geq \mathrm{l}$ by Axiom **level_behavior_4**. □

D. Proofs for the Integration Framework

D.1. Transforming $^{\mathbb{R}}_{\mathbb{Z}}$TRIO Formulas into Normal Form

Elimination of negations. In order to eliminate negations in a formula, we just push them inward to boolean conditions. Since conditions are closed under complement (as it will be clear in their definition, see Section 8.3), and we have the dual of each basic operator, we can always eliminate negations from formulas. More explicitly, we perform the substitutions:

$$\begin{array}{lcl} \neg(\neg\phi) & \longrightarrow & \phi \\ \neg(\phi_1 \wedge \phi_2) & \longrightarrow & \neg\phi_1 \vee \neg\phi_2 \\ \neg\mathrm{Until}_{I\rangle}(\phi_1, \phi_2) & \longrightarrow & \mathrm{Releases}_{I\rangle}(\neg\phi_1, \neg\phi_2) \\ \neg\mathrm{Since}_{I\langle}(\phi_1, \phi_2) & \longrightarrow & \mathrm{Released}_{I\langle}(\neg\phi_1, \neg\phi_2) \end{array}$$

recursively, until all negations are eliminated or on conditions.

Elimination of nesting. Nesting can be eliminated from $^{\mathbb{R}}_{\mathbb{Z}}$TRIO by introducing auxiliary basic items (which are primitive conditions). More explicitly, if ξ is a primitive condition, ξ' is an auxiliary item (not used anywhere else in the specification), ϕ is a generic formula, and Op is any temporal operator (i.e., Until, Since, Releases, or Released), we perform the substitutions:

$$\begin{array}{lcl} \mathrm{Op}_{I\rangle}(\phi, \xi) & \longrightarrow & (\xi' \Leftrightarrow \phi) \ \wedge \ \mathrm{Op}_{I\rangle}(\xi', \xi) \\ \mathrm{Op}_{I\rangle}(\xi, \phi) & \longrightarrow & (\xi' \Leftrightarrow \phi) \ \wedge \ \mathrm{Op}_{I\rangle}(\xi, \xi') \end{array}$$

recursively, until nestings of temporal operators are eliminated. It is simple to check, by direct application of the definition of the operators, that the above substitutions preserve the semantics of the formula. Exactly, one can check that if $\xi' \Leftrightarrow \phi$ then $\mathrm{Op}_{I\rangle}(\phi, \xi) \equiv \mathrm{Op}_{I\rangle}(\xi', \xi)$ and $\mathrm{Op}_{I\rangle}(\xi, \phi) \equiv \mathrm{Op}_{I\rangle}(\xi, \xi')$.

Finally, one has obviously to eliminate the double implications by introducing the corresponding definitions: $\xi' \Leftrightarrow \phi \equiv (\xi' \wedge \phi) \vee (\neg\xi' \wedge \neg\phi)$; the negations which are introduced in the process are then eliminated as explained in the above step.

Elimination of $\text{Until}_{)}$ and $\text{Releases}_{]}$ in discrete time. Let us first note the following equivalences. If ξ_1, ξ_2 are any conditions, and ξ' is an auxiliary item (not used anywhere else in the specification), then:

$$\begin{aligned}
&(\xi' \Leftrightarrow \text{NowOn}(\xi_2)) &&\Rightarrow \big(\text{Until}_{\langle l,u\rangle)}(\xi_1,\xi_2) \equiv \text{Until}_{\langle l-1,u-1\rangle]}(\xi_1,\xi')\big)\\
&(\xi' \Leftrightarrow \text{UpToNow}(\xi_2)) &&\Rightarrow \big(\text{Since}_{\langle l,u\rangle(}(\xi_1,\xi_2) \equiv \text{Since}_{\langle l-1,u-1\rangle[}(\xi_1,\xi')\big)\\
&(\xi' \Leftrightarrow \text{UpToNow}(\xi_2)) &&\Rightarrow \big(\text{Releases}_{\langle l,u\rangle]}(\xi_1,\xi_2) \equiv \text{Releases}_{\langle l+1,u+1\rangle)}(\xi_1,\xi')\big)\\
&(\xi' \Leftrightarrow \text{NowOn}(\xi_2)) &&\Rightarrow \big(\text{Released}_{\langle l,u\langle[}(\xi_1,\xi_2) \equiv \text{Released}_{\langle l+1,u+1\langle(}(\xi_1,\xi')\big)
\end{aligned}$$

Let us sketch the proof of the above equivalences, for a generic discrete-time behavior $b \in \mathcal{B}_{\mathbb{Z}}$.

- *Until and Since.* Assume that $b \models_{\mathbb{Z}} \xi' \Leftrightarrow \text{NowOn}(\xi_2)$. Then, if $b(k) \models_{\mathbb{Z}} \text{Until}_{\langle l,u\rangle)}(\xi_1,\xi_2)$ for some instant k, then there exists a $d \in \langle l,u\rangle$ such that ξ_2 is true at $k+d$ and ξ_1 is true in the whole interval $[0,d)$ from k, which in discrete time is equivalent to $[0,d-1]$. Thus, for $d' = d-1$, we have that ξ' is true at $k+d'$, since ξ_2 is true at $k+d = k+d'+1$. Notice that $d' \in \langle l-1, u-1\rangle$. All in all we have that $b(k) \models_{\mathbb{Z}} \text{Until}_{\langle l-1,u-1\rangle]}(\xi_1,\xi')$.
  For the converse, assume that $b(k) \models_{\mathbb{Z}} \text{Until}_{\langle l-1,u-1\rangle]}(\xi_1,\xi')$. So, there exists a $d \in \langle l-1, u-1\rangle$ such that ξ' holds at $k+d$ and ξ_1 holds in the whole interval $[0,d]$ from k. Therefore, ξ_2 holds at $k+d+1$, and $d+1 \in \langle l,u\rangle$. Hence, for $d' = d+1$ we have that $b(k) \models_{\mathbb{Z}} \text{Until}_{I)}(\xi_1,\xi_2)$ holds.

  The proof for the Since is all similar, but for the past direction of time.

- *Releases and Released.* The equivalences follow from the above equivalence about the Until operator, exploiting the fact that the Releases is its dual, as well as the easily verifiable equivalences of $\neg\text{NowOn}(\xi)$ with $\text{NowOn}(\neg\xi)$, and of $\xi' \Leftrightarrow \text{NowOn}(\xi'')$ with $\xi'' \Leftrightarrow \text{UpToNow}(\xi')$.

  All similarly for the Released, but derived from the property of the Since.

Therefore, we perform the following substitutions to put a formula into normal form.

$$\begin{array}{ll}
\text{Until}_{\langle l,u\rangle)}(\xi_1,\xi_2) & \longrightarrow (\xi' \Leftrightarrow \text{NowOn}(\xi_2)) \ \wedge \ \text{Until}_{\langle l-1,u-1\rangle]}(\xi_1,\xi') \\
\text{Since}_{\langle l,u\rangle(}(\xi_1,\xi_2) & \longrightarrow (\xi' \Leftrightarrow \text{UpToNow}(\xi_2)) \ \wedge \ \text{Since}_{\langle l-1,u-1\rangle[}(\xi_1,\xi') \\
\text{Releases}_{\langle l,u\rangle]}(\xi_1,\xi_2) & \longrightarrow (\xi' \Leftrightarrow \text{UpToNow}(\xi_2)) \ \wedge \ \text{Releases}_{\langle l+1,u+1\rangle)}(\xi_1,\xi') \\
\text{Released}_{\langle l,u\rangle[}(\xi_1,\xi_2) & \longrightarrow (\xi' \Leftrightarrow \text{NowOn}(\xi_2)) \ \wedge \ \text{Releases}_{\langle l+1,u+1\rangle(}(\xi_1,\xi')
\end{array}$$

D.2. Proof of Theorem 8.3.1

D.2.1. Size of an Interval

In order to prove the theorem, let us first introduce the definition of size of an interval, for both time models. To this end, we introduce here the $|\cdot|_{\mathbb{T}}$ operator, which is defined as follows, according to the time model. If the time domain is dense, then:

$$|\langle l,u\rangle|_{\mathbb{R}} = \begin{cases} u-l & \text{if } u \geq l \\ 0 & \text{if } u < l \end{cases}$$

If the time domain in instead discrete, we have the following definitions:

$$\begin{array}{rcl}
|(l,u\rangle|_{\mathbb{Z}} & = & |[l+1,u\rangle|_{\mathbb{Z}} \\
|\langle l,u)|_{\mathbb{Z}} & = & |\langle l,u-1]|_{\mathbb{Z}} \\
|[l,u]|_{\mathbb{Z}} & = & \begin{cases} u-l+1 & \text{if } u \geq l \\ 0 & \text{if } u < l \end{cases}
\end{array}$$

D.2.2. Items Change Points

Let us notice the following fact that we will use in the following.

Lemma D.2.1 (Items Change Points)**.** *For any behavior obeying the constraint $\chi_\circ$, if for some time $t \in \mathbb{R}$ it is $b(t) \models_{\mathbb{R}} \langle \psi_1,\ldots,\psi_n\rangle = v$ for some v, then there exists $c_n \geq t$ and $c_p \leq t$ such that:*

- $b(c_n) \models_{\mathbb{R}} \text{Becomes}(\langle \psi_1,\ldots,\psi_n\rangle \neq v)$ *or*
  $b(t) \models_{\mathbb{R}} \text{AlwF}(\langle \psi_1,\ldots,\psi_n\rangle = v)$*;*
- $b(c_p) \models_{\mathbb{R}} \text{Becomes}(\langle \psi_1,\ldots,\psi_n\rangle = v)$ *or*
  $b(t) \models_{\mathbb{R}} \text{AlwP}(\langle \psi_1,\ldots,\psi_n\rangle = v)$

- *for all* $t' \in (c_p, c_n)$ *it is* $b(t') \models_{\mathbb{R}} \langle \psi_1, \ldots, \psi_n \rangle = v$;
- $c_n - c_p \geq \delta$.

Moreover, the same property holds for any condition ξ.

Proof. Since $\chi_\circ$ holds for b, then there exists a $p \in [0, \delta]$ such that for all $u \in [-p, -p + \delta]$ it is $b(t + u) \models_{\mathbb{R}} \langle \psi_1, \ldots, \psi_n \rangle = v$. In particular, it is $b(t - p) \models_{\mathbb{R}} \langle \psi_1, \ldots, \psi_n \rangle = v$ and $b(t - p + \delta) \models_{\mathbb{R}} \langle \psi_1, \ldots, \psi_n \rangle = v$. Since $\chi_\circ$ implies the non-Zenoness of the items values, then [GM01] there exist a $d_p, d_n \geq 0$ such that:

- either for all $u' \in [0, d_n)$ it is $b(t - p + \delta + u') \models_{\mathbb{R}} \langle \psi_1, \ldots, \psi_n \rangle = v$, and there exists an $\epsilon_n > 0$ such that for all $u'' \in (0, \epsilon_n)$ it is $b(t - p + \delta + d_n + u'') \models_{\mathbb{R}} \langle \psi_1, \ldots, \psi_n \rangle \neq v$, or $\langle \psi_1, \ldots, \psi_n \rangle = v$ always in the future;
- either for all $u' \in [0, d_p)$ it is $b(t - p - u') \models_{\mathbb{R}} \langle \psi_1, \ldots, \psi_n \rangle = v$, and there exists an $\epsilon_p > 0$ such that for all $u'' \in (0, \epsilon_n)$ it is $b(t - p - d_n - u'') \models_{\mathbb{R}} \langle \psi_1, \ldots, \psi_n \rangle \neq v$, or $\langle \psi_1, \ldots, \psi_n \rangle = v$ always in the past.

In other words, either there is a point where the items change (some of) their values, or they keep their current value indefinitely.

If the latter is the case, then the lemma follows immediately. Otherwise, notice that $(-p+\delta+d_n)-(-p) = \delta+d_n \geq \delta$, therefore we have $b(t-p+\delta+d_n) \models_{\mathbb{R}} \text{UpToNow}(\langle \psi_1, \ldots, \psi_n \rangle = v)$. Similarly, notice that $-((-p-d_p)-(-p+\delta)) = d_p+\delta \geq \delta$, therefore $b(t-p-d_p) \models_{\mathbb{R}} \text{NowOn}(\langle \psi_1, \ldots, \psi_n \rangle = v)$.

Finally, by considering condition $\chi_\circ$ again for the instants $t-p+\delta+d_n+\epsilon'$ and $t-p-d_p-\epsilon'$ "to the limit" as ϵ goes to 0, then we realize that it must also be $b(t-p+\delta+d_n) \models_{\mathbb{R}} \text{NowOn}(\langle \psi_1, \ldots, \psi_n \rangle \neq v)$ and $b(t-p-d_p) \models_{\mathbb{R}} \text{UpToNow}(\langle \psi_1, \ldots, \psi_n \rangle \neq v)$.

Therefore, for $c_n = t - p + \delta + d_n$ and $c_p = t - p - d_p$ the lemma holds.

Finally, since the value of a condition changes only if some values change their values, the lemma holds immediately for conditions as well. □

D.2.3. Theorem 8.3.1

Theorem D.2.2 (Sampling Invariance for Discrete-valued Items)**.** *Normal-form* $\frac{\mathbb{R}}{\mathbb{Z}}$*TRIO is sampling invariant, for items mapping to a* discrete *set* D*, with respect to the behavior constraint* $\chi_\circ$*, the adaptation functions* $\eta^{\mathbb{R}}_\delta\{\cdot\}$ *and* $\eta^{\mathbb{Z}}_\delta\{\cdot\}$*, for any sampling period* δ *and origin* z*.*

Proof. The proof is split into two main parts: first we show that any $^{\mathbb{R}}_{\mathbb{Z}}$TRIO formula ϕ, interpreted in the continuous-time domain, is closed under sampling; then we show that any formula ϕ, interpreted in the discrete-time domain, is closed under inverse sampling. We assume formulas written in the normal form presented above; this is without loss of generality as it does not restrict the expressiveness of the language as it was shown.

For ease of exposition, let us also introduce the following abbreviations: for a real number r, let us denote by $\Omega(r)$ the sampling instant $z + \lfloor (r - z)/\delta \rfloor \delta$, which is immediately before r, and by $O(r)$ the sampling instant $z + \lceil (r - z)/\delta \rceil \delta$ which is immediately after r. Moreover, we also denote by $\omega(r)$ and $o(r)$ the distances between r and its previous and next sampling instant, respectively, that is $\omega(r) = r - \Omega(r)$ and $o(r) = O(r) - r$. Obviously $\omega(r), o(r) \geq 0$.

(Closure under sampling).[1] Let b be a continuous-time behavior in $[\![\chi_\circ]\!]_{\mathbb{R}}$ and let $\phi' = \eta^{\mathbb{R}}_{\delta}\{\phi\}$. Then, let b' be the sampling $\sigma_{\delta,z}[b]$ of behavior b with the given origin and sampling period.

Now, for a generic sampling instant $t = z + k\delta$, we show that $b(t) \models_{\mathbb{R}} \phi$ implies $b'(k) \models_{\mathbb{Z}} \phi'$, by induction on the structure of ϕ.

- $\phi = \xi$.
  By definition of sampling of a behavior, the truth value of ξ at t and k is the same, i.e., $\xi|_{b(t)} = \xi|_{b(z+k\delta)} = \xi|_{b'(k)}$.

- $\phi = \mathrm{Until}_{\langle l,u \rangle}(\xi_1, \xi_2)$.
  Notice that ϕ' is $\mathrm{Until}_{[l',u']}(\xi_1, \xi_2)$, with $l' = \lfloor l/\delta \rfloor$ and $u' = \lceil u/\delta \rceil$.

  Let d be a time instant in $\langle l, u \rangle$ such that $b(t + d) \models_{\mathbb{R}} \xi_2$ and, for all time instants $e \in [0, d\rangle$ it is $b(t+e) \models_{\mathbb{R}} \xi_1$. Condition $\chi_\circ$ at time $t+d$ implies that there exists a $p \in [0, \delta]$ such that for all $f \in [-p, -p + \delta]$ it is $b(t + d + f) \models_{\mathbb{R}} \xi_2$. In other words, ξ_2 holds on the closed real interval $I = [t+d-p, t+d-p+\delta]$. Since I has size δ, its intersection with the sampling points (which are δ time units apart) must be non-empty. In particular, it is either $p \geq \omega(t + d)$ or $-p + \delta \geq o(t + d)$: otherwise it would be $\delta = p + (-p + \delta) < \omega(t + d) + o(t + d) = O(t + d) - \Omega(t + d) = \delta(\lceil (t + d - z)/\delta \rceil - \lfloor (t + d - z)/\delta \rfloor) \leq \delta$, a contradiction (where we exploited the property: $\lceil r \rceil - \lfloor r \rfloor \leq 1$ for any real r).

[1] This proof exploits some properties of the floor and ceiling functions. We refer the reader to [GKP94] for a thorough treatment of these functions.

So, let t' be the sampling instant:

$$t' = \begin{cases} \Omega(t+d) & \text{if } p \geq \omega(t+d) \\ O(t+d) & \text{otherwise} \end{cases}$$

It is not difficult to check that $(t'-t)/\delta \in [l', u']$. In fact:

- if $p \geq \omega(t+d)$, then $t'-t = \Omega(t+d) - t = \delta(\lfloor (k\delta+d)/\delta \rfloor - k) = \delta\lfloor d/\delta \rfloor$. Recall that $d \in \langle l, u\rangle$, and then *a fortiori* $d \in [l,u] \supseteq \langle l, u\rangle$. So $d/\delta \in [l/\delta, u/\delta]$, and $(t'-t)/\delta = \lfloor d/\delta \rfloor \in [\lfloor l/\delta \rfloor, \lfloor u/\delta \rfloor] \subseteq [l', u']$.
- if $p < \omega(t+d)$, then $t'-t = O(t+d) - t = \delta(\lceil (k\delta+d)/\delta \rceil - k) = \delta\lceil d/\delta \rceil$. Recall that $d \in \langle l, u\rangle$, and then *a fortiori* $d \in [l,u] \supseteq \langle l, u\rangle$. So $d/\delta \in [l/\delta, u/\delta]$, and $(t'-t)/\delta = \lceil d/\delta \rceil \in [\lceil l/\delta \rceil, \lceil u/\delta \rceil] \subseteq [l', u']$.

So far, we have shown that $b(t') \models_{\mathbb{R}} \xi_2$. By inductive hypothesis, it follows that for $d' = (t'-t)/\delta$ it is $b'(k+d') \models_{\mathbb{Z}} \xi_2$. Since $d' \in [l', u']$, to conclude this branch of the proof we have to show that for all $e' \in [0, d'-1]$ it is $b'(k+e') \models_{\mathbb{Z}} \xi_1$. To this end, we just have to realize that $\delta(d'-1) < d$. In fact, we have shown above that $d' \leq \lceil d/\delta \rceil < d/\delta + 1$, since $\lceil r \rceil < r+1$ for any real number r. So obviously $\delta(d'-1) < \delta(d/\delta) = d$. Since for all $e \in [0, d\rangle$ we have $b(t+e) \models_{\mathbb{R}} \xi_1$, and since $[0, \delta(d'-1)] \subset [0, d\rangle$, then *a fortiori* for all $e \in [0, \delta(d'-1)]$ it is $b(t+e) \models_{\mathbb{R}} \xi_1$. Finally, by inductive hypothesis, it follows that for all integers $e' \in [0, d'-1] = [0, d')$ it is $b'(k+e') \models_{\mathbb{Z}} \xi_1$, which lets us conclude that $b'(k) \models_{\mathbb{Z}} \phi'$.

- $\phi = \mathrm{Since}_{\langle l,u\rangle\langle}(\xi_1, \xi_2)$.
  Notice that ϕ' is $\mathrm{Since}_{[l',u']\langle}(\xi_1, \xi_2)$, with $l' = \lfloor l/\delta \rfloor$ and $u' = \lceil u/\delta \rceil$.

  Let d be a time instant in $\langle l, u\rangle$ such that $b(t-d) \models_{\mathbb{R}} \xi_2$ and, for all time instants $e \in \langle -d, 0]$ it is $b(t+e) \models_{\mathbb{R}} \xi_1$. Condition $\chi_\circ$ at time $t-d$ implies that there exists a $p \in [0, \delta]$ such that for all $f \in [-p, -p+\delta]$ it is $b(t-d+f) \models_{\mathbb{R}} \xi_2$. In other words, ξ_2 holds on the closed real interval $I = [t-d-p, t-d-p+\delta]$. Since I has size δ, its intersection with the sampling points (which are δ time units apart) must be non-empty. In particular, it is either $p \geq \omega(t-d)$ or $-p+\delta \geq o(t-d)$: otherwise it would be $\delta = p + (-p+\delta) < \omega(t-d) + o(t-d) = O(t-d) - \Omega(t-d) = \delta(\lceil (t-d-z)/\delta \rceil - \lfloor (t-d-z)/\delta \rfloor) \leq \delta$, a

contradiction (where we exploited the property: $\lceil r \rceil - \lfloor r \rfloor \leq 1$ for any real r).

So, let t' be the sampling instant:

$$t' = \begin{cases} \Omega(t-d) & \text{if } p \geq \omega(t-d) \\ O(t-d) & \text{otherwise} \end{cases}$$

It is not difficult to check that $(t-t')/\delta \in [l', u']$. In fact:

- if $p \geq \omega(t-d)$, then $t - t' = t - \Omega(t-d) = \delta(k - \lfloor (k\delta - d)/\delta \rfloor) = \delta(-\lfloor -d/\delta \rfloor) = \delta\lceil d/\delta \rceil$, since $\lfloor -r \rfloor = -\lceil r \rceil$ for any real r. Recall that $d \in \langle l, u \rangle$, and then *a fortiori* $d \in [l, u] \supseteq \langle l, u \rangle$. So $d/\delta \in [l/\delta, u/\delta]$, and $(t-t')/\delta = \lceil d/\delta \rceil \in [\lceil l/\delta \rceil, \lceil u/\delta \rceil] \subseteq [l', u']$.
- if $p < \omega(t+s)$, then $t - t' = t - O(t-d) = \delta(k - \lceil (k\delta - d)/\delta \rceil) = \delta(-\lceil -d/\delta \rceil) = \delta\lfloor d/\delta \rfloor$, since $\lceil -r \rceil = -\lfloor r \rfloor$ for any real r. Recall that $d \in \langle l, u \rangle$, and then *a fortiori* $d \in [l, u] \supseteq \langle l, u \rangle$. So $d/\delta \in [l/\delta, u/\delta]$, and $(t-t')/\delta = \lfloor d/\delta \rfloor \in [\lfloor l/\delta \rfloor, \lfloor u/\delta \rfloor] \subseteq [l', u']$.

So far, we have shown that $b(t') \models_{\mathbb{R}} \xi_2$. By inductive hypothesis, it follows that for $d' = (t-t')/\delta$ it is $b'(k-d') \models_{\mathbb{Z}} \xi_2$. Since $d' \in [l', u']$, to conclude this branch of the proof we have to show that for all $e' \in [-(d'-1), 0]$ it is $b'(k+e') \models_{\mathbb{Z}} \xi_1$. To this end, we just have to realize that $\delta(d'-1) < d$. In fact, we have shown above that $d' \leq \lceil d/\delta \rceil < d/\delta + 1$, since $\lceil r \rceil < r + 1$ for any real number r. So obviously $\delta(d'-1) < \delta(d/\delta) = d$. Since for all $e \in \langle -d, 0]$ we have $b(t+e) \models_{\mathbb{R}} \xi_1$, and since $[-\delta(d'-1), 0] \subset \langle -d, 0]$, then *a fortiori* for all $e \in [-\delta(d'-1), 0]$ it is $b(t+e) \models_{\mathbb{R}} \xi_1$. Finally, by inductive hypothesis, it follows that for all integers $e' \in [-(d'-1), 0] = (-d', 0]$ it is $b'(k+e') \models_{\mathbb{Z}} \xi_1$, which lets us conclude that $b'(k) \models_{\mathbb{Z}} \phi'$.

