Integrating Discrete- and Continuous-Time Metric Temporal Logics Through Sampling

Carlo A. Furia and Matteo Rossi

Dipartimento di Elettronica e Informazione, Politecnico di Milano
32, Piazza Leonardo da Vinci, 20133, Milano, Italy
{furia, rossi}@elet.polimi.it

Abstract. Real-time systems usually encompass parts that are best described by a continuous-time model, such as physical processes under control, together with other components that are more naturally formalized by a discrete-time model, such as digital computing modules. Describing such systems in a unified framework based on metric temporal logic requires to integrate formulas which are interpreted over discrete and continuous time.
In this paper, we tackle this problem with reference to the metric temporal logic TRIO, that admits both a discrete-time and a continuous-time semantics. We identify sufficient conditions under which TRIO formulas have a consistent truth value when moving from continuous-time to discrete-time interpretations, or vice versa. These conditions basically involve the restriction to a proper subset of the TRIO language and a requirement on the finite variability over time of the basic items in the specification formulas. We demonstrate the approach with an example of specification and verification.

Keywords: formal methods, real-time, integration, discretization, metric temporal logic, discrete time, continuous time, dense time

1 Introduction and Motivation

The application of formal methods to the description and analysis of large and heterogenous 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, 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 a good thing, 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.

A particularly relevant instance of the above general problem is encountered when describing real-time systems, which require a quantitative modeling of time. Commonly, such 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 object of the present paper.

In particular, let us consider the framework of descriptive specifications based on (metric) temporal logic. Some temporal logic languages have semantics for both a continuous-time model and a discrete-time one: each formula of the language can be interpreted in one of the two classes of models. TRIO is an example of these logics [5], the one we are considering in this paper; MTL [14] is another well-known instance.[1] The discrete-time semantics and the continuous-time one, however, 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 introduce the notion of *sampling invariance* of a specification formula. 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 (modulo some additional technical requirements). 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 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.

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-time model, which often benefits from more automated

[1] We note that all the results drawn in this paper about TRIO can be applied to MTL with little effort.

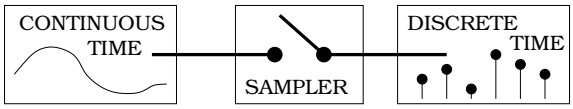

Fig. 1. A system with a sampler.

approaches, while still being able to describe naturally physical processes in a continuous-time model.

Another interesting approach that could 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.

Section 2 introduces $\frac{\mathbb{R}}{\mathbb{Z}}$TRIO, a subset of TRIO for which sampling invariance can be achieved. Then, Section 3 defines the sampling invariance requirement and demonstrates that $\frac{\mathbb{R}}{\mathbb{Z}}$TRIO formulas are invariant under sampling. Section 4 demonstrates the use of the notion of sampling invariance for integration by developing a simple example of specification and verification. Finally, Section 5 compares our approach with related works, and Section 6 draws conclusions and outlines future (and current) work. For the lack of space, we have omitted some technical details and proofs; they can be found in [9].

2 The $\frac{\mathbb{R}}{\mathbb{Z}}$TRIO Metric Temporal Logic

Let us start by presenting our reference metric temporal logic, namely TRIO [10, 15, 5]. More precisely, this section introduces a fragment of full TRIO that we shall call $\frac{\mathbb{R}}{\mathbb{Z}}$TRIO[2]; it is a syntactic and expressive subset of the former, robust with respect to our notion of sampling invariance (as it will be defined in the following Section 3). The presentation of $\frac{\mathbb{R}}{\mathbb{Z}}$TRIO will be aided by an example consisting in the formal description of a simple controlled reservoir system, which will be further analyzed in Section 4.