- $\phi = \text{Releases}_{\langle l,u \rangle}(\xi_1, \xi_2)$.
  Notice that ϕ' is $\text{Releases}_{\langle l',u' \rangle]}(\xi_1, \xi_2)$, where l', u' depend on the kind of interval $I = \langle l, u \rangle$ is.

  Let d' be a generic integer in $\langle l', u' \rangle$. We have to show that either $b'(k+d') \models_{\mathbb{Z}} \xi_2$ or there exists a $e' \in [0, d']$ such that $b'(k+e') \models_{\mathbb{Z}} \xi_1$.

  Now, we claim that $\langle l', u' \rangle \subseteq \langle l/\delta, u/\delta \rangle$. To show this we discuss the four possible cases for the interval $I' = \langle l', u' \rangle$.

- $I = [l, u]$, so $I' = [l', u']$, where $l' = \lceil l/\delta \rceil$ and $u' = \lfloor u/\delta \rfloor$. Thus, $[l', u'] \subseteq [l/\delta, u/\delta]$, since $\lfloor r \rfloor \leq r$ and $\lceil r \rceil \geq r$ for any real number r.
- $I = [l, u)$, so $I' = [l', u')$, where $l' = \lceil l/\delta \rceil$ and $u' = \lceil u/\delta \rceil$. Thus, $[l', u') \subseteq [l/\delta, u/\delta)$, since $[l', u') = [\lceil l/\delta \rceil, \lceil u/\delta \rceil - 1] \subseteq [l/\delta, u/\delta)$, noting that $\lceil r \rceil \geq r$, and that $\lceil r \rceil - 1 < r$, for any real r.
- $I = (l, u]$, so $I' = (l', u']$, where $l' = \lfloor l/\delta \rfloor$ and $u' = \lfloor u/\delta \rfloor$. Thus, $(l', u'] \subseteq (l/\delta, u/\delta]$, since $(l', u'] = [\lfloor l/\delta \rfloor + 1, \lfloor u/\delta \rfloor] \subseteq (l/\delta, u/\delta]$, noting that $\lfloor r \rfloor \leq r$, and that $\lfloor r \rfloor + 1 > r$, for any real r.
- $I = (l, u)$, so $I' = (l', u')$, where $l' = \lfloor l/\delta \rfloor$ and $u' = \lceil u/\delta \rceil$. Thus, $(l', u') \subseteq (l/\delta, u/\delta)$, since $(l', u') = [\lfloor l/\delta \rfloor + 1, \lceil u/\delta \rceil - 1] \subset (l/\delta, u/\delta)$, noting that $\lfloor r \rfloor + 1 > r$, and that $\lceil r \rceil - 1 < r$, for any real r.

All in all, by hypothesis it is either $b(t + \delta d') \models_{\mathbb{R}} \xi_2$ or there exists a $e \in [0, \delta d'\rangle$ such that $b(t + e) \models_{\mathbb{R}} \xi_1$.

In the former case, by inductive hypothesis we have that $b'(k + d') \models_{\mathbb{Z}} \xi_2$, which fulfills the goal. In the latter case, we can write that $b(t) \models_{\mathbb{R}} \text{Until}_{[0,\delta d'\rangle\rangle}(\text{true}, \xi_1)$. Thus, by inductive hypothesis, we infer that $b'(k) \models_{\mathbb{Z}} \text{Until}_{[0,d'\rangle)}(\text{true}, \xi_1)$. Therefore, from the definition of the Until operator, we have that there exists a $e' \in [0, d'\rangle \subseteq [0, d']$ such that $b'(k + e') \models_{\mathbb{Z}} \xi_1$, as required.

- $\phi = \text{Released}_{\langle l,u\rangle\langle}(\xi_1, \xi_2)$.
  Notice that ϕ' is $\text{Released}_{\langle l',u'\rangle[}(\xi_1, \xi_2)$, where l', u' depend on the kind of interval $I = \langle l, u\rangle$ is.

  Let d' be a generic integer in $\langle l', u'\rangle$. We have to show that either $b'(k - d') \models_{\mathbb{Z}} \xi_2$ or there exists a $e' \in [-d', 0]$ such that $b'(k + e') \models_{\mathbb{Z}} \xi_1$.

  Now, we claim that $\langle l', u'\rangle \subseteq \langle l/\delta, u/\delta\rangle$. In fact, we showed this fact in the above proof for the Releases operator (notice that the bounds l', u' are the same for the Releases and Released operators).

  All in all, by hypothesis it is either $b(t - \delta d') \models_{\mathbb{R}} \xi_2$ or there exists a $e \in \langle -\delta d', 0]$ such that $b(t + e) \models_{\mathbb{R}} \xi_1$.

In the former case, by inductive hypothesis we have that $b'(k-d') \models_{\mathbb{Z}} \xi_2$, which fulfills the goal. In the latter case, we can write that $b(t) \models_{\mathbb{R}} \text{Since}_{[0,\delta d'\rangle\langle}(\text{true}, \xi_1)$. Thus, by inductive hypothesis, we infer that $b'(k) \models_{\mathbb{Z}} \text{Since}_{[0,d'\rangle(}(\text{true}, \xi_1)$. Therefore, from the definition of the Since operator, we have that there exists a $e'' \in [0, d'\rangle \subseteq [0, d']$ such that $b'(k - e'') \models_{\mathbb{Z}} \xi_1$, or equivalently, by substituting e' for $-e''$, there exists a $e' \in [-d', 0]$ such that $b(k + e') \models_{\mathbb{Z}} \xi_1$, as required.

- $\phi = \phi_1 \wedge \phi_2$.
  Since $b(t) \models_{\mathbb{R}} \phi_i$ implies $b'(k) \models_{\mathbb{Z}} \phi_i'$ for $i = 1, 2$, then also $b(t) \models_{\mathbb{R}} \phi_1 \wedge \phi_2$ implies $b'(k) \models_{\mathbb{Z}} \phi_1' \wedge \phi_2'$.

- $\phi = \phi_1 \vee \phi_2$.
  Since $b(t) \models_{\mathbb{R}} \phi_i$ implies $b'(k) \models_{\mathbb{Z}} \phi_i'$ for $i = 1$ or $i = 2$, then also $b(t) \models_{\mathbb{R}} \phi_1 \vee \phi_2$ implies $b'(k) \models_{\mathbb{Z}} \phi_1' \vee \phi_2'$.

Since the above holds for a generic sampling instant t, we have shown that if $b \models_{\mathbb{R}} \phi$ then $\sigma_{\delta,z}[b] \models_{\mathbb{Z}} \eta_\delta^{\mathbb{R}}\{\phi\}$ for any behavior $b \in [\![\chi_\circ]\!]_{\mathbb{R}}$. Therefore, we have shown that any $\frac{\mathbb{R}}{\mathbb{Z}}$TRIO formula ϕ, interpreted in the continuous-time domain, is closed under sampling

(Closure under inverse sampling.) Let b be a discrete-time behavior, and let $\phi' = \eta_\delta^{\mathbb{Z}}\{\phi\}$. Then, let b' be a continuous-time behavior such that $b' \in [\![\chi_\circ]\!]_{\mathbb{R}}$ and $b = \sigma_{\delta,z}[b']$ for the given sampling period δ and origin z. In the remainder, for a generic sampling instant $t = z + k\delta$, we first show that if $b(k) \models_{\mathbb{Z}} \phi$ then $b'(t) \models_{\mathbb{R}} \phi'$.

Let us also introduce the following terminology. For any formula $\phi' = \eta_\delta^{\mathbb{Z}}\{\phi\}$ which holds at a sampling point $t = z + k\delta$:

- if for all $s \in (-\delta, 0)$ it is $b'(t + s) \models_{\mathbb{R}} \phi'$, we say that ϕ' "shifts to the left";

- if for all $s \in (0, \delta)$ it is $b'(t + s) \models_{\mathbb{R}} \phi'$, we say that ϕ' "shifts to the right";

- if there exists a $c \in (0, \delta)$ such that for all $t' \in [t, t+c)$ it is $b'(t') \models_{\mathbb{R}} \phi'$ and for all $t'' \in (t + c, t + c + \epsilon)$ it is $b'(t'') \models_{\mathbb{R}} \neg\phi'$, we say that ϕ' "turns false at c".

- if there exists a $c \in (-\delta, 0)$ such that for all $t' \in (t+c, t]$ it is $b'(t') \models_{\mathbb{R}} \phi'$ and for all $t'' \in (t + c - \epsilon, t + c)$ it is $b'(t'') \models_{\mathbb{R}} \neg\phi'$, we say that ϕ' "turned false at c".

In the following proof, we will also show that, for any $\phi' = \eta_\delta^{\mathbb{Z}}\{\phi\}$:

- either ϕ' shifts to the right, or there exists a $c \in (0, \delta)$ and a condition ξ such that ϕ' and ξ turn false at c;
- either ϕ' shifts to the left, or there exists a $c \in (-\delta, 0)$ such that ϕ' and ξ turned false at c.

We call the point c the "change point" of ϕ'. Notice that, whenever the above properties hold, ϕ' becomes false *together with some condition* ξ becoming false, at any change point. Therefore, in such cases, ϕ' becoming false is ultimately a consequence of some basic item changing its value.

Let us now proceed by induction on the structure of the formula ϕ.

- $\phi = \xi$.
  By definition of sampling of a behavior, the truth value of ξ at k and t is the same, i.e., $\xi|_{b(k)} = \xi|_{b'(z+k\delta)} = \xi|_{b'(t)}$.

  By Lemma D.2.1, there exist c_n, c_p such that the condition ξ changes its value at these points, or ξ is indefinitely true in the past and/or future. If the latter is the case, then ξ obviously shifts to the left/right respectively. Thus, let us assume that the former is the case.

  Let us consider c_p first, and show that either ξ shifts to the left, or there exists a $c \in (-\delta, 0)$ such that ξ turned false at c. Indeed, if $c_p - t \in (-\delta, 0)$, then ξ is true for all $t' \in (c_p, t] = (t + c, t]$, and for $\epsilon = \delta$, we have that for all $t'' \in (c_p - \epsilon, c_p)$ it is $b'(t'') \models_{\mathbb{R}} \neg\xi$, therefore ξ turned false at $t - c_p$. Otherwise, ξ shifts to the left.
  Similarly, by considering c_n, we show that either ξ shifts to the right, or there exists a $c \in (0, \delta)$ such that ξ turns false at c. Indeed, if $c_n - t \in (0, \delta)$, then ξ is true for all $t' \in [t, c_n) = [t, t + c)$, and for $\epsilon = \delta$, we have that for all $t'' \in (c_n, c_n + \epsilon)$ it is $b'(t'') \models_{\mathbb{R}} \neg\xi$, therefore ξ turns false at $c_n - t$. Otherwise, ξ shifts to the right.

- $\phi = \mathrm{Until}_{[l,u]]}(\xi_1, \xi_2)$.
  Notice that $\phi' = \mathrm{Until}_{[l',u']]}(\xi_1, \xi_2)$, with $l' = (l-1)\delta$ and $u' = (u+1)\delta$.

  First, let us show that $b'(t) \models_{\mathbb{R}} \mathrm{Until}_{[l\delta,u\delta]}(\xi_1, \xi_2)$. Notice that this implies $b'(t) \models_{\mathbb{R}} \phi'$, as $[l\delta, u\delta] \subset [(l-1)\delta, (u+1)\delta]$.
  Let $d \in [l, u]$ be the integer time instant such that $b(k + d) \models_{\mathbb{Z}} \xi_2$, which exists by hypothesis. Let $d' = d\delta$, and notice that $d' \in [l\delta, u\delta]$. Then, by inductive hypothesis, it is $b'(t + d') \models_{\mathbb{R}} \xi_2$.

By hypothesis we also know that for all integers $e \in [0, d]$ it is $b(k + e) \models_{\mathbb{Z}} \xi_1$. Thus, by inductive hypothesis, we have that for all δ-multiples $e' \in [0, d']$ it is $b'(t + e') \models_{\mathbb{R}} \xi_1$. Finally, because of the condition $\chi_\circ$, it must also be that, for all *real* values $e'' \in [0, d']$, it is $b'(t + e'') \models_{\mathbb{R}} \xi_1$. Therefore, we have shown that $b'(t) \models_{\mathbb{R}} \text{Until}_{[l\delta, u\delta]]}(\xi_1, \xi_2)$.

Now, let us show that Until shifts to the right, that is for all $s \in (0, \delta)$ it is $b'(t + s) \models_{\mathbb{R}} \phi'$.
Let s be a generic time instant in $(0, \delta)$. Let us consider $c = d' - s$, and notice that $c > d' - \delta \geq l\delta - \delta = l'$, and $c < d' \leq u\delta < u'$, thus $c \in [l', u']$. Since $t + s + c = t + d'$, we have shown above that $b'((t + s) + c) = b'(t + d') \models_{\mathbb{R}} \xi_2$.
Moreover, $[s, s + c] \subset [0, d']$, thus *a fortiori* for all $f \in [0, c]$ it is $b'((t + s) + f) \models_{\mathbb{R}} \xi_1$, from what we have shown above. This shows that $b'(t + s) \models_{\mathbb{R}} \phi'$ for any $s \in (0, \delta)$.

Finally, let us show that either ϕ' also shifts to the left, or there exist a $c \in (-\delta, 0)$ and a condition ξ such that ϕ' and ξ turned false at c.
To this end, take any $f \in (-\delta, 0)$, and let $\xi = \xi_1$. For $d'' = d' - f$ we have that $d'' \in [l\delta, (u+1)\delta] \subset [(l-1)\delta, (u+1)\delta]$ and $b'(t+f+d'') \models_{\mathbb{R}} \xi_2$.
Now, let us consider ξ_1. By inductive hypothesis, either there exists a $c \in (-\delta, 0)$ such that ξ_1 turned false at c, or ξ_1 shifts to the left. In the latter case, ϕ' shifts to the left as well. In the former case, it is easy to realize that for all $t' \in (t + c, t]$ we have $b'(t') \models_{\mathbb{R}} \phi' \wedge \xi_1$ and ϕ' turned false at c, since "right before" $t + c$ the Until must be false, since ξ_1 is false there.

- $\phi = \text{Since}_{[l,u][}(\xi_1, \xi_2)$.
  Notice that $\phi' = \text{Since}_{[l',u'][}(\xi_1, \xi_2)$, with $l' = (l-1)\delta$ and $u' = (u+1)\delta$.

  First, let us show that $b'(t) \models_{\mathbb{R}} \text{Since}_{[l\delta, u\delta][}(\xi_1, \xi_2)$. Notice that this implies $b'(t) \models_{\mathbb{R}} \phi'$, as $[l\delta, u\delta] \subset [(l-1)\delta, (u+1)\delta]$.
  Let $d \in [l, u]$ be the integer time instant such that $b(k - d) \models_{\mathbb{Z}} \xi_2$, which exists by hypothesis. Let $d' = d\delta$, and notice that $d' \in [l\delta, u\delta]$. Then, by inductive hypothesis, it is $b'(t - d') \models_{\mathbb{R}} \xi_2$.
  By hypothesis we also know that for all integers $e \in [-d, 0]$ it is $b(k + e) \models_{\mathbb{Z}} \xi_1$. Thus, by inductive hypothesis, we have that for all δ-multiples $e' \in [-d', 0]$ it is $b'(t + e') \models_{\mathbb{R}} \xi_1$. Finally, because of the condition $\chi_\circ$, it must also be that, for all *real* values $e'' \in [-d', 0]$,

it is $b'(t+e'') \models_{\mathbb{R}} \xi_1$. Therefore, we have shown that $b'(t) \models_{\mathbb{R}} \text{Since}_{[l\delta,u\delta][}(\xi_1,\xi_2)$.

Now, let us show that Since shifts to the left, that is for all $s \in (-\delta, 0)$ it is $b'(t+s) \models_{\mathbb{R}} \phi'$.
Let s be a generic time instant in $(-\delta, 0)$. Let us consider $c = d' + s$, and notice that $c > d' - \delta \geq l\delta - \delta = l'$, and $c < d' \leq u\delta < u'$, thus $c \in [l', u']$. Since $t - (c - s) = t - d'$, we have shown above that $b'((t+s)-c) = b'(t-d') \models_{\mathbb{R}} \xi_2$.
Moreover, $[s-c, s] \subset [-d', 0]$, thus *a fortiori* for all $f \in [-c, 0]$ it is $b'((t+s)+f) \models_{\mathbb{R}} \xi_1$, from what we have shown above. This shows that $b'(t+s) \models_{\mathbb{R}} \phi'$ for any $s \in (-\delta, 0)$.

Finally, let us show that either ϕ' also shifts to the right, or there exist a $c \in (0, \delta)$ and a condition ξ such that ϕ' and ξ turn false at c. To this end, take any $f \in (0, \delta)$, and let $\xi = \xi_1$. For $d'' = d' + f$ we have that $d'' \in [l\delta, (u+1)\delta] \subset [(l-1)\delta, (u+1)\delta]$ and $b'((t+f)-d'') = b'(t-d') \models_{\mathbb{R}} \xi_2$.
Now, let us consider ξ_1. By inductive hypothesis, either there exists a $c \in (-\delta, 0)$ such that ξ_1 turns false at c, or ξ_1 shifts to the right. In the latter case, ϕ' shifts to the right as well. In the former case, it is easy to realize that for all $t' \in [t, t+c)$ we have $b'(t') \models_{\mathbb{R}} \phi' \wedge \xi_1$ and ϕ' turns false at c, since "right after" $t + c$ the Since must be false, since ξ_1 is false there.

- $\phi = \text{Releases}_{[l,u])}(\xi_1, \xi_2)$.
  Notice that $\phi' = \text{Releases}_{[l',u'])}(\xi_1, \xi_2)$, with $l' = (l+1)\delta$ and $u' = (u-1)\delta$.

  First, let us show that $b'(t) \models_{\mathbb{R}} \text{Releases}_{[l\delta,u\delta])}(\xi_1, \xi_2)$. Notice that this implies $b'(t) \models_{\mathbb{R}} \phi'$, as $[l\delta, u\delta] \supset [l', u']$.
  Let d' be a generic instant in $[l\delta, u\delta]$: we have to show that either $b'(t+d') \models_{\mathbb{R}} \xi_2$ or there exists a $e' \in [0, d')$ such that $b'(t+e') \models_{\mathbb{R}} \xi_1$. We distinguish two cases, whether $t + d'$ is a sampling instant or not.

  - If $t + d'$ is a sampling instant, then $d = d'/\delta$ is an integer, and $d \in [l, u]$ by hypothesis.
    Therefore, by hypothesis it is either $b(k+d) \models_{\mathbb{Z}} \xi_2$ or there exists an integer $e \in [0, d-1]$ such that $b(k+e) \models_{\mathbb{Z}} \xi_1$.
    In the former case, by inductive hypothesis we have that $b'(t + d') \models_{\mathbb{R}} \xi_2$, which satisfies the definition of the Releases operator

for d'.
Otherwise, by inductive hypothesis we have that $b'(t+e') \models_{\mathbb{R}} \xi_1$, for $e' = e\delta$. Since $e' \in [0, d'-\delta] \subset [0, d')$, we have that the definition of the Releases operator is satisfied for d' in this case as well.

– If $t + d'$ is not a sampling instant, let us consider the distances $p' = d' - \omega(t + d')$ and $n' = d' + o(t + d')$; these are δ-multiples by definition of $\omega(\cdot)$ and $o(\cdot)$. Notice that $p' > d' - \delta \geq l\delta - \delta = (l-1)\delta$, and $n' < d' + \delta \leq u\delta + \delta = (u+1)\delta$. Therefore, the two integers $p = p'/\delta$ and $n = n'/\delta$ are $\geq l$ and $\leq u$ respectively, that is $p, n \in [l, u]$.
Thus, by hypothesis we have that:

  * $b(k + p) \models_{\mathbb{Z}} \xi_2$ or there exists a $e_p \in [0, p-1]$ such that $b(k + e_p) \models_{\mathbb{Z}} \xi_1$; and that:
  * $b(k + n) \models_{\mathbb{Z}} \xi_2$ or there exists a $e_n \in [0, n-1]$ such that $b(k + e_n) \models_{\mathbb{Z}} \xi_1$.

Therefore, we distinguish the following three cases (covering all the possibilities):

  * $b(k + p) \models_{\mathbb{Z}} \xi_2$ and $b(k + n) \models_{\mathbb{Z}} \xi_2$;
  * there exists a $e_p \in [0, p-1]$ such that $b(k + e_p) \models_{\mathbb{Z}} \xi_1$;
  * there exists a $e_n \in [0, n-1]$ such that $b(k + e_n) \models_{\mathbb{Z}} \xi_1$.

Let us first consider the case: $b(k + p) \models_{\mathbb{Z}} \xi_2$ and $b(k + n) \models_{\mathbb{Z}} \xi_2$. Notice that $n = p + 1$, so ξ_2 is true on two adjacent time instants. By inductive hypothesis, we have that $b'(t+p') \models_{\mathbb{R}} \xi_2$ and $b'(t + n') \models_{\mathbb{R}} \xi_2$. Moreover, thanks to the constraint $\chi_\circ$ which holds for b by hypothesis, ξ_2 must also be true for all time instants in the interval $[t + p', t + n'] = [t + p', t + p' + \delta]$. In particular, since $d' \in [p', p' + \delta]$, then $b'(t + d') \models_{\mathbb{R}} \xi_2$. That is, the definition of the Releases is satisfied for d'.
Now, for the other two cases, notice that $(p-1)\delta \leq (n-1)\delta < (d' + \delta) - \delta = d'$; hence $e_p, e_n \in [0, n-1]$ and $\delta e_p, \delta e_n \in [0, (n-1)\delta] \subset [0, d')$. Therefore, if there exists a $e_p \in [0, p-1]$ such that $b(k + e_p) \models_{\mathbb{Z}} \xi_1$, or if there exists a $e_n \in [0, n-1]$ such that $b(k + e_n) \models_{\mathbb{Z}} \xi_1$, then there exists an $e' = \delta e_p$ or $e' = \delta e_n$, respectively, such that $e' \in [0, d')$ and $b'(t + e') \models_{\mathbb{R}} \xi_1$. That is, the definition of the Releases is satisfied for d' in both these cases.

All in all, since d' is generic, we have shown that
$b'(t) \models_{\mathbb{R}} \text{Releases}_{[l\delta,u\delta])}(\xi_1, \xi_2)$, and thus *a fortiori* $b'(t) \models_{\mathbb{R}} \phi'$.

Now, let us show that Releases shifts to the left, that is for all $s \in (-\delta, 0)$ it is $b'(t+s) \models_{\mathbb{R}} \phi'$.
Let s be any time instant in $(-\delta, 0)$, and consider a generic $d \in [l', u']$. Since $s + d \in [l\delta, u\delta]$, we have shown above that it is either $b'(t + (s + d)) \models_{\mathbb{R}} \xi_2$ or there exists a $e' \in [0, s + d)$ such that $b'(t + e') \models_{\mathbb{R}} \xi_1$.
In the former case, we have shown that $b'((t + s) + d) \models_{\mathbb{R}} \xi_2$. Otherwise, let $e'' = e' - s$; then $e'' \in [-s, d) \subset [0, d)$, as $s < 0$. Therefore, we have that there exists a $e'' \in [0, d)$ such that $b'(t + e') = b'((t + s) + e'') \models_{\mathbb{R}} \xi_1$. All in all, we have shown that $b'(t + s) \models_{\mathbb{R}} \phi'$ for a generic $s \in (-\delta, 0)$.

Finally, let us show that either ϕ' also shifts to the right, or there exist a $c \in (0, \delta)$ and a condition ξ such that ϕ' and ξ turn false at c.
To this end, take generic $f \in (0, \delta)$ and $d'' \in [(l + 1)\delta, (u - 1)\delta]$, and let $\xi = \xi_1$. Notice that $d'' + f \in [l\delta, u\delta]$, therefore by hypothesis it is either $b'(t + d'' + f) \models_{\mathbb{R}} \xi_2$ or there exists a $c \in [0, d'' + f)$ such that $b'(t + c) \models_{\mathbb{R}} \xi_1$.
Now let us consider two cases:

- For all f and d'' as above, it is either $b'((t + f) + d'') \models_{\mathbb{R}} \xi_2$ or there exists a $c \in [f, d'' + f)$ such that $b'(t + c) = b'((t + f) + (c - f)) \models_{\mathbb{R}} \xi_1$.
  Since $c - f \in [0, d'')$, this shows that $b'(t + f) \models_{\mathbb{R}} \phi'$, and thus ϕ' shifts to the right since f is generic.
- Otherwise, assume that the previous case is false, that is there is some f and d'' as above such that $b'((t+f)+d'') \models_{\mathbb{R}} \neg\xi_2$ and for all $c' \in [f, d'' + f)$ it is $b'(t + c') = b'((t + f) + (c' - f)) \models_{\mathbb{R}} \neg\xi_1$. Therefore, it must be that there exists a $c \in [0, f)$ such that $b'(t + c) \models_{\mathbb{R}} \xi_1$.
  Now, if we consider Lemma D.2.1, and because of the non-Zeno behavior of the basic items values, a point $t + v$ at which ξ_1 becomes false must exist for some v, that is for some v it must be $b'(t+v) \models_{\mathbb{R}} \text{Becomes}(\neg\xi_1)$. Then, ξ_1 is true in the whole interval $[t, t + v)$, since $v = c < f < \delta$ in this branch of the proof, and false afterward, and in particular in the interval $(t+v, t+v+\delta)$. Consequently, ϕ' is also true in the whole interval $[t, t + v)$, and

it becomes false "right after" $t+v$, as a consequence of ξ_1 turning false.

A little reasoning should convince us that the two cases do indeed cover all the possibilities. Then, since f is generic, we have shown that either ϕ' shifts to the right, or there exists a $c \in (0, \delta)$ such that ϕ' and ξ_1 turn false at c.

- $\phi = \text{Released}_{[l,u](}(\xi_1, \xi_2)$.
  Notice that $\phi' = \text{Released}_{[l',u'](}(\xi_1, \xi_2)$, with $l' = (l+1)\delta$ and $u' = (u-1)\delta$.

  First, let us show that $b'(t) \models_{\mathbb{R}} \text{Released}_{[l\delta,u\delta](}(\xi_1, \xi_2)$. Notice that this implies $b'(t) \models_{\mathbb{R}} \phi'$, as $[l\delta, u\delta] \supset [l', u']$.
  Let d' be a generic instant in $[l\delta, u\delta]$: we have to show that either $b'(t-d') \models_{\mathbb{R}} \xi_2$ or there exists a $e' \in (-d', 0]$ such that $b'(t+e') \models_{\mathbb{R}} \xi_1$.
  We distinguish two cases, whether $t - d'$ is a sampling instant or not.

  - If $t - d'$ is a sampling instant, then $d = d'/\delta$ is an integer, and $d \in [l, u]$ by hypothesis.
    Therefore, by hypothesis it is either $b(k-d) \models_{\mathbb{Z}} \xi_2$ or there exists an integer $e \in [-(d-1), 0]$ such that $b(k+e) \models_{\mathbb{Z}} \xi_1$.
    In the former case, by inductive hypothesis we have that $b'(t - d') \models_{\mathbb{R}} \xi_2$, which satisfies the definition of the Released operator for d'.
    Otherwise, by inductive hypothesis we have that $b'(t+e') \models_{\mathbb{R}} \xi_1$, for $e' = e\delta$. Since $e' \in [-(d' - \delta), 0] \subset (-d', 0]$, we have that the definition of the Released operator is satisfied for d' in this case as well.

  - If $t - d'$ is not a sampling instant, let us consider the distances $p' = d' - o(t - d')$ and $n' = d' + \omega(t - d')$; these are δ-multiples by definition of $\omega()$ and $o()$. Notice that $p' > d' - \delta, \geq l\delta - \delta = (l-1)\delta$, and $n' < d' + \delta \leq u\delta + \delta = (u+1)\delta$. Therefore, the two integers $p = p'/\delta$ and $n = n'/\delta$ are $\geq l$ and $\leq u$ respectively, that is $p, n \in [l, u]$.
    Thus, by hypothesis we have that:

    * $b(k-p) \models_{\mathbb{Z}} \xi_2$ or there exists a $e_p \in [-(p-1), 0]$ such that $b(k+e_p) \models_{\mathbb{Z}} \xi_1$; and that:

* $b(k-n) \models_{\mathbb{Z}} \xi_2$ or there exists a $e_n \in [-(n-1), 0]$ such that $b(k+e_n) \models_{\mathbb{Z}} \xi_1$.

Therefore, we distinguish the following three cases (covering all the possibilities):

* $b(k-p) \models_{\mathbb{Z}} \xi_2$ and $b(k-n) \models_{\mathbb{Z}} \xi_2$;
* there exists a $e_p \in [-(p-1), 0]$ such that $b(k+e_p) \models_{\mathbb{Z}} \xi_1$;
* there exists a $e_n \in [-(n-1), 0]$ such that $b(k+e_n) \models_{\mathbb{Z}} \xi_1$.

Let us first consider the case: $b(k-p) \models_{\mathbb{Z}} \xi_2$ and $b(k-n) \models_{\mathbb{Z}} \xi_2$. Notice that $-n = -p - 1$, so ξ_2 is true on two adjacent time instants. By inductive hypothesis, we have that $b'(t-p') \models_{\mathbb{R}} \xi_2$ and $b'(t-n') \models_{\mathbb{R}} \xi_2$. Moreover, thanks to the constraint $\chi_\circ$ which holds for b by hypothesis, ξ_2 must also be true for all time instants in the interval $[t-n', t-p'] = [t-n', t-(n'-\delta)]$. In particular, since $d' \in [n'-\delta, n']$, then $b'(t-d') \models_{\mathbb{R}} \xi_2$. That is, the definition of the Released is satisfied for d'.
Now, for the other two cases, notice that $-(p-1)\delta \geq -(n-1)\delta > -d' + \delta - \delta = -d'$; hence $e_p, e_n \in [-(n-1), 0]$ and $\delta e_p, \delta e_n \in [-(n-1)\delta, 0] \subset (-d', 0]$. Therefore, if there exists a $e_p \in [-(p-1), 0]$ such that $b(k+e_p) \models_{\mathbb{Z}} \xi_1$, or if there exists a $e_n \in [-(n-1), 0]$ such that $b(k+e_n) \models_{\mathbb{Z}} \xi_1$, then there exists an $e' = \delta e_p$ or $e' = \delta e_n$, respectively, such that $e' \in (-d', 0]$ and $b'(t+e') \models_{\mathbb{R}} \xi_1$. That is, the definition of the Released is satisfied for d' in both these cases.

All in all, since d' is generic, we have shown that
$b'(t) \models_{\mathbb{R}} \text{Released}_{[l\delta, u\delta](}(\xi_1, \xi_2)$, and thus *a fortiori* $b'(t) \models_{\mathbb{R}} \phi'$.

Now, let us show that Released shifts to the right, that is for all $s \in (0, \delta)$ it is $b'(t+s) \models_{\mathbb{R}} \phi'$.
Let s be any time instant in $(0, \delta)$, and consider a generic $d \in [l', u']$. Since $d - s \in [l\delta, u\delta]$, we have shown above that it is either $b'(t-(d-s)) \models_{\mathbb{R}} \xi_2$ or there exists a $e' \in (-(d-s), 0]$ such that $b'(t+e') \models_{\mathbb{R}} \xi_1$.
In the former case, we have shown that $b'((t+s)-d) \models_{\mathbb{R}} \xi_2$. Otherwise, let $e'' = e' - s$; then $e'' \in (-d, -s] \subset (-d, 0]$, as $s > 0$. Therefore, we have that there exists a $e'' \in (-d, 0]$ such that $b'(t+e') = b'((t+s)+e'') \models_{\mathbb{R}} \xi_1$. All in all, we have shown that $b'(t+s) \models_{\mathbb{R}} \phi'$ for a generic $s \in (0, \delta)$.

Finally, let us show that either ϕ' also shifts to the left, or there exist a $c \in (-\delta, 0)$ and a condition ξ such that ϕ' and ξ turned false at c. To this end, take generic $f \in (-\delta, 0)$ and $d'' \in [(l+1)\delta, (u-1)\delta]$, and let $\xi = \xi_1$. Notice that $d'' - f \in [l\delta, u\delta]$, therefore by hypothesis it is either $b'(t-(d''-f)) \models_{\mathbb{R}} \xi_2$ or there exists a $c \in (d''+f, 0]$ such that $b'(t+c) \models_{\mathbb{R}} \xi_1$.
Now let us consider two cases:

- For all f and d'' as above, it is either $b'((t+f)-d'') \models_{\mathbb{R}} \xi_2$ or there exists a $c \in (-(d''-f), f]$ such that $b'(t+c) = b'((t+f)+(c-f)) \models_{\mathbb{R}} \xi_1$.
  Since $c-f \in (-d'', 0]$, this shows that $b'(t+f) \models_{\mathbb{R}} \phi'$, and thus ϕ' shifts to the left since f is generic.
- Otherwise, assume that the previous case is false, that is there is some f and d'' as above such that $b'((t+f)-d'') \models_{\mathbb{R}} \neg\xi_2$ and for all $c' \in (-(d''-f), f]$ it is $b'(t+c') \models_{\mathbb{R}} \neg\xi_1$. Therefore, it must be that there exists a $c \in (f, 0]$ such that $b'(t+c) \models_{\mathbb{R}} \xi_1$. Now, if we consider Lemma D.2.1, and because of the non-Zeno behavior of the basic items values, a point $t+v$ at which ξ_1 becomes false must exist for some $v < 0$, that is for some $v < 0$ it must be $b'(t+v) \models_{\mathbb{R}} \text{Becomes}(\xi_1)$. Then, ξ_1 is true in the whole interval $(t+v, t]$, since $-v = -c < -f < \delta$ in this branch of the proof, and false beforehand, and in particular in the interval $(t+v, t+v-\delta)$. Consequently, ϕ' is also true in the whole interval $(t+v, t]$, and it became false "right before" $t+v$, as a consequence of ξ_1 turned false.

A little reasoning should convince us that the two cases do indeed cover all the possibilities. Then, since f is generic, we have shown that either ϕ' shifts to the left, or there exists a $c \in (-\delta, 0)$ such that ϕ' and ξ_1 turned false at c.

- $\phi = \phi_1 \wedge \phi_2$.
  Since $b(k) \models_{\mathbb{Z}} \phi_i$ implies $b'(t) \models_{\mathbb{R}} \phi_i'$ for $i = 1, 2$ by inductive hypothesis, then also $b(k) \models_{\mathbb{Z}} \phi_1 \wedge \phi_2$ implies $b'(t) \models_{\mathbb{R}} \phi_1' \wedge \phi_2'$.

  Now, by inductive hypothesis, for both $i = 1, 2$: either ϕ_i' shifts to the right, or there exists a $c_i \in (0, \delta)$ and some condition ξ_i such that ϕ_i' and ξ_i turn to false at c_i. Thus, if both ϕ_1' and ϕ_2' shift to the right, then $\phi' = \phi_1' \wedge \phi_2'$ also shifts to the right; otherwise there exists

a $c \in (0, \delta) = \min_{i=1,2} c_i$ such that $\phi_1' \wedge \phi_2'$ turns to false at c.
Similarly, by inductive hypothesis for both $i = 1, 2$: either ϕ_i' shifts to the left, or there exists a $c_i \in (-\delta, 0)$ and some condition ξ_i such that ϕ_i' and ξ_i turned to false at c_i. Thus, if both ϕ_1' and ϕ_2' shift to the left, then $\phi' = \phi_1' \wedge \phi_2'$ also shifts to the left; otherwise there exists a $c \in (-\delta, 0) = \max_{i=1,2} c_i$ such that $\phi_1' \wedge \phi_2'$ turned false at c.

- $\phi = \phi_1 \vee \phi_2$.
  Since $b(k) \models_{\mathbb{Z}} \phi_i$ implies $b'(t) \models_{\mathbb{R}} \phi_i'$ for $i = 1$ or $i = 2$ by inductive hypothesis, then also $b(k) \models_{\mathbb{Z}} \phi_1 \vee \phi_2$ implies $b'(t) \models_{\mathbb{R}} \phi_1' \vee \phi_2'$.

  Now, we realize that if $b'(t) \models_{\mathbb{R}} \phi_i'$ for some $i = 1$ or $i = 2$, then by inductive hypothesis either ϕ_i' shifts to the right or there exists a $c \in (0, \delta)$ and a condition ξ_i such that ϕ_i' and ξ_i turn false at c. Thus, *a fortiori* either ϕ' shifts to the right, or there exist a $c \in (0, \delta)$ and a condition $\xi = \xi_i$ such that ϕ' and ξ turn false at c.
  Similarly, by inductive hypothesis either ϕ_i' shifts to the left or there exists a $c \in (-\delta, 0)$ and a condition ξ_i such that ϕ_i' and ξ_i turned false at c. Thus, *a fortiori* either ϕ' shifts to the left, or there exist a $c \in (-\delta, 0)$ and a condition ξ such that ϕ' and $\xi = \xi_i$ turned false at c.

(Filling in the gaps between sampling points.) So far, we have proved that, for any formula ϕ, if $b \models_{\mathbb{Z}} \phi$ then for all $t = z + k\delta$ for *integer* k's, it is $b'(t) \models_{\mathbb{R}} \phi'$. Moreover, we have proved some results about shifting of temporal operators that we are going to use in the remainder.

Now, in order to finish the proof by showing that $b'(t) \models_{\mathbb{R}} \phi'$ for all $t \in \mathbb{R}$, that is $b' \models_{\mathbb{R}} \phi'$, we have to demonstrate that any formula ϕ' which is true at all sampling points cannot become false between any two of them. More precisely, we prove that whenever $\phi' = \eta_\delta^{\mathbb{Z}}\{\phi\}$ is true at two adjacent sampling points $t_p = z + k\delta$ and $t_n = t_p + \delta = z + (k+1)\delta$, then for all $t' \in [t_p, t_n]$ it is $b'(t') \models_{\mathbb{R}} \phi'$.

If ϕ' at t_p shifts to the right, or ϕ' at t_n shifts to the left, then the conclusion follows trivially.

Thus, let us assume that ϕ' does not shift to the right at t_p and does not shift to the left at t_n. Therefore, we proved above that:

- there exist a $c_p \in (0, \delta)$ and a condition ξ_p such that ϕ' and ξ_p turn false at c_p; and

- there exist a $c_n \in (-\delta, 0)$ and a condition ξ_n such that ϕ' and ξ_n turned false at c_n.

Now notice that $|(t_n+c_n)-(t_p+c_p)| = |\delta+c_n-c_p| < \delta$. Therefore, because of condition $\chi_\circ$, the two changing points must actually coincide, otherwise there would be two changing points (corresponding to two changing points for the value of the conditions ξ_p and ξ_n, and thus to the changing points of some basic items) within δ time units. But the two formulas above contradict each other if the two changing points coincide. Thus, this means that ϕ' actually *never* becomes false in the interval $[t_p, t_n]$, which is what we had to prove.

Since t_p, t_n are actually generic, we have proved that $b \models_{\mathbb{R}} \phi'$, that is any $^{\mathbb{R}}_{\mathbb{Z}}$TRIO formula ϕ is closed under inverse sampling.

Since any formula of the $^{\mathbb{R}}_{\mathbb{Z}}$TRIO language is both closed under sampling and closed under inverse sampling, then $^{\mathbb{R}}_{\mathbb{Z}}$TRIO is sampling invariant. □

D.3. Sufficient Conditions for Non-Degenerate Intervals

Let us provide the proofs for the formulas in Table 9.1.

Adapting from continuous time to discrete time.

- *Until and Since.* Let $l' = \lfloor l/\delta \rfloor$ and $u' = \lceil u/\delta \rceil$. Thus, $u' - l' + 1 \geq u/\delta - l/\delta + 1 \geq 1$ by definition of floor and ceilings, that is $|[l', u']|_{\mathbb{Z}} > 0$.
- *Releases and Released.* Let $I' = \langle l', u' \rangle$ be the adapted interval. We distinguish the following four cases.
  - $I' = [l', u']$, with $l' = \lceil l/\delta \rceil$ and $u' = \lfloor u/\delta \rfloor$. Thus, $\lfloor u/\delta \rfloor - \lceil l/\delta \rceil + 1 \geq \lfloor u/\delta \rfloor - \lfloor l/\delta \rfloor - 1 + 1 = \lfloor u/\delta \rfloor - \lfloor l/\delta \rfloor \geq \lfloor u/\delta - l/\delta \rfloor \geq 1$, since $|[u, l]|_{\mathbb{R}} \geq \delta$ by hypothesis, and $\lceil r \rceil \leq r + 1$ and $\lfloor r_1 \rfloor - \lfloor r_2 \rfloor \geq \lfloor r_1 - r_2 \rfloor$ for any real numbers $r, r_1 \geq r_2$. That is, $|[l', u']|_{\mathbb{Z}} \geq 1$.
  - $I' = [l', u')$, with $l' = \lceil l/\delta \rceil$ and $u' = \lceil u/\delta \rceil$. Thus, $|[l', u')|_{\mathbb{Z}} = |[l', u' - 1]|_{\mathbb{Z}} = u' - l'$, and then $\lceil u/\delta \rceil - \lceil l/\delta \rceil \geq u/\delta - \lceil l/\delta \rceil > u/\delta - (l/\delta + 1) \geq 0$, since $|[l, u)|_{\mathbb{R}} \geq \delta$ by hypothesis, $\lceil r \rceil < r + 1$ for any real r, and we are considering integer numbers (hence > 0 implies ≥ 1). That is, $|[l', u')|_{\mathbb{Z}} \geq 1$.

- $I' = (l', u']$, with $l' = \lfloor l/\delta \rfloor$ and $u' = \lfloor u/\delta \rfloor$. Thus, $|(l', u']|_{\mathbb{Z}} = |[l'+1, u']|_{\mathbb{Z}} = u' - l'$, and then $\lfloor u/\delta \rfloor - \lfloor l/\delta \rfloor \geq \lfloor u/\delta \rfloor - l/\delta > (u/\delta - 1) - l/\delta \geq 0$, since $|(l, u]|_{\mathbb{R}} \geq \delta$ by hypothesis, $\lfloor r \rfloor > r - 1$ for any real r, and we are considering integer numbers (hence > 0 implies ≥ 1). That is, $|(l', u']|_{\mathbb{Z}} \geq 1$.
- $I' = (l', u')$, with $l' = \lfloor l/\delta \rfloor$ and $u' = \lceil u/\delta \rceil$. Thus, $|(l', u')|_{\mathbb{Z}} = |[l'+1, u'-1]|_{\mathbb{Z}} = u' - l' - 1$, and then $\lceil u/\delta \rceil - \lfloor l/\delta \rfloor - 1 \geq u/\delta - l/\delta - 1 \geq 1$, since $|(l, u)|_{\mathbb{R}} \geq 2\delta$ by hypothesis, and $\lfloor r \rfloor \leq r$ and $\lceil r \rceil \geq r$ for any real number r. That is, $|(l', u')|_{\mathbb{Z}} \geq 1$.

Adapting from discrete time to continuous time.

- *Until and Since.* Let $I' = [l', u']$, with $l' = (l-1)\delta$ and $u' = (u+1)\delta$. Thus, $u' - l' = \delta(u - l + 2) = \delta((u - l + 1) + 1) \geq 2\delta > 0$, since $|[l, u]|_{\mathbb{Z}} = u - l + 1 \geq 1$ by hypothesis.

- *Releases and Released.* Let $I' = [l', u']$, with $l' = (l+1)\delta$ and $u' = (u-1)\delta$. Thus, $u' - l' = \delta(u - l - 2) = \delta((u - l + 1) - 3) \geq 0$, since $|[l, u]|_{\mathbb{Z}} = u - l + 1 \geq 3$ by hypothesis.

D.4. Theorems about Strict vs. Non-Strict Operators

D.4.1. Proof of Formula 9.3

Proof of Formula (9.3). Let us start with the $\Rightarrow$ direction: assume that $b(t) \models_{\mathbb{R}} \widetilde{\text{Until}}_{(a,b\rangle)}(\phi_1, \phi_2)$. That is, there exists a $u \in (t+a, t+b\rangle$ such that $b(u) \models_{\mathbb{R}} \phi_2$ and, for all $v \in (t, u)$ it is $b(v) \models_{\mathbb{R}} \phi_1$. From $b(u) \models_{\mathbb{R}} \phi_2$ it follows immediately that $\text{Until}_{(a,b\rangle)}(\text{true}, \phi_2)$, so the first conjunct is proved.

Then, let us show that $b(t) \models_{\mathbb{R}} \text{Lasts}_{\text{ei}}\big(\text{Until}_{(0,+\infty))}(\phi_1, \phi_2), a\big)$. Let α be any instant in $(0, a]$; we have to show that $b(t+\alpha) \models_{\mathbb{R}} \text{Until}_{(0,+\infty))}(\phi_1, \phi_2)$. Notice that $(t+a, t+b\rangle \subseteq (t+\alpha, +\infty)$, as $t + a \geq t + \alpha$, therefore $u \in (t+\alpha, +\infty)$ *a fortiori*. Moreover, $[t+\alpha, u) \subset (t, u)$, as $\alpha > 0$, so for all $v' \in [t+\alpha, u)$ it is $b(v') \models_{\mathbb{R}} \phi_1$. Therefore, we have shown that $b(t+\alpha) \models_{\mathbb{R}} \text{Until}_{(0,+\infty))}(\phi_1, \phi_2)$.

Let us now consider the $\Leftarrow$ direction. First of all, let us notice that $b(t) \models_{\mathbb{R}} \text{Lasts}_{\text{ei}}\big(\text{Until}_{(0,+\infty))}(\phi_1, \phi_2), a\big)$ implies that $b(t) \models_{\mathbb{R}} \text{Lasts}_{\text{ee}}(\phi_1, a)$.

Moreover, in particular $b(t+a) \models_{\mathbb{R}} \text{Until}_{(0,+\infty))}(\phi_1, \phi_2)$. That is, there exists a $u \in (t+a, +\infty)$ such that $b(u) \models_{\mathbb{R}} \phi_2$ and, for all $v \in [t+a, u)$ it is $b(v) \models_{\mathbb{R}} \phi_1$.

Let us now consider the case $u \in (t+a, t+b\rangle$. All in all, ϕ_1 holds over the interval (t, u), and ϕ_2 holds at u; therefore, we have $b(t) \models_{\mathbb{R}} \widetilde{\text{Until}}_{(a,b\rangle)}(\phi_1, \phi_2)$.