2.1 Syntax

Whereas TRIO is based on a single modal operator named Dist [5], $\frac{\mathbb{R}}{\mathbb{Z}}$TRIO adopts the bounded Until and Since as primitive operators, because this permits

[2] You can read it as "ar-zee-TRIO".

a simpler statement of the sampling invariance results. More precisely, let Ξ be a set of time-dependent *conditions*. These are basically Boolean expressions obtained by functional combination of basic time-dependent items with constants. We are going to define them precisely later on (in Section 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 symbols representing constants. 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.

$$\phi ::= \xi \mid \text{Until}_{I\rangle}(\phi_1, \phi_2) \mid \text{Since}_{I\langle}(\phi_1, \phi_2) \mid \neg\phi \mid \phi_1 \wedge \phi_2$$

From these basic operators, it is customary to define a number of *derived* operators: Table 1 lists those used in this paper.[3]

Operator	$\equiv$	Definition
$\text{Releases}_{I\rangle}(\phi_1, \phi_2)$	$\equiv$	$\neg\text{Until}_{I\rangle}(\neg\phi_1, \neg\phi_2)$
$\text{Released}_{I\langle}(\phi_1, \phi_2)$	$\equiv$	$\neg\text{Since}_{I\langle}(\neg\phi_1, \neg\phi_2)$
$\exists t \in I = \langle l, u\rangle : \text{Dist}(\phi, t)$	$\equiv$	$\begin{cases} \text{Until}_{\langle l,u\rangle\rangle}(\text{true}, \phi) & \text{if } u \geq l \geq 0 \\ \text{Since}_{\langle -l,-u\rangle\rangle}(\text{true}, \phi) & \text{if } u \leq l \leq 0 \end{cases}$
$\forall t \in I = \langle l, u\rangle : \text{Dist}(\phi, t)$	$\equiv$	$\begin{cases} \text{Releases}_{\langle l,u\rangle\rangle}(\text{false}, \phi) & \text{if } u \geq l \geq 0 \\ \text{Released}_{\langle -l,-u\rangle\rangle}(\text{false}, \phi) & \text{if } u \leq l \leq 0 \end{cases}$
$\text{Dist}(\phi, d)$	$\equiv$	$\forall t \in [d, d] : \text{Dist}(\phi, t)$
$\text{Futr}(\phi, d)$	$\equiv$	$d \geq 0 \wedge \text{Dist}(\phi, d)$
$\text{Som}(\phi)$	$\equiv$	$\exists t \in (-\infty, 0] : \text{Dist}(\phi, t) \vee \exists t \in [0, +\infty) : \text{Dist}(\phi, t)$
$\text{Alw}(\phi)$	$\equiv$	$\forall t \in (-\infty, 0] : \text{Dist}(\phi, t) \wedge \forall t \in [0, +\infty) : \text{Dist}(\phi, t)$
$\text{AlwP}(\phi)$	$\equiv$	$\forall t \in (-\infty, 0) : \text{Dist}(\phi, t)$
$\text{AlwP}_{\text{i}}(\phi)$	$\equiv$	$\forall t \in (-\infty, 0] : \text{Dist}(\phi, t)$
$\text{WithinP}(\phi, \tau)$	$\equiv$	$\exists t \in (-\tau, 0) : \text{Dist}(\phi, t)$
$\text{WithinP}_{\text{ii}}(\phi, \tau)$	$\equiv$	$\exists t \in [-\tau, 0] : \text{Dist}(\phi, t)$
$\text{Lasts}(\phi, \tau)$	$\equiv$	$\forall t \in (0, \tau) : \text{Dist}(\phi, t)$
$\text{Lasts}_{\text{ii}}(\phi, \tau)$	$\equiv$	$\forall t \in [0, \tau] : \text{Dist}(\phi, t)$

Table 1. Some $\frac{\mathbb{R}}{\mathbb{Z}}$TRIO derived temporal operators

2.2 Semantics

In defining $\frac{\mathbb{R}}{\mathbb{Z}}$TRIO semantics we assume that constants 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

[3] Disjunction $\vee$ is defined as usual. 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.

$\mathbb{T}$ 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 an $\frac{\mathbb{R}}{\mathbb{Z}}$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 $\frac{\mathbb{R}}{\mathbb{Z}}$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 $\mathbb{T}$; in practice this will be either the reals $\mathbb{R}$ or the integers $\mathbb{Z}$.

$b(t) \models_\mathbb{T} \xi$	iff	$\xi\|_{b(t)}$
$b(t) \models_\mathbb{T} \text{Until}_{I\rangle}(\phi_1, \phi_2)$	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, an $\frac{\mathbb{R}}{\mathbb{Z}}$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., $[\![\phi]\!]_\mathbb{T} \equiv \{b \in \mathcal{B}_\mathbb{T} \mid b \models_\mathbb{T} \phi\}$. In the remainder we will sometime use the expression "behaviors of a formula ϕ" to mean the set $[\![\phi]\!]_\mathbb{T}$.