Otherwise, let us consider the case $u \notin (t+a, t+b\rangle$; therefore $u \in \langle' t+b, +\infty)$, where $\langle'$ is the "complement" of $\rangle$.[2] In particular, this implies that for all $v \in (t, t+b)$ it is $b(v) \models_{\mathbb{R}} \phi_1$. Moreover, we are also assuming that $b(t) \models_{\mathbb{R}} \text{Until}_{(a,b\rangle)}(\text{true}, \phi_2)$ in this branch of the proof. That is, there exists a $u' \in (t+a, t+b\rangle$ such that $b(u') \models_{\mathbb{R}} \phi_2$. Since $(t, u') \subseteq (t, t+b)$, then we have shown that $b(t) \models_{\mathbb{R}} \widetilde{\text{Until}}_{(a,b\rangle]}(\phi_1, \phi_2)$, as required. □

D.4.2. Proof of Formula 9.4

Proof of Formula (9.4). Beginning with the $\Rightarrow$ direction, assume that there exists a $u \in [t+a, t+b\rangle$ such that $b(u) \models_{\mathbb{R}} \phi_2$ and for all $v \in (t, u)$ it is $b(v) \models_{\mathbb{R}} \phi_1$. Then, if $u \in (t+a, t+b\rangle$, obviously $b(t) \models_{\mathbb{R}} \widetilde{\text{Until}}_{(a,b\rangle)}(\phi_1, \phi_2)$. Otherwise, it is $u = t+a$; therefore $b(t) \models_{\mathbb{R}} \text{Futr}(\phi_2, a)$ and $b(t) \models_{\mathbb{R}} \text{Lasts}_{\text{ee}}(\phi_1, a)$.

For the $\Leftarrow$ direction, let us first consider $b(t) \models_{\mathbb{R}} \widetilde{\text{Until}}_{(a,b\rangle)}(\phi_1, \phi_2)$; then there exists a $u \in (t+a, t+b\rangle)$ such that $b(u) \models_{\mathbb{R}} \phi_2$ and for all $v \in (t, u)$ it is $b(v) \models_{\mathbb{R}} \phi_1$. Since $(t+a, t+b\rangle \subset [t+a, t+b\rangle$, then *a fortiori* $b(t)\widetilde{\text{Until}}_{[a,b\rangle)}(\phi_1, \phi_2)$. Otherwise, $b(t) \models_{\mathbb{R}} \text{Lasts}_{\text{ee}}(\phi_1, a) \wedge \text{Futr}(\phi_2, a)$ implies $b(t) \models_{\mathbb{R}} \widetilde{\text{Until}}_{[a,b\rangle)}(\phi_1, \phi_2)$ for $u = t+a$. □

D.4.3. Proofs of Formulas 9.7–9.9

Proof of Formula 9.7. Let us start with the $\Rightarrow$ direction, and let t be the current instant. We assume that there exists a $u \in [t+a, t+b\rangle$ such that $b(u) \models_{\mathbb{R}} \phi_2$ and, for all $v \in (t, u]$ it is $b(v) \models_{\mathbb{R}} \phi_1$. Clearly, $b(t) \models_{\mathbb{R}} \text{Until}_{[a,b]}(\text{true}, \phi_2 \wedge \phi_1)$ is immediately implied. We still have to show that, for all $d \in (t, t+a]$, it is $b(d) \models_{\mathbb{R}} \text{Until}_{[0,+\infty)]}(\phi_1, \phi_2)$.

So, let us consider a generic d; notice that $u \in [d, +\infty)$ as $d \leq t+a \leq u$, and recall that $b(u) \models_{\mathbb{R}} \phi_2$. Moreover, let v' be any point in $[d, u]$; since $d > t$, *a fortiori* $v' \in (t, u]$. Thus, by hypothesis $b(v') \models_{\mathbb{R}} \phi_1$, so we are done with proving $b(d) \models_{\mathbb{R}} \text{Until}_{[0,+\infty)]}(\phi_1, \phi_2)$.

[2] That is $\rangle'$ is $[$ and $]'$ is $($.

Let us now consider the $\Leftarrow$ direction. First of all, let us realize that $b(t) \models_{\mathbb{R}} \text{Lasts}_{\text{ei}}\big(\text{Until}_{[0,+\infty)]}(\phi_1, \phi_2), a\big)$ implies $b(t) \models_{\mathbb{R}} \text{Lasts}_{\text{ee}}(\phi_1, a)$. In fact, otherwise there would be a $y \in (t, t+a)$ such that $b(y) \models_{\mathbb{R}} \neg\phi_1$; but since it is also $b(y) \models_{\mathbb{R}} \text{Until}_{[0,+\infty)]}(\phi_1, \phi_2)$ we clearly have a contradiction.

Next, let us consider the consequences of $b(t+a) \models_{\mathbb{R}} \text{Until}_{[0,+\infty)]}(\phi_1, \phi_2)$: it means that there exists a $u' \in [t+a, +\infty)$ such that $b(u') \models_{\mathbb{R}} \phi_2$ and for all $v' \in [t+a, u']$ it is $b(v') \models_{\mathbb{R}} \phi_1$.

Let us first distinguish the case $u' \in [t+a, t+b\rangle$. All in all, ϕ_1 holds over the interval $(t, u']$, and ϕ_2 holds at u'; therefore, we have $b(t) \models_{\mathbb{R}} \widetilde{\text{Until}}_{[a,b)]}(\phi_1, \phi_2)$.

Otherwise, let us consider the case $u' \notin [t+a, t+b\rangle$; therefore $u' \in \langle' t+b, +\infty)$, where $\langle'$ is the "complement" of $\rangle$. In particular, this implies that for all $v' \in (t, t+b)$ it is $b(v') \models_{\mathbb{R}} \phi_1$. Moreover, we are also assuming that $b(t) \models_{\mathbb{R}} \text{Until}_{[a,b)]}(\text{true}, \phi_2 \wedge \phi_1)$ in this branch of the proof. That is, there exists a $u'' \in [t+a, t+b\rangle$ such that $b(u'') \models_{\mathbb{R}} \phi_2 \wedge \phi_1$. Since $(t, u'') \subseteq (t, t+b)$, then we have shown that $b(t) \models_{\mathbb{R}} \widetilde{\text{Until}}_{[a,b\rangle)}(\phi_1, \phi_2)$, as required. □

Proof of Formula 9.8. Let us start with the $\Rightarrow$ direction, and let t be the current instant. We assume that there exists a $u \in [t+a, t+b\rangle$ such that $b(u) \models_{\mathbb{R}} \phi_2$ and, for all $v \in (t, u)$ it is $b(v) \models_{\mathbb{R}} \phi_1$. If, furthermore, $b(u) \models_{\mathbb{R}} \phi_1$, then $b(t) \models_{\mathbb{R}} \widetilde{\text{Until}}_{[a,b\rangle]}(\phi_1, \phi_2)$ and we are done. Otherwise, let us consider the case $b(u) \models_{\mathbb{R}} \neg\phi_1$: clearly, $b(t) \models_{\mathbb{R}} \text{Until}_{[a,b\rangle]}(\text{true}, \neg\phi_1 \wedge \phi_2)$ is immediately implied (as $\neg\phi_1 \wedge \phi_2$ holds at u). We still have to show that, for all $d \in (t, t+a)$, it is $b(d) \models_{\mathbb{R}} \text{Until}_{(0,+\infty))}(\phi_1, \neg\phi_1 \wedge \phi_2)$.

So, let us consider a generic d; notice that $u \in (d, +\infty)$ as $d < t+a \leq u$, and recall that $b(u) \models_{\mathbb{R}} \phi_2 \wedge \neg\phi_1$. Moreover, let v' be any point in $[d, u)$; since $d > t$, *a fortiori* $v' \in (t, u)$. Thus, by hypothesis $b(v') \models_{\mathbb{R}} \phi_1$, so we are done with proving $b(d) \models_{\mathbb{R}} \text{Until}_{(0,+\infty))}(\phi_1, \neg\phi_1 \wedge \phi_2)$.

Let us now consider the $\Leftarrow$ direction. The implication $\widetilde{\text{Until}}_{[a,b\rangle]}(\phi_1, \phi_2) \Rightarrow \widetilde{\text{Until}}_{[a,b\rangle)}(\phi_1, \phi_2)$ is straightforward, so let us assume that $b(t) \models_{\mathbb{R}} \text{Lasts}_{\text{ee}}\big(\text{Until}_{(0,+\infty))}(\phi_1, \neg\phi_1 \wedge \phi_2), a\big)$ and $b(t) \models_{\mathbb{R}} \text{Until}_{[a,b\rangle]}(\text{true}, \neg\phi_1 \wedge \phi_2)$.

First of all, let us realize that $b(t) \models_{\mathbb{R}} \text{Lasts}_{\text{ee}}\big(\text{Until}_{(0,+\infty))}(\phi_1, \neg\phi_1 \wedge \phi_2), a\big)$ implies $b(t) \models_{\mathbb{R}} \text{Lasts}_{\text{ee}}(\phi_1, a)$. In fact, otherwise there would be a $y \in (t, t+a)$ such that $b(y) \models_{\mathbb{R}} \neg\phi_1$; but since it is also $b(y) \models_{\mathbb{R}} \text{Until}_{(0,+\infty))}(\phi_1, \neg\phi_1 \wedge \phi_2)$ we clearly have a contradiction, as the latter requires in particular that $b(y) \models_{\mathbb{R}} \phi_1$.

Next, let us realize that the facts $b(t) \models_{\mathbb{R}} \text{Lasts}_{\text{ee}}(\phi_1, a)$ and $b(t) \models_{\mathbb{R}}$

$\text{Lasts}_{\text{ee}}\big(\text{Until}_{(0,+\infty))}(\phi_1, \neg\phi_1 \wedge \phi_2), a\big)$ holding together require that $b(t + a) \models_{\mathbb{R}} \text{Until}_{[0,+\infty))}(\phi_1, \neg\phi_1 \wedge \phi_2)$. In fact, ϕ_1 cannot become false before $t + a$, but it *must* become false somewhere after (or at) $t + a$ to satisfy $\text{Until}_{(0,+\infty))}(\phi_1, \neg\phi_1 \wedge \phi_2, a)$. For the same reason, ϕ_1 must hold until it becomes false after or at $t + a$. All this is exactly what is required by $b(t + a) \models_{\mathbb{R}} \text{Until}_{[0,+\infty))}(\phi_1, \neg\phi_1 \wedge \phi_2)$.

Next, let us consider the consequences of this fact: it means that there exists a $u' \in [t + a, +\infty)$ such that $b(u') \models_{\mathbb{R}} \neg\phi_1 \wedge \phi_2$ and for all $v' \in [t + a, u')$ it is $b(v') \models_{\mathbb{R}} \phi_1$.

Let us first distinguish the case $u' \in [t + a, t + b\rangle$. All in all, ϕ_1 holds over the interval (t, u'), and ϕ_2 holds at u'; therefore, we have $b(t) \models_{\mathbb{R}} \widetilde{\text{Until}}_{[a,b\rangle)}(\phi_1, \phi_2)$.

Otherwise, let us consider the case $u' \notin [t + a, t + b\rangle$; therefore $u' \in \langle' t + b, +\infty)$, where $\langle'$ is the "complement" of $\rangle$. In particular, this implies that for all $v' \in (t, t + b)$ it is $b(v') \models_{\mathbb{R}} \phi_1$. Moreover, we are also assuming that $b(t) \models_{\mathbb{R}} \text{Until}_{[a,b\rangle)}(\text{true}, \neg\phi_1 \wedge \phi_2)$ in this branch of the proof. That is, there exists a $u'' \in [t + a, t + b\rangle$ such that $b(u'') \models_{\mathbb{R}} \neg\phi_1 \wedge \phi_2$. Since $(t, u'') \subseteq (t, t + b)$, then we have shown that $b(t) \models_{\mathbb{R}} \widetilde{\text{Until}}_{[a,b\rangle)}(\phi_1, \phi_2)$, as required. □

Proof of Formula (9.9). Let us start with the $\Rightarrow$ direction: assume that $b(t) \models_{\mathbb{R}} \widetilde{\text{Until}}_{(0,b))}(\phi_1, \phi_2)$. That is, there exists a $u \in (t, t + b\rangle$ such that $b(u) \models_{\mathbb{R}} \phi_2$ and, for all $v \in (t, u)$ it is $b(v) \models_{\mathbb{R}} \phi_1$. From $b(u) \models_{\mathbb{R}} \phi_2$ it follows immediately that $\text{Until}_{(0,b\rangle)}(\text{true}, \phi_2)$, so the first conjunct is proved.

Then, let us show that $b(t) \models_{\mathbb{R}} \epsilon\widetilde{\text{NowOn}}\big(\text{Until}_{(0,+\infty))}(\phi_1, \phi_2)\big)$. More precisely, we can show that for all $\alpha \in (t, u)$ it is $b(\alpha) \models_{\mathbb{R}} \text{Until}_{(0,+\infty))}(\phi_1, \phi_2)$. In fact, notice that $u > \alpha$, so $u \in (\alpha, +\infty)$; moreover $[\alpha, u) \subset (t, u)$, as $\alpha > t$. All in all, we have that $b(\alpha) \models_{\mathbb{R}} \text{Until}_{(0,+\infty))}(\phi_1, \phi_2)$.

Let us now consider the $\Leftarrow$ direction, and let us assume that $b(t) \models_{\mathbb{R}} \text{Until}_{(0,b\rangle)}(\text{true}, \phi_2)$. That is, there exists a $u \in (t, t + b\rangle$ such that $b(u) \models_{\mathbb{R}} \phi_2$. If, furthermore, for all $v \in (t, u)$ it is $b(v) \models_{\mathbb{R}} \phi_1$, then the left-hand side holds obviously.

Otherwise, let v' be the smallest instant in (t, u) such that $b(v') \models_{\mathbb{R}} \neg\phi_1$ or $b(v') \models_{\mathbb{R}} \epsilon\widetilde{\text{NowOn}}(\neg\phi_1)$. Recall that we are assuming as hypothesis the right-hand side of the double implication. So, from the definition of the $\epsilon\widetilde{\text{NowOn}}$ operator, we are also assuming that there exists a $\beta > 0$ such that for all $\gamma \in (t, t + \beta)$ it is $b(\gamma) \models_{\mathbb{R}} \text{Until}_{(0,+\infty))}(\phi_1, \phi_2)$. Notice that this implies that for all $\gamma \in (t, t + \beta)$ it is also $b(\gamma) \models_{\mathbb{R}} \phi_1$.

Next, let m be the minimum of the two time distances $\beta/2$ and $(v'-t)/2$. Since $t+m \in (t, t+\beta)$ we have that $b(t+m) \models_{\mathbb{R}} \text{Until}_{(0,+\infty))}(\phi_1, \phi_2)$. That is there exists a $u' \in (t+m, +\infty)$ such that $b(u') \models_{\mathbb{R}} \phi_2$ and for all $v'' \in [t+m, u')$ it is $b(v'') \models_{\mathbb{R}} \phi_1$. But since $m \leq (v'-t)/2$, then we conclude that for all instants t' in (t, u') it is $b(t') \models_{\mathbb{R}} \phi_1$.

Now, if $u' > v'$, we have a contradiction. Thus, it must be $u' \leq v'$. In this case, notice that: $u' \in (t, u) \subset (t, t+b\rangle$, which implies that $b(t) \models_{\mathbb{R}} \widetilde{\text{Until}}_{(0,b\rangle)}(\phi_1, \phi_2)$ as required. □

D.5. Analyticity and Uniform Continuity

Let us study the relations between the notions of continuity, uniform continuity, and analyticity to understand how non-Zenoness and non-Berkleyness are related for dense-valued items. In particular, we show why uniform continuity and analyticity are orthogonal notions. In the remainder, unless otherwise stated, we consider the whole real axis $\mathbb{R}$ as the domain of functions.

Analytic and not uniformly continuous. The simple polynomial function $f_1(x) = x^2$ is trivially analytic. However, its first-order derivative $f_1'(x) = x$ is not bounded, and indeed f_1 is not uniformly continuous.

Uniformly continuous and not differentiable. It is apparent from the definitions that a uniformly continuous function is also *a fortiori* continuous. However, there exist functions which are uniformly continuous but are not even differentiable (and thus *a fortiori* not even C^1-smooth and thus clearly not analytic). The following function f_2 is an example of such functions.

$$f_2(x) \quad = \quad |x - [x]|$$

where $[\cdot]$ denotes the nearest integer function:

$$[x] \quad = \quad \begin{cases} \lfloor x \rfloor & \text{if } x \leq \lfloor x \rfloor + 1/2 \\ \lceil x \rceil & \text{otherwise} \end{cases}$$

$f_2(x)$ indicates the distance from the integer which is nearest to x; as it can be seen clearly from the plot of the function if Figure D.1, f_2 is not differentiable at all integer and half-integer[3] points.

[3] I.e., points $k + 1/2$ for some $k \in \mathbb{Z}$.

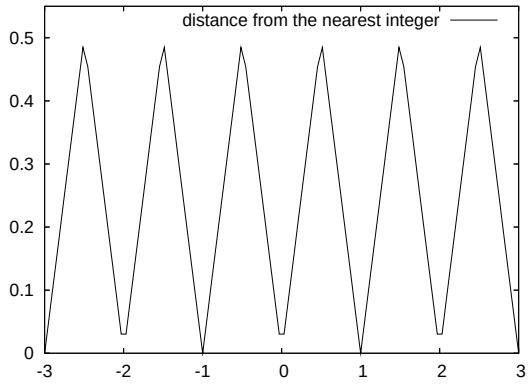

Figure D.1.: A function f_2 which uniformly continuous but not differentiable.

Uniformly continuous, C^∞-smooth, and not analytic. Let us now show an example of function which is uniformly continuous, has continuous derivatives of all orders (i.e., it is C^∞-smooth), but nonetheless fails to be analytic.

$$f_3(x) \quad = \quad \begin{cases} 0 & \text{if } x = 0 \\ e^{-1/x^2} & \text{if } x \neq 0 \end{cases}$$

Figure D.2 shows a plot of f_3, for reference.

First of all, it can be shown by induction that the n-th order derivative of f_3 is of the form:

$$f_3^{(n)}(x) \quad = \quad \begin{cases} 0 & \text{if } x = 0 \\ \rho_n(x) e^{-1/x^2} & \text{if } x \neq 0 \end{cases}$$

where $\rho_n(x)$ is some rational function without singularities in $\mathbb{R}_{\neq 0}$; therefore, each derivative is continuous over $\mathbb{R}_{\neq 0}$, and the function is infinitely differentiable. Moreover, the form of the derivative implies that its limit as x goes to zero is also zero, so all the derivatives are indeed continuous over all $\mathbb{R}$.

However, f_3 is not analytic at the origin. In fact, since $f_3^{(n)}(0) = 0$ for all n, the Taylor series of f_3 at 0 is simply the constant 0. Therefore, the series about the origin does not converge to the function.

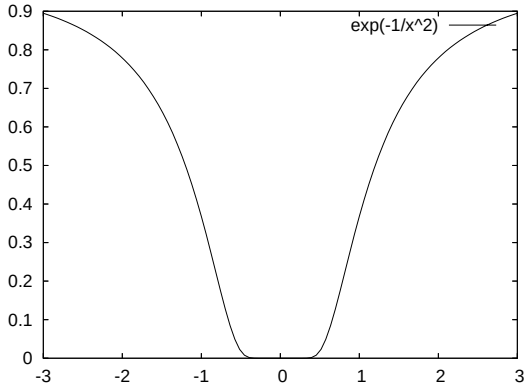

Figure D.2.: A function f_3 which is uniformly continuous and C^∞, but not analytic.

Finally, let us demonstrate that f_3 is uniformly continuous. To this end, let us just consider the first-order derivative f_3' of f_3:

$$f_3'(x) \quad = \quad \begin{cases} 0 & \text{if } x = 0 \\ \frac{2}{x^3} e^{-1/x^2} & \text{if } x \neq 0 \end{cases}$$

Without a detailed analysis, let us just look at the plot of f_3' in Figure D.3 and see that it is indeed bounded; therefore f_3 is uniformly continuous, having bounded derivative.

D.6. Axiomatization of Timed Automata

Completeness of the axiomatization. In order to show the completeness of the above axiomatization, let us consider a generic run of a given timed automaton, for which the axioms introduced in Section 9.5 have been spelled out.

$$(s_0, \nu_0) \xrightarrow{\sigma_1,\tau_1} (s_1, \nu_1) \xrightarrow{\sigma_2,\tau_2} (s_2, \nu_2) \xrightarrow{\sigma_3,\tau_3} \cdots$$

We have to show that all the above axioms are made true by the timed behavior associated with the given timed word. First of all, let us notice that axiom 9.16 is true by construction, if we consider the behavior of the start predicate associated to any behavior. Considering the other axioms,

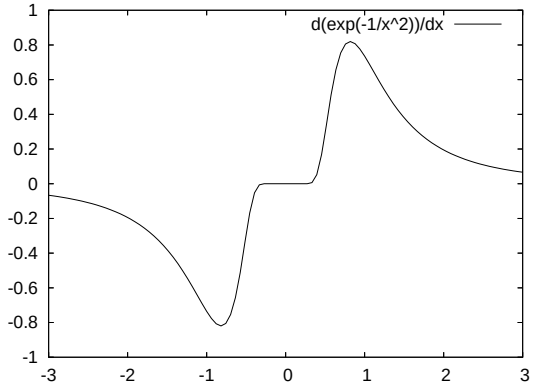

Figure D.3.: The derivative f_3' of f_3 is bounded everywhere.

since they all are implicitly universally quantified with respect to time (as every TRIO formula), we prove their truth by induction of the length of the timed word.

Base case: up to time τ_1. Without loss of generality, we assume that $\tau_1 > \delta$. Recall that:

- start is true from $-\infty$ to δ;
- $s_0 \in S_0$ is true from 0 to $\tau_1 > \delta$;
- $\mathsf{in} = \epsilon$ holds from 0 to τ_1;
- $\mathsf{rs}(c)$ holds from 0 to δ for all clocks $c \in C$.

As a consequence, all axioms (but axiom 9.14) are trivially true from $-\infty$ until δ excluded, as all their antecedents are false (since they consider an interval of length δ with their UpToNow operators). For what concerns axiom 9.14, it is also true, as at time 0 NowOn(start) holds.

Now, from time δ up to time τ_1 excluded, we have that:

- start is false;
- $s_0 \in S_0$ is true;
- $\mathsf{in} = \epsilon$ holds;

- $\mathsf{rs}(c)$ is false for all clocks $c \in C$.

Consequently, axioms 9.12–9.14 are trivially true since their antecedents are false. Axiom 9.15 is also true, since after δ its antecedent is false, whereas exactly at δ its consequent is satisfied by construction. Regarding axiom 9.17, it is also true as the state will change to s_1 at τ_1.

Inductive case: from time τ_i to time τ_{i+1} . We now show that, assuming the axioms are satisfied until time τ_i excluded, then they are also satisfied until time τ_{i+1} excluded. Let us first consider what happens exactly at time τ_i. Let $\langle s_{i-1}, s_i, \sigma_i, \Lambda_i, \xi_i \rangle \in E$ be the transition that is taken; therefore:

- start is false;
- UpToNow($\mathsf{st} = s_{i-1}$) is true;
- NowOn($\mathsf{st} = s_i$) is true;
- $\mathsf{in} = \sigma_i$ holds from τ_i to $\tau_i + \delta$;
- $\mathsf{rs}(c)$ holds from τ_i to $\tau_i + \delta$ for all clocks $c \in \Lambda_i$.

Therefore, axiom 9.12 holds, as the symbol σ_i is inputed, the right clocks are reset and the condition Ξ is met by construction, thus making true both sides of the implication. Similarly for axioms 9.13 and 9.14, both sides of their implications are true. Axiom 9.15 holds trivially as start is false, whereas axiom 9.17 is satisfied for $s = s_i$.

From time τ_i until time τ_{i+1}, we have that:

- start is false;
- s_i is true;
- $\mathsf{in} = \sigma_i$ holds until $\tau_i + \delta$, and $\mathsf{in} = \epsilon$ holds afterward;
- $\mathsf{rs}(c)$ for all clocks $c \in \Lambda_i$ holds until $\tau_i + \delta$, and $\mathsf{rs}(c)$ is false for all clocks $c \in C$ afterward.

Thus, all axioms are trivially true, their antecedents being false, except for axiom 9.17, which is however satisfied for $s = s_{i+1}$.

Correctness of the axiomatization. Conversely, in order to assess the correctness of the axiomatization, we have to show that any timed behavior satisfying all the axioms corresponds to a run which is a valid run of the automaton. This is also shown by induction on the timed behavior.

Base case: up to time τ_0. Let us first of all consider axiom 9.16. Let t_s be the time instant at which AlwP(start) holds. Because of the condition $\chi_\circ$, it must be $t_s \geq \delta$. Moreover, axiom 9.16 also implies that start cannot be always true, but it must exist an instant $\tau_0 \geq t_s$ where start changes its value from true to false; more precisely, still because of the regularity conditions, it must be UpToNow(start) and NowOn($\neg$start) at τ_0.

Let us consider axiom 9.15 at τ_0. Since AlwP($\mathsf{st} = s$), AlwP($\mathsf{in} = \epsilon$), and AlwP($\mathsf{rs}(\lambda)$) for all clocks λ, the value of the state, input, and resets is completely defined until τ_0. Therefore, we realize that this corresponds to the first element $(s, 0)$ of a valid timed word, which just specified the reset conditions of the automaton.

Inductive case: from time τ_i to time τ_{i+1}. The state s_i holds until time τ_i by inductive hypothesis. It cannot be that $\mathsf{st} = s_i$ holds forever, because of axiom 9.17. More precisely, thanks to the regularity of the behaviors, it is well defined a time instant $\tau_{i+1} > \tau_i + \delta > \tau_i$ such that st changes its value from s_i to some s_{i+1} at τ_{i+1}. Let us now trace the run of the automaton from time τ_i to τ_{i+1}.

Let us first notice that start is constantly false, as it has been so since time τ_0, due to axiom 9.16.

Exactly at time τ_i, it might be that NowOn($\mathsf{in} = \sigma$) for some input character σ. This would correspondingly set the resets $\mathsf{rs}(\lambda)$ for the appropriate clocks λ's; this behavior would of course be compatible with both axioms 9.12 and 9.14. Otherwise, if NowOn(ϵ) at τ_i, notice that there would be no resets from time τ_i to time $\tau_i + \delta$.

Now, notice that in must equal ϵ for all remaining time instants, up to τ_{i+1}. In fact, if this would not be the case, then we should record a state transition at that time, because of axiom 9.13. Similarly, by axiom 9.14, $\mathsf{rs}(c)$ must be false for all clocks c, otherwise a state transition should also occur (as we already noted, NowOn(start) is false).

Correspondingly, we can add the next step in the timed word $(s_{i-1}, \nu_{i-1}) \xrightarrow{\sigma, \tau_i} (s_i, \nu_i)$, compatibly with the timed behavior of the axiomatized automaton.

E. Integration and Metric Temporal Logics: Related Work

This chapter reviews several works broadly related with the notion of integration, and with the problem of expressiveness of metric temporal logics close to $\frac{\mathbb{R}}{\mathbb{Z}}$TRIO.

In particular, Section E.1 presents other notions of discretization and integration, different from both our notion of sampling and Henzinger, Manna, and Pnueli's digitization. Section E.2 collects several recent results about the expressiveness of (metric) temporal logics of expressive power close to $\frac{\mathbb{R}}{\mathbb{Z}}$TRIO. Section E.3 presents several formalisms that are variations of timed automata, and summarizes results about their relative expressiveness. Finally, Section E.4 reports about expressiveness and decidability issues concerning formalisms other than "standard" metric temporal logics and timed automata.

E.1. Other Notions of Integration and Discretization

A straightforward way to compose continuous-time and discrete-time models is to integrate discrete models into continuous ones, by introducing some suitable conventions. This is the approach followed by Fidge in [Fid99], with reference to timed refinement calculus. The overall simplicity of the approach is probably its main strength; nonetheless we notice that integrating everything into a continuous-time setting has some disadvantages in terms of verification complexity, as continuous time usually introduces some peculiar difficulties that render verification less automatizable. On the contrary, our approach wants to achieve a sort of *equivalence* between discrete-time and continuous-time descriptions, so that one can resort to the simpler discrete time when verifying properties, while still being able

to describe naturally physical systems using a full continuous-time model.

Hung and Giang [HG96] define a *sampling semantics* for Duration Calculus (DC), which is a formal way to relate the models of a formula to its discrete observations, approximating DC's integrals of state variables by sums. The notion is based on the notion of digitization [HMP92].

As expected, for general DC formulas the standard semantics and the sampling semantics differ. Hence, the authors derive some inference rules that outline what is the relation between the two semantic models under certain assumptions. In particular, the main aim is to formulate sufficient conditions that guarantee the soundness of *refinement* rules, that permit to move from a continuous-time description to a sampled one, thus moving from specification to implementation.

Notice that our framework deals instead with integration *of* continuous and discrete, which, in a sense, coexist within the same formal model. On the contrary, [HG96] focuses on refinement *from* continuous *to* discrete towards implementation.

The general problem of integrating different temporal (and modal) logics is studied by Chen and Liu in [CL04] in a very abstract setting. Their solution is based on the introduction of a very general semantic structure, called *resource cumulator*, that can accommodate several concrete semantics and their respective modalities. The framework is then based on *refinement* rules that permit to translate from one logic language to another one, while mapping appropriately the corresponding resource cumulators. The approach seems indeed general and abstract, even if we believe that one may desire more intuitiveness and simplicity of use from that. Moreover, it focuses on refinement among semantically different temporal logic specifications, while our framework focuses on the different problem of integrating different semantics at the same level of abstraction, thus stressing modularity and reuse of specifications.

Asarin, Maler and Pnueli in [AMP98] tackle the problem of discretizing the behavior of digital circuits in a way that preserves the qualitative behavior of the circuit in a strict sense, that is such that the ordering of events in continuous time must be unchanged in the discretized behavior. In particular, this implies that events occurring at different times, however close, must occur at *distinct* discrete time instant, in the same

order. This is a stronger notion than — in particular — those by Henzinger et al. [HMP92], where a strictly monotonic sequence is allowed to be discretized into a weakly monotonic one.

More precisely, the authors are basically considering Boolean signals, that is interval-based behaviors over infinite time (i.e., the time domain is $\mathbb{R}_{\geq 0}$). The same authors have shown in [MP95] that the subclass signals that are definable by their digital circuits can also be accepted by timed automata.

The results about digitizability are demonstrated by geometric arguments about polyhedra. While for the simpler acyclic circuits digitizability is always possible for some suitable choice of discretization step δ, for general cyclic circuits the distance between state changes can become arbitrarily small, so that discretization is impossible for any choice of step $\delta > 0$. On the contrary, the same cyclic circuits are discretizable according to the weaker notion of discretization introduced in [HMP92] (see Section 9.4).

E.2. Temporal Logics

A detailed summary about expressiveness, decidability and complexity results for real-time temporal logics is given by Henzinger [Hen98]. In the remainder, we report on some (more or less) recent result about expressiveness for temporal logic that is not dealt with extensively in [Hen98].

E.2.1. Nesting Operators in Discrete-Time Linear Temporal Logic

The problem of the expressiveness induced by the nesting depth of operators in LTL (with the "traditional" interpretation over discrete time) has been thoroughly studied by Wilke et al. in various works; see [Wil99] for a survey of related results.

In particular, Etessami and Wilke [EW96] have shown that the hierarchy of LTL formulas defined by the maximum depth of nested *until* operators (independently of the number of *next* operators) is expressively strict, and the same holds for LTL formulas where one evaluates the depth of nested *until* and *since* operators. The proof is constructive and based on game-theoretic techniques. More exactly, it is shown that, for all $k \geq 2$ the property denoted as STAIR_k, indicating the strings over the ternary al-

phabet $\Sigma = \{a, b, c\}$ such that there is a substring where a occurs k times but no b occurs, is expressible with LTL formulas with $k-1$ nested *until* operators, but it is not expressible with fewer than $k-1$ nestings. We note that the given LTL formula for STAIR_k uses $k-1$ nested *next* operators as well. A similar result holds if we allow the past *since* operator as well.

Thérien and Wilke further characterize the so-called "*until-since* hierarchy" of LTL in [TW04] by algebraic means. Without describing the details which are quite convoluted and technical, let us just mention that any LTL property is expressible with a formula with nesting depth of k if and only if the (syntactic) semigroup associated with the property (considered as a formal language) belongs to a certain (decidable) class of finite semigroups.

Kučera and Strejček [KS05] tackle similar problems as [EW96], but from a more "practical" (and less algebraic) perspective. [KS05] can be seen as a nice generalization of previously known results about the *stuttering invariance* of LTL. Let us start by noting that we consider future-only LTL, which basically consists of all formulas expressible with the two modalities *next* (X) and *until* (U). Note that we consider a *weak until*, i.e., one which does include the current instant; therefore *next* is not reducible to it. For this definition of LTL, we define a *strict syntactic hierarchy* of formulas, according to the maximum number of nested *until* and *next* operators in a formula: $\text{LTL}(\mathsf{U}^m, \mathsf{X}^n)$ denotes the set of all LTL formulas where the nesting depth of *until* is at most m, and the nesting depth of *next* is at most n. If one of the two superscript is omitted, we place no restrictions on the nesting depth of that operator: for instance $\text{LTL}(\mathsf{U}^m, \mathsf{X})$ denotes the set of all formulas where there are at most m nested *until*s, and any number of nested *next*s. The natural question, answered affirmatively in [KS05], is whether the above hierarchy is also *semantically strict*. The answer constitutes an extension of the classical results by Lamport [Lam83], and Peled and Wilke [PW97] about stutter-free languages.

In order to state the main result of the paper, we have to briefly introduce the notion of (m, n)-stutter equivalence. Let $\Sigma^\omega \ni \alpha = \alpha_0\alpha_1\alpha_2\cdots$ an ω-word of alphabet Σ. For $m, n \geq 1$, a finite subword $\beta = \alpha_i\alpha_{i+1}\cdots\alpha_{i+(j-1)}$ of α is (m, n)-redundant whenever the subword $\alpha_{i+j}\alpha_{i+j+1}\cdots$ $\alpha_{i+j+(mj-(m-n-1))-1}$ is a prefix of β^ω. Informally, the subword β is redundant if it is repeated "basically" $m+1$ times, except for a suffix of length $m-1-n$ which can be ignored. We naturally introduce a partial order relation between words according to whether one can be obtained from another by removing some (even infinitely many) non-overlapping

(m, n)-redundant subwords. The partial order induces a least equivalence among words, which we call (m, n)-stutter equivalence.

Let us now state the main results of the paper.

- every LTL($\mathsf{U}^m, \mathsf{X}^n$) language is closed under (m, n)-stutter equivalence;
- a language is definable in LTL(U, X) iff it is (m, n)-stutter closed for some $m, n \geq 1$;
- the following results are corollaries of the above main facts, but we report them explicitly for their significance:
  - for all $n \geq 1$, the LTL($\mathsf{U}^0, \mathsf{X}^n$) formula $\overbrace{\mathsf{X}\mathsf{X}\cdots\mathsf{X}}^{n \text{ times}} p$ cannot be expressed in LTL($\mathsf{U}, \mathsf{X}^{n-1}$);
  - for all $m \geq 1$, the LTL($\mathsf{U}^m, \mathsf{X}^0$) formula $\mathsf{F}(p_1 \wedge \mathsf{F}(p_2 \wedge \cdots \wedge \mathsf{F}(p_{m-1} \wedge \mathsf{F}p_m)\cdots))$ cannot be expressed in LTL($\mathsf{U}^{m-1}, \mathsf{X}$);[1]
  - for all $m, n \geq 1$, the classes LTL($\mathsf{U}^{m-1}, \mathsf{X}^n$) and LTL($\mathsf{U}^m, \mathsf{X}^{n-1}$) have incomparable expressive power.

E.2.2. (Un)Decidability and Expressiveness of MTL

Ouaknine and Worrel investigate in [OW05] some questions on the decidability of the temporal logic MTL with a point-based semantics. It is known that MTL is undecidable for interval-based semantics, as it has been discussed in [AH92b, AH93, Hen98]. In [AFH96] it has been demonstrated that the key property of MTL that renders it undecidable is its possibility of expressing punctuality, that is *exact* time distances between events. Due to the denseness of the time domain, undecidability carries over to finite interval-based semantics (see also [DP06]).

Somewhat surprisingly, things change when considering a point-based (timed word) semantics. First of all, notice that if we allow past operators in the language (i.e. we consider MTL$^{\mathrm{P}}$), undecidability stays (supposedly, with finite domains as well) [Hen91]. As pointed out in [OW05, Footnote 3], this is a consequence of the undecidability proof in [AH93,Hen91], which was carried out in a monadic second-order theory of event sequences, which subsumes both future and past operators. However, the authors show that MTL over finite timed words is instead decidable.

[1] Assuming the usual definition for the *eventually* operator: $\mathsf{F}p \equiv \top \mathsf{U} p$

The proof goes through the use of timed alternating automata (TAA, also introduced in [LW05]), that are an extension of the traditional idea of alternating automata to the timed case. More precisely, let us consider TAA with just one clock; this restriction is necessary in order to attain decidability, since the emptiness problem is undecidable for general TAA, but decidable for single-clock ones. Notice that, since the class TAA is closed under all Boolean operations (and complementation, in particular), the universality problem is also decidable for one-clock TAA. Then, the authors show how to construct a one-clock TAA that accepts the same language as that of any given MTL formula. By the way, notice that the construction exploits strict semantics [Hen98] for the *until* operator of MTL, but it is easy to pass to the weak semantics while keeping the same results. Finally, notice that, again due to closure under complementation of MTL formulas, the model-checking problem is equivalent to satisfiability.

[OW05] also discusses additional results, among which we only mention the complexity of deciding MTL formulas over finite timed words. It is shown to be nonprimitive recursive (this means that, while being recursive, the complexity measure grows faster than any primitive recursive function; it also implies that it is nonelementary — as elementary languages are a strict subset of primitive recursive ones).

The decidability of full MTL over infinite timed words is settled in [OW06], again by Ouaknine and Worrel. The result is a negative one, so the decidability results over finite timed words [OW05] do not carry over to the infinite case. More precisely, it is shown that both satisfiability and model-checking (against timed automata) are undecidable for MTL over infinite-length timed words, even if we limit ourselves to the syntactic fragment with constrained *next*[2], *eventually*, and *always* operators. As a side result, combining these results with the open problems about one-clock alternating timed automata discussed in [OW05], one also gets that the emptiness problem for one-clock timed alternating automata with weak parity acceptance conditions is undecidable; *a fortiori*, this implies the undecidability of the same problem for the more expressive one-clock timed alternating automata with Büchi acceptance conditions.

The main results of the paper are drawn by showing how to translate satisfiability questions for MTL (for both infinite and finite point-based

[2]For timed words, this means referencing to the next timestamped symbol in the word.

semantics) formulas into recurrent-state[3] and halting problems for a particular class of state machines known as *insertions channel machines with emptiness testing*, or ICMETs for short. ICMETs are queue machines with finite-state control and with insertions errors, that is where channels (where data is accessed with a FIFO policy as in a queue) may nondeterministically insert or remove symbols. Channel machines without insertion errors are as expressive as Turing machines; faulty channel machines are instead less powerful, to the extent that their halting and recurrent-state problems are decidable with polynomial-time complexity. By endowing faulty channel machines with emptiness testing (i.e., the ability to test whether a channel is empty), we get a class of automata of intermediate power: their halting problem is decidable (albeit with non-primitive recursive complexity), whereas their recurrent-state problem is still undecidable. Therefore, the aforementioned correspondence between MTL and ICMETs immediately translates to the stated (un)decidability results.

D'Souza and Prabhakar in [DP06] compare the expressiveness of MTL in the point-based and interval-based (or, as they call it, "continuous") semantics over finite-length interpretations. It is known that MTL is undecidable over interval-based semantics [AH93, AFH96], whereas it is decidable over finite-length point-based (i.e., finite timed words) semantics [OW05]. The authors exploit this difference to show that, over finite-length interpretations, interval-based MTL is more expressive than point-based MTL. Let us add some detail about how they get to the result.

First of all, they introduce a point-based semantics (they call it *pointwise* MTL, and denote it as $\mathrm{MTL}^{\mathrm{pw}}$), and a *continuous* semantics (denoted as $\mathrm{MTL}^{\mathrm{c}}$) which is different from the traditional interval-based semantics. In fact, both semantics interpret MTL formulas over finite-length timed words, so that the two semantics can be put on a common ground and compared. The difference is that, while in pointwise semantics each formula can predicate only about instants of time that correspond to events in the word, in the continuous semantics formulas assert at all instants of time, including in between any two timestamped events.

It is fairly easy to show that $\mathrm{MTL}^{\mathrm{c}}$ is at least as expressive as $\mathrm{MTL}^{\mathrm{pw}}$. In order to show that the inclusion is strict, the authors consider the undecidability proof for $\mathrm{MTL}^{\mathrm{c}}$, which relies on encoding deterministic 2-counter machines. Then, the separation proof proceeds by contradiction: if one as-

[3] That is, determining whether a state is visited infinitely often in a computation.

sumes that $\mathrm{MTL}^{\mathrm{pw}}$ is as expressive as $\mathrm{MTL}^{\mathrm{c}}$, then it must be possible to perform the same encoding of 2-counter machines in $\mathrm{MTL}^{\mathrm{pw}}$. But then, the satisfiability problem for $\mathrm{MTL}^{\mathrm{pw}}$ would undecidable, a contradiction of a previously known result [OW05].

Bouyer et al. [BCM05] analyze the expressiveness of the two well-known real-time temporal logics TPTL [AH94] and MTL [Koy90, AH93]. More precisely, they carefully analyze a long-held conjecture [AH92b,AH93,Hen98] about TPTL being more expressive than MTL over dense time.

The main contribution paper is an in-depth proof of the conjecture for both the point-based and the interval-based semantics. However, it is shown that the formula $\Box(\mathsf{p} \Rightarrow x.\Diamond(\mathsf{q} \wedge \Diamond(\mathsf{r} \wedge x \leq 5)))$ proposed in [AH93] as a witness of the conjecture is actually not expressible in MTL *only* in the point-based semantics. The authors, however, provide a different witness (namely, the formula $\Box(\mathsf{p} \Rightarrow x.\Diamond(\mathsf{q} \wedge x \leq 1 \wedge \Box(x \leq 1 \Rightarrow \neg \mathsf{r})))$) which separate the expressiveness classes in the interval-based semantics as well. The result is very technical and quite convoluted (as most expressiveness results are), so we do not report here the low-level details, for simplicity.

Let us remark, however, that both families of models that separate TPTL and MTL in the point-based or interval-based semantics are (simply) characterizable using *past* operator of MTL's. Therefore, a corollary of the expressiveness results for TPTL and MTL is that $\mathrm{MTL}^{\mathrm{P}}$ (i.e., MTL with past operators) is strictly more expressive than future-only MTL in both the point-based and the interval-based semantics.

Finally, it is shown that the existential fragments of TPTL and MTL — defined as the fragments where the only allowed modality is $\Diamond$ — are fully decidable, and the two language fragments are *equally* expressive.

Prabhakar and D'Souza present several results about the expressiveness of MTL, and syntactic and semantic variants thereof, in [PD06]. Several of the results use similar techniques as those in [BCM05], while generalizing and extending them to handle other cases.

The authors consider four basic semantics for the temporal logic languages they analyze: point-based semantics (called *pointwise* in the paper), a variant of interval-based semantics (called *continuous* semantics; basically it consists of an interval-based semantics where all items are constrained to behave as instantaneous *events*. This is the same semantics as in [DP06].), and both infinite- and finite-length variants of the two seman-

tics. Concerning syntax, they consider three variants of the MTL language: one is "regular" MTL (i.e., with the constrained *until* as only modality); then we have MTL_S, which is MTL enriched with the *unconstrained since* operator for the past; and finally we have MTL_{S_I}, which is MTL with both constrained *until* and constrained *since* operators.

Before succinctly summarizing the results of the paper, let us briefly present a preliminary result which is used in the paper to extend known results about the expressiveness of MTL over infinite behaviors (and in particular, those in [BCM05]) to finite-length behaviors as well. The result, which holds over both the pointwise and the continuous semantics, is a sort of *counter-freeness* property of MTL, which can be regarded as an analogue of the counter-freeness characterization of LTL over discrete time domains [MP72] (see also [Wil99] for a brief history of the subject). Informally, the result can be stated as follows. Given an (infinite) sequence of finite timed words $\sigma_0, \sigma_1, \sigma_2, \ldots$, which are *periodic* (i.e., one can be obtained from a previous one by "pumping" a fixed finite subword), we say that the sequence *ultimately satisfies* an MTL formula ϕ if from some index $k \geq 0$ all words in the sequence satisfy ϕ. If, from some index $k \geq 0$, all words in the sequence do not satisfy ϕ, we say that the sequence *ultimately does not satisfy* ϕ. The counter-freeness property says that, given an MTL formula ϕ, all periodic sequences of finite timed words either ultimately satisfy or ultimately do not satisfy ϕ. This characterization is useful in extending expressiveness results for MTL to finite words.

Finally, here it is a short summary of the expressiveness results of the paper.

- MTL in the continuous semantics is strictly more expressive then MTL in the pointwise semantics, for both finite and infinite behaviors; here, the proof for infinite behaviors of [BCM05] is extended to the finite case;
- in the continuous semantics, MTL is strictly contained in MTL_S, for both finite and infinite behaviors; again, the proof for infinite behaviors of [BCM05] is extended here to the finite case (in accordance with the results in [DP06]);
- the language MTL in the continuous semantics is strictly less expressive than the language MTL_S, over both finite and infinite behaviors;
- the language MTL_S is strictly less expressive than MTL_{S_I} in the pointwise semantics, over both finite and infinite behaviors; note that

whether the same inclusion is strict in the continuous semantics is still unknown.

E.2.3. Decision Techniques for MITL

Maler, Nickovic and Pnueli consider in [MNP05] the language $\mathrm{MITL}^{\mathrm{P}}_{[a,b]}$, which is a bounded future-and-past version of the temporal logic MITL [AFH96]. More precisely, it is MITL enhanced with past modalities (i.e., the bounded *since* operator), with the restriction to temporal intervals of the form $[a, b]$, with $0 \leq a < b$ such that $a, b \in \mathbb{N}$. The language $\mathrm{MITL}^{\mathrm{P}}_{[a,b]}$ is interpreted over finite-length Boolean signals (also called state sequences), that is behaviors b that are (total) functions from $[0, r)$ to $\mathbb{B}^n$ for some $r \in \mathbb{N}$, with non-Zeno requirements. Notice that the restriction to natural numbers is basically the same as the common restriction to rationals, after a proper scaling. Moreover, since we consider finite-length signals, the restriction to finite intervals in MITL formulas is also irrelevant. Thus, we are basically dealing with the same logic as in [AFH96], except that we enhance it with past modalities and restrict it to finite-length state sequences.

About $\mathrm{MITL}^{\mathrm{P}}_{[a,b]}$ with the finite-length interval-based semantics, the authors prove the following results:

- past-$\mathrm{MITL}^{\mathrm{P}}_{[a,b]}$ is deterministic, that is for every past-$\mathrm{MITL}^{\mathrm{P}}_{[a,b]}$ formula π one can construct a *deterministic* timed automaton accepting exactly the behaviors satisfying π;

- future-$\mathrm{MITL}^{\mathrm{P}}_{[a,b]}$ is nondeterministic, that is there exist future-$\mathrm{MITL}^{\mathrm{P}}_{[a,b]}$ formulas ϕ that are accepted by some nondeterministic timed automaton but by no deterministic timed automaton; notice that this implies *a fortiori* the same result for MITL over infinite *interval* sequences.

The reason for this asymmetry turns out to be related to the fact that past formulas have a syntactic idiosyncrasy that guarantees that "fast" changes (i.e., changes whose duration is less than the size of the smallest interval appearing in formulas) can be forgotten as they do not influence the truth of past formulas. This is also due to the fact that we avoid punctual intervals (this is MITL's fundamental feature), otherwise the asymmetry between past and future would not arise.

Finally, the authors point out that determinizability may be obtained by "enforcing some minimal delay of sub-formulas that imply something toward the future"; this requirement is captured by our notion of behavior constraint for formulas in normal form [FR05], or, equivalently, by the inertial bi-bounded delay operator of Maler and Pnueli [MP95].

Maler, Nickovic and Pnueli in [MNP06] consider once more the language MITL, and propose a translation of MITL formulas to timed automata alternative to the first one proposed by Alur et al. in [AFH96]. Their alternative translation is simpler and is therefore much more amenable to implementation than the original one.

In accordance with [AFH96], the authors choose an infinite-length interval-based semantics of signals. With respect to the semantics assumptions, there are only two departures from what chosen in [AFH96]. First, they disallow punctual behaviors in signals; more precisely, they consider behaviors over $\mathbb{R}_{\geq 0}$ that are constant over left-closed right-open intervals. Second, they consider a variation of the (bounded) until operator such that whenever $p \mathsf{U} q$ is true, then p must hold as well when q holds in the future, not just before. The authors claim that both assumptions simplify the exposition without appreciable sacrifice of generality.

Correspondingly, the authors show how to build a *timed signal transducer* for any MITL formula. It is a nondeterministic — as future MITL operators are in general nondeterministic [MNP05] — timed automaton that takes as input a signal, representing a behavior of primitive items, and outputs a Boolean signal corresponding to the truth value of the formula over time for the given input. The construction is based on two basic ideas. The first is *modularity*: the structure of the automaton mirrors closely the structure of the formula it models. Moreover, the construction builds on similar work for untimed automata [KP05]. The second in an observation, also made in [AFH96], that Boolean signals representing the truth over time of any MITL formula have the property of holding their truth values for at least δ time units, where δ is the size of the smallest interval appearing in the formula. This property is crucial in ensuring that the testing the truth of signals that range over a continuous domain can be done with finitely many clock, even if, *a priori*, a signal could change its value an infinite number of times in any finite interval.

E.2.4. Expressive Completeness of Future Linear Temporal Logic

Hirshfeld and Rabinovich consider in [HR03] the problem of finding a (untimed) temporal logic which is expressively complete for the future fragment of the monadic logic of order (MLO).

MLO is traditionally considered the standard setting in which ordering properties about linear sequences of events can be expressed. Similarly, Linear Temporal Logic is traditionally considered a language suitable to express the same properties as with MLO, but in a much more natural and intuitive way. In other words, the use of modalities in place of first-order logic with ordering predicates can be considered as a way of "syntactic-sugaring" the underlying MLO. The sugaring is desirable as long as the modal logic language is expressively complete with respect to MLO. For LTL the expressive completeness has been first proved by Kamp [Kam68]. Kamp's result holds for LTL with the *until* and *since* modalities, for MLO interpreted over Dedekind-complete structures[4]; in particular the standard models of natural and nonnegative real numbers — with respect to standard ordering — are comprised in Kamp's result.

[HR03] considers the derived problem of finding a temporal logic, using only future operators, which is expressively complete for the *future fragment* of MLO, which is characterized in a precise way. This apparently simple restriction yields nonetheless surprising results. First of all, the use of the sole *until* operator does not bring expressive completeness over the reals (whereas it is well-known that it does so over the naturals [GPSS80]). Even more surprisingly, [HR03] shows that no temporal logic with a finite number of future modal operators can be expressively complete for future properties. Note, however, that this limitation holds only for general behaviors over the reals (and rationals as well). If we restrict our analysis to finitely-variable (i.e., non-Zeno) behaviors only — as it is routinely done — then the expected result (that *until* is expressively complete) holds.

E.2.5. Decidable Metric Temporal Logics over the Real Line

Hirshfeld and Rabinovich in [HR04] (and, previously, in [HR99b,HR99a]) present some results about decidability and expressiveness of modal logics for real-time, comparing them with other well-known contributions in the same area. The followed approach is rather different than those of most —

[4] That is such that every non empty bounded subset has a lowest upper bound.

if not all — other similar works, and it is insightful in considering common problems and questions from a different perspective.

The semantic base is the canonical time model of the nonnegative real numbers. Notice that one of the main features of the work is that we do not restrict ourselves to non-Zeno behaviors over such a time model, but consider any possible behavior of Boolean-valued monadic predicates (i.e., time-dependent propositions, with another wording). From a purely mathematical point of view, it is quite natural to introduce a monadic language to express metric properties in these models. Since qualitative properties are naturally expressed with the monadic logic of order (MLO), its straightforward generalization that introduces a unary function +1 over the time sort is considered as the "standard" language in which to express quantitative temporal properties. We call this language MLO^+.

Since MLO^+ is undecidable, we seek a "sufficiently expressive" subset of it which is decidable. To do this, [HR04] starts by defining a metric modal logic with two metric modalities $\Diamond_1$, $\overleftarrow{\Diamond}_1$, in addition to the standard *until* and *since*. It is called Quantitative Temporal Logic (QTL). The formula $\Diamond_1 \mathsf{p}$ (respectively, $\overleftarrow{\Diamond}_1 \mathsf{p}$) denotes that p will hold (has held) within the next (previous) time unit. The corresponding syntactic fragment of MLO^+, for which QTL is expressively complete, is denoted as QMLO.

Although QTL may seem not very expressive, [HR04] demonstrates that this is not the case. In particular, it is at least as expressive as all known decidable real-time logics (such as MITL). Indeed, QTL is also decidable. [HR04] shows this with purely logical methods, without resorting to automata theory. In a nutshell, the basic idea is to reduce the satisfiability problem of any QTL formula to the satisfiability of another formula in *timer normal form*, using auxiliary variables. The timer normal form is made of a conjunction of a MLO (and thus non-metric) formula, and a particular QMLO formula *Timer* expressing quantitative constraints. Then, it is shown that *Timer* can be further reduced to a pure MLO formula by observing that, roughly speaking, its satisfiability is preserved by "stretchings" of the real line. Therefore, since MLO is fully decidable, QTL (and QMLO, being as expressive) is decidable as well.

Still resorting solely to logic-based methods, it is shown that the satisfiability problem for QTL is in **PSPACE**. Just like the other results, this holds both for unrestricted behaviors, and for non-Zeno (a.k.a. finitely-variable) ones.

Next, [HR04] considers possible extensions of QMLO and QTL that

are more expressive, but still decidable. In particular, let us mention the introduction of a hierarchy of modal logics, of increasing expressive power, that are all decidable: for each $n > 0$, the logic $\mathrm{QTL}(n)$ has a modality capable of expressing the fact that n predicates eventually become true (in the given order) within one time unit. Other enrichments using automata connectives and variants thereof are briefly discussed.

In this vein, the other paper [HR06] proves a conjecture by Pnueli, and a generalization of it, about the expressiveness of "counting modalities" similar to those of the logic $\mathrm{QTL}(n)$ mentioned in the previous paragraph. Even more generally, it is shown that for any temporal logic — even with infinitely many modalities — all of whose modalities are expressible in the second-order monadic logic of order (enriched with the unary addition predicate to account for metric properties) with a bounded quantifier depth, there exists some n such that the property that a predicate is true at least n times within the next unit of time is not expressible in the logic. This result is strong, even if we have to consider that several variants may open different perspectives. In particular, one may ask what happens if we allow more arithmetic than just addition by one, or, more generally, what other modalities should be considered as "basic". Finally, notice that the general result is proved for the whole real line, even if it is conjectured that it holds for the nonnegative reals as well.

Finally, [HR04] concludes with a critical discussion and comparison with existing similar works. In particular, it remarks how the two widely used ω-models of timed words and timed interval sequences lack some basic features and their customary use has refrained some basic results about more general (and natural) models from being discovered (especially unrestricted behaviors over the reals). On the other hand, the proliferation of different models and temporal logic languages for quantitative properties is mostly due to the shortcoming of such models, in the opinion of [HR04]. As an example, it is remarked how point-based models fail to satisfy intuitively perfectly reasonable properties such as $\Box \overleftarrow{\Diamond}_1 \mathtt{true}$, which is not satisfied by timed words where some pair of timestamps differ by more than one time unit. For what concerns interval-based sequences, although they represent precisely the finite-variability requirement, they are not unique representations, in that the same behavior is representable with more than one choice of interval coverage. Moreover, as it is also well-known [AFH96], the corresponding logics do not distinguish between a real and a rational model; according to [HR04] this is an indication that the accepted models need

reconsideration.

E.3. Timed Automata

This section reviews some variants of timed automata, and discusses the expressiveness of the resulting formalisms.

Alur and Henzinger consider in [AH92a] *two-way timed automata*, that is timed automata that can go back and forth on a timed word. They adopt an point-based (i.e., timed word) finite-length semantics. The basic nondeterministic version of these automata is denoted as 2NTA. Their deterministic subset is denoted by 2DTA; notice that determinism ensures that there is a unique run for any timed word, but a 2-way deterministic timed automaton may become nondeterministic when rendered as a (1-way) traditional timed automaton. Let us also introduce the notion of bounded timed automata: the class 2NTA_k is the subclass of two-way (nondeterministic) timed automata such that in every run over any timed word the automaton visits any symbol in the word at most $2k+1$ times; the class 2DTA_k is its deterministic counterpart. Notice that standard nondeterministic and deterministic timed automata (usually denoted by NTA and DTA, respectively) correspond to the classes 2NTA_0 and 2DTA_0, respectively.

The authors draw the following results:

- the automata in each class 2DTA_k give a strictly less expressive formalism than those in the class 2DTA_{k+1};
- conversely, increasing the number of nondeterministic reversals does not add expressiveness to that achieved by 2NTA_0 (i.e., NTA);
- since deterministic (two-way) timed automata are closed under all Boolean operations, while NTA are not, and it is easily seen that one.way nondeterminism can simulate any finite number of deterministic reversals, the class of languages defined by 2DTA_k for any $k \in \mathbb{N}$ is strictly contained in the class of languages defined by NTA;
- for every MITL formula ϕ, there exists an automaton in 2DTA_1 that accepts precisely the behaviors that satisfy ϕ; the converse is not true, as there exist MITL formulas that require nondeterminism (see also [MNP05]);

- let us consider $\mathrm{MITL}^{\mathrm{P}}$, that is MITL enriched with past operators. As the number of alternations in the nesting of subformulas between past and future modalities in a $\mathrm{MITL}^{\mathrm{P}}$ formula increases, the number of reversals required by an automaton in $2\mathrm{DTA}_k$ — accepting the same language as the formula — increases correspondingly. A byproduct of this result is that MITL remains decidable even if enriched with past operators;

- let us finally consider the language $\mathrm{MITL}_{=}$, the extension of MITL that allows singular (i.e., pointwise) intervals in temporal operators; remember that in [AFH96] it was shown that this introduces undecidability over infinite-time interval-based interpretations, but also see [OW05] for related results with different interpretations. The authors show that the models of any $\mathrm{MITL}_{=}$ formula can be recognized by some automaton in 2DTA, but not by any finite number of reversals (i.e., an unbounded number is required in the worst case);

- the authors also *conjecture* that unbounded deterministic two-wayness is not powerful enough to model nondeterminism, even if the latter is without reversals; they also propose a language that they believe cannot be accepted by any 2DTA, but is recognized by a NTA.

Alur, Fix and Henzinger introduce in [AFH99] three classes of related variations of timed automata: event-recording automata, event-predicting automata, and event-clock automata. The semantics they consider is over point-based sequences of finite length, that is finite timed words.

Event-recording automata are automata endowed with clock that record the time of the last occurrence of any event in the alphabet. *Event-predicting* automata have instead clock that "predict" the time of the occurrence of an event. Finally, *event-clock* automata have both recording and predicting clocks. We let ERA, EPA, and ECA denote the classes of even-recording, event-predicting, and event-clock automata, respectively.

Event-clock automata have the nice properties of being determinizable, and so are the subclasses of event-recording and event-predicting automata. Moreover, their emptiness problem is decidable, through a region construction that draws from the corresponding one for timed automata [AD94], and the languages they accept are closed under all Boolean operations. Also notice that event-clock automata are instead not closed under renaming or hiding of input symbols (contrarily to (nondeterministic) timed automata).

The authors draw the following results about the expressiveness of the various classes of automata they have introduced:

- the classes of languages recognized by automata in ERA and EPA are incomparable;
- the possibility of having both predicting and recording clocks in the same automaton (as in ECA) gives stronger expressive power than that given by union of ERA and EPA;
- ECA are strictly less expressive than nondeterministic timed automata NTA (obviously, as the former are determinizable, while the latter are not):
- ERA can be fully translated to deterministic timed automata DTA, while the converse is in general not true;
- EPA and DTA yield language classes that are incomparable;
- ECA and DTA also yield language classes that are incomparable.

Lasota and Walukiewicz introduce in [LW05] the notion of alternating timed automata (denoted as ATA). An ATA is a classic alternating automaton [CKS81,Var97] enriched with clocks, just like timed automata are an enrichment of classic automata. Note that ATA were also independently introduced by Ouaknine and Worrel [OW05]. The authors give a game-theoretic semantics for ATA, based on finite-length timed words (i.e., a point-based semantics).

Languages accepted by ATA are easily shown to be closed under all Boolean operations. Moreover, it is also simple to show that the emptiness problem is undecidable for ATA with more than one clock: the universality problem for timed automata with the same number of clocks — well-known to be undecidable [AD94] — can be easily encoded as an emptiness problem for ATA. Therefore, the paper focuses on one-clock ATA. For this class of automata, the following results are drawn.

- emptiness (and thus, as a simple corollary deriving from closure under Boolean operations, universality and language containment as well) is decidable for one-clock ATA. The proof relies on a region construction that builds a finite bisimilar transition system;
- the complexity of the emptiness problem is non-primitive recursive;

- ATA enriched with ϵ-moves become undecidable even with one clock: this is shown by reduction from the reachability problem for perfect channel systems (whereas without ϵ-moves one can only encode lossy channel systems);

- the authors also briefly hint at the fact that universality for one-clock nondeterministic timed automata over infinite words (and thus for ATA over infinite words as well) is undecidable; this is given without proof, but it is discussed in [ADOW05].

Abdulla et al. [ADOW05] study the language inclusion problem for timed automata with one clock. More precisely, they consider a point-based (i.e., timed word) semantics and study the problem of deciding, given a 1-clock TA A and another TA B (with no restrictions on B), whether every word accepted by B is also accepted by A. Previously, Alur and Dill [AD94] showed the problem undecidable for A with two or more clocks, and decidable for A without clocks.

The results drawn in the paper can be summarized as follows; they all exploit reductions from reachability problems in channel machines [AJ96, CFP96, Sch02].

- over finite-length words, the one-clock language inclusion problem is decidable with non-primitive recursive complexity. This was already shown by the same authors in [OW04];

- if we allow ϵ-transitions, the problem is undecidable even for one-clock automata over finite words;

- the problem is also undecidable for one-clock timed automata over infinite words with Büchi acceptance conditions.

E.4. Other Formalisms

A decidable monadic second-order logic. Wilke considers in [Wil94] a decidable monadic second-order logic, which is suitable to express properties of timed state sequences. The starting point is the *monadic logic of distance*, denoted by $\mathcal{L}_{\mathrm{d}}$; this is a monadic second-order language which allows one to predicate about distances between instants in timed words.

Therefore, notice that the work assumes a point-based infinite-length semantics (i.e., timed ω-words). This language is undecidable, as the same results of [AH93] carry over for $\mathcal{L}_{\mathrm{d}}$.

Thus, the author introduces a fragment of $\mathcal{L}_{\mathrm{d}}$ where the use of expressions about distance is restricted to relative distances only. The resulting fragment, called the *monadic logic of relative distance*, is denoted by $\mathcal{L}_{\underset{\mathrm{d}}{\leftrightarrow}}$. $\mathcal{L}_{\underset{\mathrm{d}}{\leftrightarrow}}$ is shown to define exactly the same class of timed languages as timed automata. Therefore, it is a logical characterization of timed automata, and its satisfiability problem is decidable (by building the corresponding timed automaton and checking for emptiness).

Being as expressive as timed automata, $\mathcal{L}_{\underset{\mathrm{d}}{\leftrightarrow}}$ is in general not closed under complement. However, if we restrict the interpretation of formulas in the language to timed words with *bounded variability* (i.e., such that there is an upper bound on the number of events that can occur in any finite-length interval), then the resulting class of languages is closed under complement. Clearly, the resulting theory is also fully decidable.

Finally, the author embeds some known temporal logics in the language $\mathcal{L}_{\underset{\mathrm{d}}{\leftrightarrow}}$, thus showing their effective decidability. The chosen logics are TL_Γ [MP93], $\mathrm{MITL}^{\mathrm{P}}$ (i.e., MITL with past operators) and an extension thereof called $\mathrm{EMITL}^{\mathrm{P}}$ (which extends $\mathrm{MITL}^{\mathrm{P}}$ with *automata operators* [WVS83]).

The regular real-time languages. Henzinger et al. [HRS98] provide a classification of (mostly decidable) real-time formalisms, interpreted over timed state sequences (i.e., for interval-based semantics). The paper summarizes previous results, and adds new contributions to the picture.

Basically, the paper identifies three levels of expressiveness. Each level is represented by one (or more) temporal logic languages, a monadic logic theory, and variants of timed automata. Let us briefly consider them.

- The lowest expressiveness level is labeled $\mathbb{R}$-*timed counter-free* ω-*regular*, and represents a fully decidable class of languages.

  The corresponding monadic theory is called $\textsc{MinMaxML}_1$; in a nutshell, it is a first-order sequential calculus over timed sequences, endowed with two basic quantifiers $\min, \max$ (with the intuitive meanings) for quantification over the time sort (as a side remark, notice that the interpretation is nonstandard, encompassing both undefined values and infinitesimal values to handle open and closed intervals in a uniform manner).

The corresponding temporal logic languages are MITL [AFH96] and SCL [RS97] (named EventClockTL in [HRS98]).

There is no corresponding automata class, as this class of languages cannot perform "modulo-counting" (hence the label *counter-free*), whereas automata can (by exploiting states).

Concerning the complexity of the decidability problem, it is nonelementary for MinMaxML$_1$, **PSPACE**-complete for SCL, **EXPSPACE**-complete for MITL. More practically, one may say that the two expressively equivalent languages MITL and SCL are orthogonal for what concerns the ease and naturalness of expressing properties of interest.

- Next, we have a class labeled $\mathbb{R}$-*timed* ω-*regular*, which is still fully decidable.

  The corresponding monadic theory is called MinMaxML$_2$; it is a second-order extension of MinMaxML$_1$, where second-order quantification over monadic predicates is allowed. Note that quantification is still restricted to predicates not occurring within the scope of a min or max quantifier.

  At the same level of expressiveness we have two equivalent extensions of both MITL and SCL that achieve *counting* for the logics. The former consists in using automata connectives (like those introduced in [WVS83]); the corresponding logics are labeled E-MITL and E-EventClockTL. The latter consists in introducing second-order quantification over predicates; a restriction (analogous to that of the monadic logic) requires that the quantified predicates do not occur in any of SCL's history or prophecy operators (denoted as $\lhd$ and $\rhd$, respectively). The resulting logics are labeled Q-MITL and Q-EventClockTL.

  The corresponding automata class if that of event-clock timed automata ECA [AFH99]; more precisely, an equally-expressive variation of them, which allows for recursive definition, is considered in [HRS98].

  Concerning the computational complexity of decidability, it is the same as that of the counter-free level of expressiveness.

- If we still increase the expressiveness of formalisms we get the class labeled *projection-closed* $\mathbb{R}$-*timed* ω-*regular*; as the name suggests,

it is achieved by introducing *projection* in our languages. The resulting theories are only positively decidable, that is satisfiability is decidable, but validity is not. This obviously implies that the corresponding formalisms are closed under positive Boolean operations, but not closed under complement,

More precisely, we introduce a monadic theory called P-MinMaxML$_2$; which is obtained by allowing outermost existential second-order quantification (i.e., projection) over monadic predicates occurring within the scope of a min or max quantifier. This monadic theory is as expressive as Wilke's $\mathcal{L}_{\overset{\rightarrow}{\text{d}}}$ theory [Wil94].

One can introduce the corresponding projection operators for the logic SCL, getting the language P-EventClockTL, and for event-clock automata, getting the automata class P-ECA. Notice that P-ECA correspond to "vanilla" timed automata [AD94], for which it is well known that emptiness is decidable, and universality is undecidable.

- Finally, one may permit full quantification over monadic predicates; the corresponding formalisms would be equivalent in expressive power to Boolean combinations of timed automata, and it would be fully undecidable.

Automata over continuous time. Rabinovich in [Rab03] considers the problem of "lifting" the classical concepts of automata theory from discrete to continuous time. Differently than most other works with similar objectives, it tries to do so in a very general and abstract way, starting from general concepts and refining them into more concrete notions. Such an approach allows one to reconsider most *ad hoc* solutions that have been provided elsewhere, and to evaluate their respective merits and shortcomings.

The fundamental entity is the signal, which is a mapping from the time domain (namely, the nonnegative reals) to some finite set Σ. Most of the times only non-Zeno signals will be considered, even if some general and useful results about general signals are also drawn. A machine is then described as a device that implements some signal-to-signal function. The author focuses on functions with the following characteristics (that we recall here informally):

- *strongly retrospective*, that is causal functions;

- *finite memory*, that is such that they distinguish their future behavior among only a finite number of classes of signal histories.

Afterward, it is shown how a non-Zeno signal can be represented by means of ω-strings (pairs of them, more precisely) and time sequences, thus introducing a description of signals which is closer to the traditional view of discrete-time automata. Notice, however, that the resulting model is different than the usual timed trace or timed interval sequences.

Then, it is shown that a function with finite memory is necessarily *speed independent*, that is it is invariant under "stretchings" (technically, order-preserving bijections) of the time axis. Therefore, every machine with finite memory is incapable of dealing with metric constraints. On the positive side, it is shown how for finite-memory strongly retrospective functions one can associate a description of a *state* which allows for a natural representation of such functions by means of finite transducers. Therefore, one can see how some familiar notions that are typically introduced as the basis for the formalization of finite-state machines can instead be derived from basic principles. Notice that this helps the understanding of some intrinsic limitations of such models.

Several other notions about signals and functions on signals are introduced and studied in [Rab03], but we do not discuss them here for brevity.

Finite automata extensions for hybrid systems. Rabinovich and Trakhtenbrot tackle the problem of extending the basic finite automaton model to deal with the description of hybrid systems in [RT97], and they do so with their typical "top-down" approach, with the profound goal of lifting some classical results about automata (or labeled transition systems), nets (i.e., systems of logic equations), and monadic logic (and its "syntactic sugar" temporal logic), which they dub the "trinity" [Tra95]. Namely, they separately consider two extensions: relativization and continuous time. Notice that we recall here the notions and results of [RT97] in a very informal, and thus also somewhat imprecise, way: our only goal is to convey an informal meaning, while we refer to [RT97] for technical, unambiguous explanations.

Relativization is a fundamental concept in theoretical computer science. To define it, a notion of product between automata and projection of an automaton with respect to an alphabet are introduced. Then, the relativization of an automaton A with respect to another automaton B (called the "oracle), denoted as A^B, is the projection over the alphabet of A which is not in the alphabet of B of the product automaton of A and B. In other

words, A^B is an automaton which "delegates" to B the computation needed for the acceptance of some "aspects" of the input.

Then it is considered which of five issues relativize, that is still hold for finite automata with access to some oracle (in general, infinite-state). The issues are:

- whether *monadic logic* is as expressive as the relativized automata;
- whether there exists a finite set of basic operators such that *nets* built out of this set of operators are as expressive as the relativized automata;
- whether relativized automata can always be made *deterministic*;
- whether relativized acceptor automata are as expressive as relativized transducer automata (that is if *uniformization* is possible);
- whether emptiness is decidable for relativized automata.

It is shown that, while the former two are retained for any oracle, the others may hold or fail depending on the particular choice of oracle.

On the other hand, it is shown that for another extension of finite automata, one that deals with continuous time without any relativization, all of the five issues are answered affirmatively for non-Zeno signals.

Finally, notice that most automata extensions that have been proposed in the literature to deal with hybrid systems, can indeed be seen as a relativization and/or continuous-time extension of classical automata, even if they have not been presented as such in the literature. In particular, timed automata [AD94] are basically a finite automata with access to a "clock" oracle.

Bibliography

[AD94] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[ADOW05] Parosh Aziz Abdulla, Johann Deneux, Joël Ouaknine, and James Worrell. Decidability and complexity results for timed automata via channel machines. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP'05)*, volume 3580 of *Lecture Notes in Computer Science*, pages 1089–1101. Springer-Verlag, 2005.

[AENT03] Nina Amla, E. Allen Emerson, Kedar Namjoshi, and Richard Trefler. Abstract patterns of compositional reasoning. In Roberto M. Amadio and Denis Lugiez, editors, *Proceedings of the 14th International Conference on Concurrency Theory*, volume 2761 of *Lecture Notes in Computer Science*, pages 431–445. Springer-Verlag, 2003.

[AFH96] Rajeev Alur, Tomás Feder, and Thomas A. Henzinger. The benefits of relaxing punctuality. *Journal of the ACM*, 43(1):116–146, 1996.

[AFH99] Rajeev Alur, Limor Fix, and Thomas A. Henzinger. Event-clock automata: A determinizable class of timed automata. *Theoretical Computer Science*, 211(1–2):253–273, 1999.

[AH92a] Rajeev Alur and Thomas A. Henzinger. Back to the future: Towards a theory of timed regular languages. In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science (FOCS'92)*, pages 177–186. IEEE Computer Society Press, 1992.

[AH92b] Rajeev Alur and Thomas A. Henzinger. Logics and models of real time: A survey. In J. W. de Bakker, Cornelis Huizing, and Willem P. de Roever, editors, *Proceedings of the Real-Time: Theory in Practice, REX Workshop*, volume 600 of *Lecture Notes in Computer Science*, pages 74–106. Springer-Verlag, 1992.

[AH93] Rajeev Alur and Thomas A. Henzinger. Real-time logics: Complexity and expressiveness. *Information and Computation*, 104:35–77, 1993.

[AH94] Rajeev Alur and Thomas A. Henzinger. A really temporal logic. *Journal of the ACM*, 41(1):181–204, 1994.

[AHLP00] Rajeev Alur, Thomas A. Henzinger, Gerardo Lafferriere, and George J. Pappas. Discrete abstractions of hybrid systems. In Antsaklis [Ant00], pages 971–984.

[AJ96] Parosh Aziz Abdulla and Bengt Jonsson. Undecidable verification problems for programs with unreliable channels. *Information and Computation*, 130(1):71–90, 1996.

[AL93] Martín Abadi and Leslie Lamport. Composing specifications. *ACM Transactions on Programming Languages and Systems*, 15(1):73–132, 1993.

[AL94] Martín Abadi and Leslie Lamport. An old-fashioned recipe for real time. *ACM Transactions on Programming Languages and Systems*, 16(5):1543–1571, 1994.

[AL95] Martín Abadi and Leslie Lamport. Conjoining specifications. *ACM Transactions on Programming Languages and Systems*, 17(3):507–535, 1995.

[ALL+06] Myla Archer, HongPing Lim, Nancy Lynch, Sayan Mitra, and Shinya Umeno. Specifying and proving properties of timed I/O automata in the TIOA toolkit. In *Proceedings of the 4th ACM-IEEE International Conference on Formal Methods and Models for Codesign (MEMOCODE'06)*. IEEE Computer Society Press, 2006.

[AM95] Martín Abadi and Stephan Merz. An abstract account of composition. In Jirí Wiedermann and Petr Hájek, editors, *Proceedings of the 20th International Symposium on Mathematical Foundations of Computer Science (MFCS'95)*, volume 969 of *Lecture Notes in Computer Science*, pages 499–508. Springer-Verlag, 1995.

[AMP98] Eugene Asarin, Oded Maler, and Amir Pnueli. On discretization of delays in timed automata and digital circuits. In Davide Sangiorgi and Robert de Simone, editors, *Proceedings of the 9th International Conference on Concurrency Theory (CONCUR'98)*, volume 1466 of *Lecture Notes in Computer Science*, pages 470–484. Springer-Verlag, 1998.

[Ang67] Ignacio Angelelli, editor. *Gottlob Frege. Kleine Schriften*. George Olms, 1967.

[Ant00] Panos J. Antsaklis, editor. *Special issue on hybrid systems: theory and applications*, volume 88 of *Proceedings of the IEEE*, 2000.

[AP93] Martín Abadi and Gordon D. Plotkin. A logical view of composition. *Theoretical Computer Science*, 114(1):3–30, June 1993.

[AS85] Bowen Alpern and Fred B. Schneider. Defining liveness. *Information Processing Letters*, 21(4):181–185, 1985.

[Ash75] Ed A. Ashcroft. Proving assertions about parallel programs. *Journal of Computer and System Sciences*, 10:110–135, 1975.

[BCC+95] M. Basso, Emanuele Ciapessoni, Ernani Crivelli, Dino Mandrioli, Angelo Morzenti, E. Ratto, and Pierluigi San Pietro. Experimenting a logic-based approach to the specification and design of the control system of a pondage power plant. In *Proceedings of the ICSE-17 Workshop on Formal Methods Applications in Software Engineering Practice*, pages 174–181, 1995.

[BCC+98] M. Basso, Emanuele Ciapessoni, Ernani Crivelli, Dino Mandrioli, Angelo Morzenti, E. Ratto, and Pierluigi San Pietro. A logic-based approach to the specification and design of the control system of a pondage power plant. In C. Tully, editor, *Improving Software Practice: Case Experiences*, Software Based Systems, pages 79–96. John Wiley & Sons, 1998.

[BCC+99] Armin Biere, Alessandro Cimatti, Edmund M. Clarke, Masahiro Fujita, and Yunshan Zhu. Symbolic model checking using SAT procedures instead of BDDs. In *Proceedings of the 36th Conference on Design Automation (DAC'99)*, pages 317–320. ACM Press, 1999.

[BCCZ99] Armin Biere, Alessandro Cimatti, Edmund M. Clarke, and Yunshan Zhu. Symbolic model checking without BDDs. In Rance Cleaveland, editor, *Proceedings of the 5th International Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS'99)*, volume 1579 of *Lecture Notes in Computer Science*, pages 193–207. Springer-Verlag, 1999.

[BCLR04] Thomas Ball, Byron Cook, Vladimir Levin, and Sriram K. Rajamani. SLAM and static driver verifier: Technology transfer of formal methods inside Microsoft. In Eerke A. Boiten, John Derrick, and Graeme Smith, editors, *Proceedings of the 4th International Conference on Integrated Formal Methods (IFM'04)*, volume 2999 of *Lecture Notes in Computer Science*, pages 1–20. Springer-Verlag, 2004.

[BCM05] Patricia Bouyer, Fabrice Chevalier, and Nicolas Markey. On the expressiveness of TPTL and MTL. In R. Ramanujam and Sandeep Sen, editors, *Proceedings of the 25th International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'05)*, volume 3821 of *Lecture Notes in Computer Science*, pages 432–443. Springer-Verlag, 2005.

[BGH+04] Richard J. Boulton, Hanne Gottliebsen, Ruth Hardy, Tom Kelsey, and Ursula Martin. Design verification for control engineering. In Eerke A. Boiten, John Derrick, and Graeme Smith, editors, *Proceedings of the 4th International Conference on Integrated Formal*

Methods (IFM'04), volume 2999 of *Lecture Notes in Computer Science*, pages 21–35. Springer-Verlag, 2004.

[BMN00] Paolo Bellini, Riccardo Mattolini, and Paolo Nesi. Temporal logics for real-time system specification. *ACM Computing Surveys*, 32(1):12–42, 2000. See comments in [FPR06].

[BMSU01] Nikolaj S. Bjørner, Zohar Manna, Henny B. Sipma, and Tomás E. Uribe. Deductive verification of real-time systems using STeP. *Theoretical Computer Science*, 253:27–60, 2001.

[CC96] Antonio Cau and Pierre Collette. Parallel composition of assumption-commitment specifications: A unifying approach for shared variable and distributed message passing concurrency. *Acta Informatica*, 3(2):153–176, 1996.

[CCPC+99] Emanuele Ciapessoni, Alberto Coen-Porisini, Ernani Crivelli, Dino Mandrioli, Piergiorgio. Mirandola, and Angelo Morzenti. From formal models to formally-based methods: an industrial experience. *ACM Transactions on Software Engineering and Methodology*, 8(1):79–113, 1999.

[CFP96] Gérard Cécé, Alain Finkel, and S. Purushothaman Iyer. Unreliable channels are easier to verify than perfect channels. *Information and Computation*, 124(1):20–31, 1996.

[CGP00] Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model Checking*. MIT Press, 2000.

[CGR95] Dan Craigen, Susan Gerhart, and Ted Ralston. Formal methods reality check: Industrial usage. *IEEE Transactions on Software Engineering*, 21(2):90–98, 1995.

[CHR02] Franck Cassez, Thomas A. Henzinger, and Jean-François Raskin. A comparison of control problems for timed and hybrid systems. In Claire Tomlin and Mark R. Greenstreet, editors, *Proceedings of the 5th International Workshop on Hybrid Systems: Computation and Control (HSCC'02)*, volume 2289 of *Lecture Notes in Computer Science*, pages 134–148. Springer-Verlag, 2002.

[CKS81] Ashok K. Chandra, Dexter Kozen, and Larry J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, 1981.

[CL04] Yifeng Chen and Zhiming Liu. Integrating temporal logics. In Eerke A. Boiten, John Derrick, and Graeme Smith, editors, *Proceedings of the 4th International Conference on Integrated Formal Methods (IFM'04)*, volume 2999 of *Lecture Notes in Computer Science*, pages 402–420, 2004.

[Coh01] Bram Cohen. BitTorrent. `http://www.bittorrent.org`, 2001.

[Coh03] Bram Cohen. Incentives build robustness in BitTorrent. In *Proceedings of the 1st Workshop on Economics of Peer-to-Peer Systems*, May 2003.

[Coo71] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing (STOC'71)*, pages 151–178. ACM Press, 1971.

[Coo78] Stephen A. Cook. Soundness and completeness of an axiom system for program verification. *SIAM Journal on Computing*, 7(1):70–90, 1978. Corrigendum in [Coo81].

[Coo81] Stephen A. Cook. Corrigendum: Soundness and completeness of an axiom system for program verification. *SIAM Journal on Computing*, 10(3):612, 1981.

[CPRS06] Pietro Colombo, Matteo Pradella, Matteo Rossi, and Giordano Sassaroli. A UML 2-compatible language and tool for formal modeling real-time system architectures. In Hisham Haddad, editor, *Proceedings of the 2006 ACM Symposium on Applied Computing (SAC'06)*, pages 1785–1790. ACM Press, 2006.

[CW96] Edmund M. Clarke and Jeannette M. Wing. Formal methods: State of the art and future directions. *ACM Computing Surveys*, 28(4):626–643, 1996.

[Das06] Manuvir Das. Formal specifications on industrial-strength code—from myth to reality. In Thomas Ball and Robert B. Jones, editors, *Proceedings of the 18th International Conference on Computer Aided Verification (CAV'06)*, volume 4144 of *Lecture Notes in Computer Science*, page 1. Springer-Verlag, 2006. (Invited talk).

[dBdR98] Frank S. de Boer and Willem-Paul de Roever. Compositional proof methods for concurrency: A semantic approach. In de Roever et al. [dRLP98], pages 632–646.

[Dij71] Edsger W. Dijkstra. Hierarchical ordering of sequential processes. *Acta Informatica*, 1(2):115–138, 1971.

[Dij72] Edsger W. Dijkstra. The humble programmer. *Communications of the ACM*, 15(10):859–866, 1972. Turing Award Lecture.

[DP06] Deepak D'Souza and Pavithra Prabhakar. On the expressiveness of MTL in the pointwise and continuous semantics. *Formal Methods Letters (Software Tools for Technology Transfer)*, 2006. To appear.

[dR85] Willem-Paul de Roever. The quest for compositionality — a survey of assertion-based proof systems for concurrent programs (part 1: concurrency based on shared variables). In *Proceedings of the IFIP Working Conference on The Role of Abstract Models in Computer Science*. North-Holland, 1985.

[dR98] Willem-Paul de Roever. The need for compositional proof systems: A survey. In de Roever et al. [dRLP98], pages 1–22.

[dRdBH+01] Willem-Paul de Roever, Frank de Boer, Ulrich Hannemann, Jozef Hooman, Yassine Lakhnech, Mannes Poel, and Job Zwiers. *Concurrency Verification: Introduction to Compositional and Noncompositional Methods*. Cambridge University Press, 2001.

[dRLP98] Willem-Paul de Roever, Hans Langmaack, and Amir Pnueli, editors. *Proceedings of the International Symposium: "Compositionality: The Significant Difference" (COMPOS'97)*, volume 1536 of *Lecture Notes in Computer Science*. Springer-Verlag, 1998.

[Dug06] Adam Duguid. Coping with the parallelism of BitTorrent: Conversion of PEPA to ODEs in dealing with state space explosion. In Eugene Asarin and Patricia Bouyer, editors, *Proceedings of the 4th International Conference on Formal Modelling and Analysis of Timed Systems (FORMATS'06)*, volume 4202 of *Lecture Notes in Computer Science*, pages 156–170. Springer-Verlag, 2006.

[Ecl] Eclipse. `http://www.eclipse.org`.

[Eme90] E. Allen Emerson. Temporal and modal logic. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 996–1072. Elsevier Science Publishers, 1990.

[EPLF03] Hans-Erik Eriksson, Magnus Penker, Brian Lyons, and David Fado. *UML 2 Toolkit*. John Wiley & Sons, 2003.

[EW96] Kousha Etessami and Thomas Wilke. An until hierarchy for temporal logic. In *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science (LICS'96)*, pages 108–117. IEEE Computer Society Press, 1996.

[Fid99] Colin J. Fidge. Modelling discrete behaviour in a continuous-time formalism. In Keijiro Araki, Andy Galloway, and Kenji Taguchi, editors, *Proceedings of 1st International Conference on Integrated Formal Methods (IFM'99)*, pages 170–188. Springer-Verlag, 1999.

[Fif98] Alessio Fifi. Un sistema per la conduzione di prove formali in TRIO basato su una codifica PVS e un'interfaccia astratta. Master's thesis, Politecnico di Milano, December 1998. (Tesi di Laurea, in Italian).

[Flo67] Robert W. Floyd. Assigning meanings to programs. In J. T. Schwartz, editor, *Mathematical Aspects of Computer Science*, volume 19 of *Proceedings of Symposia in Applied Mathematics*, pages 19–32. American Mathematical Society, 1967.

[FM94] Miguel Felder and Angelo Morzenti. Validating real-time systems by history-checking TRIO specifications. *ACM Transactions on Software Engineering and Methodology*, 3(4):308–339, 1994.

[FM05] International Symposia on Formal Methods (FM, formerly FME). Organized by Formal Methods Europe `http://www.fmeurope.org`, 1987–2005.

[FMM94] Miguel Felder, Dino Mandrioli, and Angelo Morzenti. Proving properties of real-time systems through logical specifications and Petri net models. *IEEE Transactions on Software Engineering*, 20(2):127–141, 1994.

[FMM+04] Carlo A. Furia, Dino Mandrioli, Angelo Morzenti, Matteo Pradella, Matteo Rossi, and Pierluigi San Pietro. Higher-order TRIO. Technical Report 2004.28, Dipartimento di Elettronica e Informazione, Politecnico di Milano, September 2004.

[FMMR07] Carlo A. Furia, Dino Mandrioli, Angelo Morzenti, and Matteo Rossi. Modeling time in computing: A taxonomy and a comparative survey. Technical Report 2007.22, Dipartimento di Elettronica e Informazione, Politecnico di Milano, 2007. Submitted for publication.

[FP78] Nissim Francez and Amir Pnueli. A proof methods for cyclic programs. *Acta Informatica*, 9:133–157, 1978.

[FPR06] Carlo A. Furia, Matteo Pradella, and Matteo Rossi. Comments on "Temporal logics for real-time system specification", 2006. Submitted.

[FR04] Carlo A. Furia and Matteo Rossi. A compositional framework for formally verifying modular systems. In *Proceedings of the International Workshop on Test and Analysis of Component Based Systems (TACoS'04)*, volume 116 of *Electronic Notes in Theoretical Computer Science*, pages 185–198. Elsevier, January 2004.

[FR05] Carlo A. Furia and Matteo Rossi. When discrete met continuous: on the integration of discrete- and continuous-time metric temporal logics. Technical Report 2005.44, Dipartimento di Elettronica e Informazione, Politecnico di Milano, October 2005.

[FR06] Carlo A. Furia and Matteo Rossi. Integrating discrete- and continuous-time metric temporal logics through sampling. In Eugene Asarin and Patricia Bouyer, editors, *Proceedings of the 4th International Conference on Formal Modelling and Analysis of Timed*

Systems (FORMATS'06), volume 4202 of *Lecture Notes in Computer Science*, pages 215–229. Springer-Verlag, September 2006.

[Fre23] Gottlob Frege. *Logische Untersuchungen. Dritter Teil: Gedankengefüge*, volume 3(1), chapter Beiträge zur Philosophie des Deutschen Idealismus, pages 36–51. 1923. Reprinted in [Ang67, pp. 378–394]. English translation in [Fre77].

[Fre77] Gottlob Frege. Compound thoughts. In P. Geach and N. Black, editors, *Logical Investigations*, Oxford, 1977. Blackwells.

[FRMM05a] Carlo A. Furia, Matteo Rossi, Dino Mandrioli, and Angelo Morzenti. Automated compositional proofs for real-time systems. In Maura Cerioli, editor, *Proceedings of Fundamental Aspects of Software Engineering (FASE'05)*, volume 3442 of *Lecture Notes in Computer Science*, pages 326–340. Springer-Verlag, 2005.

[FRMM05b] Carlo A. Furia, Matteo Rossi, Dino Mandrioli, and Angelo Morzenti. Automated compositional proofs for real-time systems. FASE'05 full version with appendices available online from `http://www.elet.polimi.it/upload/furia`, 2005.

[FRMM05c] Carlo A. Furia, Matteo Rossi, Dino Mandrioli, and Angelo Morzenti. TRIO/PVS proofs of the dining philosophers example. Available online from `http://www.elet.polimi.it/upload/furia` (publications section), 2005.

[FRMM07] Carlo A. Furia, Matteo Rossi, Dino Mandrioli, and Angelo Morzenti. Automated compositional proofs for real-time systems. *Theoretical Computer Science*, 2007. To appear.

[FRS+06] Carlo A. Furia, Matteo Rossi, Elisabeth A. Strunk, Dino Mandrioli, and John C. Knight. Raising formal methods to the requirements level. Technical Report 2006.64, Dipartimento di Elettronica e Informazione, Politecnico di Milano, November 2006. Also: Technical Report CS-2006-24, Department of Computer Science, University of Virginia.

[Fur03a] Carlo Alberto Furia. Compositional proofs for real-time modular systems. Master's thesis, University of Illinois at Chicago, October 2003.

[Fur03b] Carlo Alberto Furia. Compositional proofs for real-time modular systems. Master's thesis, Politecnico di Milano, December 2003. (Tesi di Laurea).

[Fur05] Carlo Alberto Furia. A compositional world: a survey of recent works on compositionality in formal methods. Technical Report 2005.22, Dipartimento di Elettronica e Informazione, Politecnico di Milano, March 2005.

[Fur06a] Carlo A. Furia. Quantum informatics: A survey. Technical Report 2006.16, Dipartimento di Elettronica e Informazione, Politecnico di Milano, January 2006.

[Fur06b] Carlo Alberto Furia. Compositionality made up. Technical Report 2006.76, Dipartimento di Elettronica e Informazione, Politecnico di Milano, December 2006.

[Fur06c] Carlo Alberto Furia. Discrete meets continuous, again. Technical Report 2006.77, Dipartimento di Elettronica e Informazione, Politecnico di Milano, December 2006.

[Gar06] Stephen J. Garland. TIOA user guide and reference manual. Technical report, Computer Science and Artificial Intelligence Laboratory, Massachussets Institute of Technology, 2006.

[GHGJ93] John V. Guttag, James J. Horning, S. J. Garland, and K. D. Jones. *Larch: Languages and Tools for Formal Specification*. Springer-Verlag, 1993.

[GHR94] Dov M. Gabbay, Ian Hodkinson, and Mark Reynolds. *Temporal Logic (vol. 1): mathematical foundations and computational aspects*, volume 28 of *Oxford Logic Guides*. Oxford University Press, 1994.

[GJ97] Carlo Ghezzi and Mehdi Jazayeri. *Programming Language Concepts*. Wiley & Sons, 3rd edition, 1997.

[GJM02] Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, 2nd edition, 2002.

[GKP94] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics: A foundation for computer science*. Addison-Wesley, 2nd edition, 1994.

[GLMZ96] Angelo Gargantini, Lilia Liberati, Angelo Morzenti, and Cristiano Zacchetti. Specifying, validating, and testing a traffic management system in the TRIO environment. In *Proceedings of the 11th Annual Conference on Computer Assurance (COMPASS'96)*, pages 65–76, 1996.

[GM93] Carlo Ghezzi and Dino Mandrioli. *Theoretical Foundations of Computer Sciences*. Krieger Publishing, 1993.

[GM01] Angelo Gargantini and Angelo Morzenti. Automated deductive requirements analysis of critical systems. *ACM Transactions on Software Engineering and Methodology*, 10(3):255–307, 2001.

[GMM90] Carlo Ghezzi, Dino Mandrioli, and Angelo Morzenti. TRIO: A logic language for executable specifications of real-time systems. *The Journal of Systems and Software*, 12(2):107–123, 1990.

[GMM99] Angelo Gargantini, Dino Mandrioli, and Angelo Morzenti. Dealing with zero-time transitions in axiom systems. *Information and Computation*, 150(2):119–131, 1999.

[GPSS80] Dov M. Gabbay, Amir Pnueli, Saharon Shelah, and Jonathan Stavi. On the temporal basis of fairness. In *Conference Record of the 7th Annual ACM Symposium on Principles of Programming Languages (POPL'80)*, pages 163–173, 1980.

[GvN63] H. H. Goldstine and John von Neumann. Planning and coding problems for an electronic computer. In A. H. Taub, editor, *Collected Works of John von Neumann*, volume Volume 5, pages 80–235. Pergamon Press, 1963. Original publication year: 1947.

[HALM06] Constance L. Heitmeyer, Myla Archer, Elizabeth I. Leonard, and John McLean. Formal specification and verification of data separation in a separation kernel for an embedded system. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS'06)*, pages 346–355. ACM Press, 2006.

[HdR86] Jozef Hooman and Willem-Paul de Roever. The quest goes on: a survey of proof systems for partial correctness of CSP. In J. W. de Bakker, Willem-Paul de Roever, and Grzegorz Rozenberg, editors, *Current Trends in Concurrency*, volume 224 of *Lecture Notes in Computer Science*, pages 343–395. Springer-Verlag, 1986.

[Hen91] Thomas A. Henzinger. *The Temporal Specification and Verification of Real-Time Systems.* PhD thesis, Department of Computer Science, Stanford University, 1991. Also Technical Report STAN-CS-91-1380.

[Hen98] Thomas A. Henzinger. It's about time: Real-time logics reviewed. In Davide Sangiorgi and Robert de Simone, editors, *Proceedings of the 9th International Conference on Concurrency Theory (CONCUR'98)*, volume 1466 of *Lecture Notes in Computer Science*, pages 439–454. Springer-Verlag, 1998.

[HG96] Dang Van Hung and Phan Hong Giang. Sampling semantics of duration calculus. In Bengt Jonsson and Joachim Parrow, editors, *Proceedings of the 4th International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT'96)*, volume 1135 of *Lecture Notes in Computer Science*, pages 188–207. Springer-Verlag, 1996.

[HM96] Constance Heitmeier and Dino Mandrioli, editors. *Formal Methods for Real-Time Computing.* John Wiley & Sons, 1996.

[HMP92] Thomas A. Henzinger, Zohar Manna, and Amir Pnueli. What good are digital clocks? In Werner Kuich, editor, *Proceedings of the 19th International Colloquium on Automata, Languages and Programming (ICALP'92)*, volume 623 of *Lecture Notes in Computer Science*, pages 545–558. Springer-Verlag, 1992.

[HMU00] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley, 2nd edition, 2000.

[Hoa69] Charles Antony Richard Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580, 1969.

[Hoa84] C. A. R. Hoare. Programming: Sorcery or science? *IEEE Software*, 1(2):5–16, 1984.

[Hoa03] C. A. R. Hoare. The verifying compiler: A grand challenge for computing research. *Journal of the ACM*, 50(1):63–69, 2003.

[Hol03] Gerard J. Holzmann. *The SPIN Model Checker: Primer and Reference Manual*. Addison-Wesley, 2003.

[Hoo94] Jozef Hooman. Correctness of real time systems by construction. In *Proceedings of the 3rd International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*, volume 863 of *Lecture Notes in Computer Science*, pages 19–40. Springer-Verlag, 1994.

[Hoo98] Jozef Hooman. Compositional verification of real-time applications. In de Roever et al. [dRLP98], pages 276–300.

[HQRT02] Thomas A. Henzinger, Shaz Qadeer, Sriram K. Rajamani, and Serdar Tasiran. An assume-guarantee rule for checking simulation. *ACM Transactions on Programming Languages and Systems*, 24(1):51–64, 2002.

[HR99a] Yoram Hirshfeld and Alexander Moshe Rabinovich. A framework for decidable metrical logics. In Jirí Wiedermann, Peter van Emde Boas, and Mogens Nielsen, editors, *Proceedings of the 26th International Colloquium on Automata, Languages and Programming (ICALP'99)*, volume 1644 of *Lecture Notes in Computer Science*, pages 422–432. Springer-Verlag, 1999.

[HR99b] Yoram Hirshfeld and Alexander Moshe Rabinovich. Quantitative temporal logic. In Jörg Flum and Mario Rodríguez-Artalejo, editors, *Proceedings of the 13th International Workshop on Computer Science Logic (CSL'99)*, volume 1683 of *Lecture Notes in Computer Science*, pages 172–187. Springer-Verlag, 1999.

[HR03] Yoram Hirshfeld and Alexander Moshe Rabinovich. Future temporal logic needs infinitely many modalities. *Information and Computation*, 187(2):196–208, 2003.

[HR04] Yoram Hirshfeld and Alexander Moshe Rabinovich. Logics for real time: Decidability and complexity. *Fundamenta Informaticae*, 62(1):1–28, 2004.

[HR06] Yoram Hirshfeld and Alexander Moshe Rabinovich. Expressiveness of metric modalities for continuous time. In Dima Grigoriev, John Harrison, and Edward A. Hirsch, editors, *Proceedings of the 1st International Computer Science Symposium in Russia: Computer Science – Theory and Applications (CSR'06)*, volume 3967 of *Lecture Notes in Computer Science*, pages 211–220. Springer-Verlag, 2006.

[HRS98] Thomas A. Henzinger, Jean-François Raskin, and Pierre-Yves Schobbens. The regular real-time languages. In Kim Guldstrand Larsen, Sven Skyum, and Glynn Winskel, editors, *Proceedings of the 25th International Colloquium on Automata, Languages and Programming (ICALP'98)*, volume 1443 of *Lecture Notes in Computer Science*, pages 580–591. Springer-Verlag, 1998.

[HS06] Thomas A. Henzinger and Joseph Sifakis. The embedded systems design challenge. In Jayadev Misra, Tobias Nipkow, and Emil Sekerinski, editors, *Proceedings of the 14th International Symposium on Formal Methods (FM'06)*, volume 4085 of *Lecture Notes in Computer Science*, pages 1–15. Springer-Verlag, 2006.

[HZB+96] Mark R. Heckman, Cui Zhang, Brian R. Becker, Dave Peticolas, Karl N. Levitt, and Ronald A. Olsson. Towards applying the composition principle to verify a microkernel operating system. In Joakim von Wright, Jim Grundy, and John Harrison, editors, *Proceedings of the 9th International Conference on Theorem Proving in Higher Order Logics (TPHOLs'96)*, volume 1125 of *Lecture Notes in Computer Science*, pages 235–250. Springer-Verlag, 1996.

[Jan98] Theo M. V. Janssen. An overview of compositional translations. In de Roever et al. [dRLP98], pages 327–349.

[Jon81] Cliff B. Jones. *Development Methods for Computer Programs Inclusing a Notion of Interference*. PhD thesis, Oxford University Computing Laboratory, 1981.

[Jon83] Cliff B. Jones. Tentative steps towards a development method for interfering programs. *ACM Transactions on Programming Languages and Systems*, 5(4):596–619, 1983.

[JOW06] Cliff Jones, Peter W. O'Hearn, and Jim Woodcock. Verified software: A grand challenge. *IEEE Computer*, 39(4):93–95, 2006.

[JT96] Bengt Jonsson and Yih-Kuen Tsay. Assumption/guarantee specifications in linear-time temporal logic. *Theoretical Computer Science*, 167(1–2):47–72, 1996.

[JT02] Eric Y. T. Juan and Jeffrey J. P. Tsai. *Compositional Verification of Concurrent and Real-Time Systems*. Kluwer Academic Publishers, 2002.

[Kam68] Johan Anthony Willem Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, University of California, Los Angeles, 1968.

[KLSV06] Dilsun K. Kaynar, Nancy Lynch, Roberto Segala, and Frits Vaandrager. *The Theory of Timed I/O Automata*. Synthesis Lectures on Computer Science. Morgan Claypool Publishers, 2006.

[Koy90] Ron Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.

[KP05] Yonit Kesten and Amir Pnueli. A compositional approach to CTL* verification. *Theoretical Computer Science*, 331(2–3):397–428, 2005.

[KS05] Antonín Kučera and Jan Strejček. The stuttering principle revisited. *Acta Informatica*, 41(7/8):415–434, 2005.

[KV00] Orna Kupferman and Moshe Y. Vardi. An automata-theoretic approach to modular model checking. *ACM Transactions on Programming Languages and Systems*, 22(1):87–128, January 2000.

[Lam77] Leslie Lamport. Proving the correctness of multiprocess programs. *IEEE Transactions on Software Engineering*, 3(2):125–143, 1977.

[Lam83] Leslie Lamport. What good is temporal logic? In R. E. A. Mason, editor, *Proceedings of the IFIP 9th World Computer Congress on Information Processing (IFIP'83)*, pages 657–668. North Holland/IFIP, 1983.

[Lam94] Leslie Lamport. The temporal logic of actions. *ACM Transactions on Programming Languages and Systems*, 16(3):872–923, 1994.

[Lam98] Leslie Lamport. Composition: A way to make proofs harder. In de Roever et al. [dRLP98], pages 402–423.

[Lee05] Edward A. Lee. Absolutely positively on time: What would it take? *IEEE Computer*, 38(7):85–87, 2005.

[Liu00] Jane W. S. Liu. *Real-Time Systems*. Prentice Hall, 2000.

[LW05] Slawomir Lasota and Igor Walukiewicz. Alternating timed automata. In Vladimiro Sassone, editor, *Proceeding of the 8th International Conference on Foundations of Software Science and Computational Structures (FOSSACS'05)*, volume 3441 of *Lecture Notes in Computer Science*, pages 250–265. Springer-Verlag, 2005.

[Mai01] Patrick Maier. A set-theoretic framework for assume-guarantee reasoning. In Fernando Orejas, Paul G. Spirakis, and Jan van Leeuwen, editors, *Proceedings of the 28th International Colloquium on Automata, Languages and Programming (ICALP'01)*, volume 2076 of *Lecture Notes in Computer Science*, pages 821–834. Springer-Verlag, 2001.

[Mai03] Patrick Maier. Compositional circular assume-guarantee rules cannot be sound and complete. In Andrew D. Gordon, editor, *Proceedings of the 16th International Conference on Foundations of Software Science and Computational Structures (FOSSACS'03)*, volume 2620 of *Lecture Notes in Computer Science*, pages 343–357. Springer-Verlag, 2003.

[MC81] Jayadev Misra and K. Mani Chandy. Proof of networks of processes. *IEEE Transactions on Software Engineering*, 7(4):417–426, 1981.

[McM99] Ken McMillan. Circular compositionl reasoning about liveness. In Laurence Pierre and Thomas Kropf, editors, *Proceedings of the 10th IFIP WG 10.5 Advanced Research Working Conference on Correct Hardware Design and Verification Methods (CHARME'99)*, volume 1703 of *Lecture Notes in Computer Science*, pages 342–345. Springer-Verlag, 1999.

[McM00] Ken L. McMillan. A methodology for hardware verification using compositional model checking. *Science of Computer Programming*, 37(1–3):279–309, 2000.

[Men97] Elliott Mendelson. *Introduction to Mathematical Logic*. Chapman & Hall, 4th edition, 1997.

[Mey97] Bertrand Meyer. *Object-Oriented Software Construction*. Prentice Hall, 2nd edition, 1997.

[MFR04] Andrea Matta, Carlo A. Furia, and Matteo Rossi. Semi-formal and formal models applied to flexible manufacturing systems. In Cevdet Aykanat, Tuğrul Dayar, and İbrahim Körpeoğlu, editors, *Proceedings of the 19th International Symposium on Computer and Information Sciences (ISCIS'04)*, volume 3280 of *Lecture Notes in Computer Science*, pages 718–728. Springer-Verlag, October 2004.

[MM97] Dino Mandrioli and Angelo Morzenti. Specifica in TRIO delle unità di elaborazione periferica: aspetti relativi alla struttura delle versioni, alla gestione della base dei tempi e al conteggio dei quanti di energia. Technical report, Dipartimento di Elettronica e Informazione, Politecnico di Milano, 1997. In Italian.

[MMG92] Angelo Morzenti, Dino Mandrioli, and Carlo Ghezzi. A model parametric real-time logic. *ACM Transactions on Programming Languages and Systems*, 14(4):521–573, 1992.

[MMM95] Dino Mandrioli, Sandro Morasca, and Angelo Morzenti. Generating test cases for real-time systems from logic specifications. *ACM Transactions on Computer Systems*, 13(4):365–398, 1995.

[MNP05] Oded Maler, Dejan Nickovic, and Amir Pnueli. Real time temporal logic: Past, present, future. In Paul Petterson and Wang Yi, editors, *Proceedings of the 3rd International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS'05)*, volume 3829 of *Lecture Notes in Computer Science*, pages 2–16. Springer-Verlag, 2005.

[MNP06] Oded Maler, Dejan Nickovic, and Amir Pnueli. From MITL to timed automata. In Eugene Asarin and Patricia Bouyer, editors, *Proceedings of the 4th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS'06)*, volume 4202 of *Lecture Notes in Computer Science*, pages 274–289. Springer-Verlag, 2006.

[Moo65] Gordon E. Moore. Cramming more components onto integrated circuits. *Electronics Magazine*, April 1965.

[MP72] Robert McNaughton and Seymour A. Papert. *Counter-Free Automata*, volume 65 of *MIT Research Monograph*. MIT Press, 1972.

[MP90] Zohar Manna and Amir Pnueli. A hierarchy of temporal properties. In *Proceedings of the 9th Annual ACM Symposium on Principles of Distributed Computing (PODC'90)*, pages 377–410. ACM Press, 1990.

[MP93] Zohar Manna and Amir Pnueli. Models for reactivity. *Acta Informatica*, 30(2):609–678, 1993.

[MP95] Oded Maler and Amir Pnueli. Timing analysis of asynchronous circuits using timed automata. In Paolo Camurati and Hans Eveking, editors, *Proceedings of the Advanced Research Working Conference on Correct Hardware Design and Verification Methods*, volume 987 of *Lecture Notes in Computer Science*, pages 189–205. Springer-Verlag, 1995.

[MPSS03] Angelo Morzenti, Matteo Pradella, Pierluigi San Pietro, and Paola Spoletini. Model-checking TRIO specifications in SPIN. In Keijiro Araki, Stefania Gnesi, and Dino Mandrioli, editors, *Proceedings of International Symposium of Formal Methods Europe (FME'03)*, volume 2805 of *Lecture Notes in Computer Science*, pages 542–561. Springer-Verlag, 2003.

[MSP94] Angelo Morzenti and Pierluigi San Pietro. Object-oriented logical specification of time-critical systems. *ACM Transactions on Software Engineering and Methodology*, 3(1):56–98, 1994.

[MW98] Merriam-Webster, editor. *Merriam-Webster's Collegiate Dictionary*. Merriam-Webster Inc., 10th edition edition, 1998.

[NT00] Kedar S. Namjoshi and Richard J. Trefler. On the completeness of compositional reasoning. In E. Allen Emerson and A. Prasad Sistla, editors, *Proceedings of the 12th International Conference on Computer Aided Verification (CAV'00)*, volume 1855 of *Lecture Notes in Computer Science*, pages 139–153. Springer-Verlag, 2000.

[OG76] Susan Owicki and David Gries. An axiomatic proof technique for parallel programs. *Acta Informatica*, 6:319–340, 1976.

[ORS92] Sam Owre, John M. Rushby, and Natarajan Shankar. PVS: A prototype verification system. In Deepak Kapur, editor, *Proceedings of the 11th International Conference on Automated Deduction (CADE'92)*, volume 607 of *Lecture Notes in Computer Science*, pages 748–752. Springer-Verlag, 1992.

[OW04] Joël Ouaknine and James Worrell. On the language inclusion problem for timed automata: Closing a decidability gap. In *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science (LICS'04)*, pages 54–63. IEEE Computer Society Press, 2004.

[OW05] Joël Ouaknine and James Worrell. On the decidability of metric temporal logic. In *Proceedings of the 20th Annual IEEE Symposium on Logic in Computer Science (LICS'05)*, pages 188–197. IEEE Computer Society Press, 2005.

[OW06] Joël Ouaknine and James Worrell. On metric temporal logic and faulty Turing machines. In Luca Aceto and Anna Ingólfsdóttir, editors, *Proceedings of the 9th International Conference Foundations of Software Science and Computation Structures (FoSSaCS'06)*, volume 3921 of *Lecture Notes in Computer Science*, pages 217–230. Springer-Verlag, 2006.

[Par72] David Lorge Parnas. On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, 15(12):1053–1058, 1972.

[PD06] Pavithra Prabhakar and Deepak D'Souza. On the expressiveness of MTL with past operators. In Eugene Asarin and Patricia Bouyer, editors, *Proceedings of the 4th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS'06)*, volume 4202 of *Lecture Notes in Computer Science*, pages 322–336. Springer-Verlag, 2006.

[Pnu77] Amir Pnueli. The temporal logic of programs. In *Proceedings of the 18th IEEE Symposium on Foundations of Computer Science (FOCS'77)*, pages 46–57, 1977.

[Pre04] Roger S. Pressman. *Software Engineering: A Practitioner's Approach*. McGraw-Hill, 6th edition, 2004.

[PRM95] Matteo Pradella, Matteo Rossi, and Dino Mandrioli. ArchiTRIO: A UML-compatible language for architectural description and its formal semantics. In Farn Wang, editor, *Proceedings of the 15th International Conference on Formal Techniques for Networked and Distributed Systems (FORTE'05)*, volume 3731 of *Lecture Notes in Computer Science*, pages 381–395. Springer-Verlag, 1995.

[PSSM03] Matteo Pradella, Pierluigi San Pietro, Paola Spoletini, and Angelo Morzenti. Practical model checking of LTL with past. In *Proceedings of the 1st International Workshop on Automated Technology for Verification and Analysis (ATVA'03)*, 2003.

[PW97] Doron Peled and Thomas Wilke. Stutter-invariant temporal properties are expressible without the next-time operator. *Information Processing Letters*, 63(5):243–246, 1997.

[Rab03] Alexander Moshe Rabinovich. Automata over continuous time. *Theoretical Computer Science*, 300(1–3):331–363, 2003.

[RS97] Jean-François Raskin and Pierre-Yves Schobbens. State clock logic: A decidable real-time logic. In Oded Maler, editor, *Proceedings of the International Workshop on Hybrid and Real-Time Systems*, volume 1201 of *Lecture Notes in Computer Science*, pages 33–47. Springer-Verlag, 1997.

[RT97] Alexander Moshe Rabinovich and Boris A. Trakhtenbrot. From finite automata toward hybrid systems. In Ludwik Czaja Bogdan S. Chlebus, editor, *Proceedings of the 11th International Symposium on Fundamentals of Computation Theory (FCT'97)*, volume 1279 of *Lecture Notes in Computer Science*, pages 411–422. Springer-Verlag, 1997.

[San92] Pierluigi San Pietro. Completezza di un sistema di assiomi per TRIO. Unpublished manuscript (in Italian), 1992.

[Sch02] Ph. Schnoebelen. Verifying lossy channel systems has nonprimitive recursive complexity. *Information Processing Letters*, 83(5):251–261, 2002.

[SFR+06] Elisabeth A. Strunk, Carlo A. Furia, Matteo Rossi, John C. Knight, and Dino Mandrioli. The engineering roles of requirements and specification. Technical Report CS-2006-21, Department of Computer Science, University of Virginia, October 2006. Also: Technical Report 2006.61, Dipartimento di Elettronica e Informazione, Politecnico di Milano. Submitted for publication.

[Sha98] Natarajan Shankar. Lazy compositional verification. In de Roever et al. [dRLP98], pages 541–564.

[Sip05] Michael Sipser. *Introduction to the Theory of Computation*. Course Technology, 2nd edition, 2005.

[SLMR05] John A. Stankovic, Insup Lee, Aloysius K. Mok, and Raj Rajkumar. Opportunities and obligations for physical computing systems. *IEEE Computer*, 38(11):23–31, 2005.

[SMM00] Pierluigi San Pietro, Angelo Morzenti, and Sandro Morasca. Generation of execution sequences for modular time critical systems. *IEEE Transactions on Software Engineering*, 26(2):128–149, 2000.

[Soa96] Robert I. Soare. Computability and recursion. *Bulletin of Symbolic Logic*, 2(3):284–321, 1996.

[Som06] Ian Sommerville. *Software Engineering*. Addison Wesley, 8th edition, 2006.

[Spo05] Paola Spoletini. *Verification of Temporal Logic Specification via Model Checking*. PhD thesis, Dipartimento di Elettronica e Informazione, Politecnico di Milano, May 2005.

[Sta88] Eugene W. Stark. Proving entailment between conceptual state specifications. *Theoretical Computer Science*, 56(1):135–154, 1988.

[Tra95] Boris A. Trakhtenbrot. Origins and metamorphoses of the trinity: Logics, nets, automata. In *Proceedings, 10th Annual IEEE Symposium on Logic in Computer Science*, pages 506–507. IEEE Computer Society Press, 1995.

[TRI] The TRIO homepage. `http://www.elet.polimi.it/res/TRIO/`.

[TRI06] TRIO tools documentation (from "Formal Methods in Concurrent and Distributed Systems" course website). `http://www.elet.polimi.it/upload/furia/teaching/fms/trio.html`, 2006.

[Tsa00] Yih-Kuen Tsay. Compositional verification in linear-time temporal logic. In Jerzy Tiuryn, editor, *Proceedings of the 3rd International Conference on Foundations of Software Science and Computation Structures (FOSSACS'00)*, volume 1784 of *Lecture Notes in Computer Science*, pages 344–358. Springer-Verlag, 2000.

[Tur36] Alan M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(42):230–265, 1936. Corrections in [Tur37].

[Tur37] Alan M. Turing. On computable numbers, with an application to the Entscheidungsproblem: A correction. *Proceedings of the London Mathematical Society*, 2(43):544–546, 1937.

[Tur49] Alan Turing. On checking a large routine. In *Report of a conference on high-speed automatic calculating machines*. University Mathematical Laboratory, Cambridge, 1949.

[TW04] Denis Thérien and Thomas Wilke. Nesting until and since in linear temporal logic. *Theory of Computing Systems*, 37(1):111–131, 2004.

[UML04] Unified modeling language specification. Technical Report formal/04-07-02, Object Management Group, 2004.

[UML05] UML 2.0 superstructure specification. Technical Report formal/05-07-04, Object Management Group, 2005.

[Var] Ada Varisco. *Geometria proiettiva*. Città Studi. (In Italian).

[Var97] Moshe Y. Vardi. Alternating automata: Unifying truth and validity checking for temporal logics. In William McCune, editor, *Proceedings of the 14th International Conference on Automated Deduction (CADE'97)*, volume 1249 of *Lecture Notes in Computer Science*, pages 191–206. Springer-Verlag, 1997.

[Wil94] Thomas Wilke. Specifying timed state sequences in powerful decidable logics and timed automata. In Hans Langmaack, Willem P. de Roever, and Jan Vytopil, editors, *Proceedings of the 3rd International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT'94)*, volume 863 of *Lecture Notes in Computer Science*, pages 694–715. Springer-Verlag, 1994.

[Wil99] Thomas Wilke. Classifying discrete temporal properties. In Christoph Meinel and Sophie Tison, editors, *Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science (STACS'99)*, volume 1563 of *Lecture Notes in Computer Science*, pages 32–46. Springer-Verlag, 1999.

[Wol83] Pierre Wolper. Temporal logic can be more expressive. *Information and Control*, 56(1):72–99, 1983.

[WVS83] Pierre Wolper, Moshe Y. Vardi, and A. Prasad Sistla. Reasoning about infinite computation paths. In *Proceedings of the 24th Annual Symposium on Foundations of Computer Science (FOCS'83)*, pages 185–194. IEEE Computer Society Press, 1983.

[XCC94] Qiwen Xu, Antonio Cau, and Pierre Collette. On unifying assumption-commitment style proof rules for concurrency. In Bengt Jonsson and Joachim Parrow, editors, *Proceedings of the 5th International Conference on Concurrency Theory (CONCUR'94)*, volume 836 of *Lecture Notes in Computer Science*, pages 267–282. Springer-Verlag, 1994.

Politecnico di Milano
Dipartimento di Elettronica e Informazione
Piazza Leonardo da Vinci 32 I 20133 — Milano