2.3 The Controlled Reservoir

Let us briefly illustrate the practical use of the $\frac{\mathbb{R}}{\mathbb{Z}}$TRIO language by building a descriptive specification (i.e., consisting of a set of logic axioms) for a controlled reservoir.

The reservoir is filled with some liquid; at any time, the (measured) level of the liquid is represented by a time-dependent item l that takes value in the set $\mathbb{R}_{\geq 0}$. Furthermore, the reservoir can leak and/or be filled with new liquid. This is modeled through two Boolean-valued time-dependent items L and F indicating the reservoir leaking and being filled, respectively. Therefore, the behaviors (i.e., the logical models) of our reservoir will be functions mapping the time domain $\mathbb{T}$ to the domain $D_\text{res} = \mathbb{R}_{\geq 0} \times \{0, 1\} \times \{0, 1\}$.

Let us start the formal specification by writing some TRIO (not $\frac{\mathbb{R}}{\mathbb{Z}}$TRIO) axioms that define how the level varies over time according to whether the reservoir is leaking and/or being filled. Thus, let us introduce two positive real constants $\mathrm{r_f}, \mathrm{r_l}$ that denote the rate at which the level increases, and at which it decreases, when the reservoir is filling and leaking, respectively. Then, we state that: whenever the reservoir is being filled (i.e., F) and is not leaking (i.e., $\neg\mathsf{L}$) for a (generic) time interval of length t (i.e., $\text{Lasts}(\mathsf{F} \wedge \neg\mathsf{L}, t)$, and the level is some l at the beginning of the interval (i.e., $\mathsf{l} = l$), then at the end of the interval the level will

have grown to $l + \mathrm{r_f} t$ (i.e., $\mathrm{Futr}(\mathsf{l} = l + \mathrm{r_f} t, t)$). Let us define similarly three other axioms to describe all the possible combinations of filling and leaking.

$$\mathrm{Lasts}(\mathsf{F} \wedge \mathsf{L}, t) \wedge \mathsf{l} = l \;\Rightarrow\; \mathrm{Futr}(\mathsf{l} = l + (\mathrm{r_f} - \mathrm{r_l})t, t) \tag{1}$$

$$\mathrm{Lasts}(\mathsf{F} \wedge \neg\mathsf{L}, t) \wedge \mathsf{l} = l \;\Rightarrow\; \mathrm{Futr}(\mathsf{l} = l + \mathrm{r_f} t, t) \tag{2}$$

$$\mathrm{Lasts}(\neg\mathsf{F} \wedge \mathsf{L}, t) \wedge \mathsf{l} = l \;\Rightarrow\; \mathrm{Futr}(\mathsf{l} = \max(l - \mathrm{r_l} t, 0), t) \tag{3}$$

$$\mathrm{Lasted}(\neg\mathsf{F} \wedge \neg\mathsf{L}, t) \wedge \mathsf{l} = l \;\Rightarrow\; \mathrm{Futr}(\mathsf{l} = l, t) \tag{4}$$

In order to define the control action, let us introduce two more constants. The overall goal of the control action is to keep the level above a minimum level l, whatever the leaking behavior is, by filling it whenever needed. To this end, we define a threshold level denoted by $\mathrm{t_l}$: whenever the level goes below $\mathrm{t_l}$, the controller activates the filling, as indicated by Axiom 5 below. Obviously, we assume $\mathrm{t_l} > \mathrm{l}$. Finally, let us also state that the level is initially above the threshold $\mathrm{t_l}$ (Formula 6).

$$\mathsf{l} < \mathrm{t_l} \;\Rightarrow\; \mathsf{F} \tag{5}$$

$$\mathrm{Som}(\mathrm{AlwP}(\mathsf{l} \geq \mathrm{t_l})) \tag{6}$$

Let us close this section with two remarks. First, we assume that formulas (1–4) and (6) are interpreted over $\mathbb{R}$ as time domain, since they model the behavior of a physical system; conversely, formula (5) is implicitly interpreted over $\mathbb{Z}$ as time domain, since it describes the behavior of a digital device that "watches" the physical system through a sampler. Second, while Formulas 5 and 6 qualify as $^{\mathbb{R}}_{\mathbb{Z}}$TRIO formulas, the other Formulas (1–4) involve free (time) variables (namely, t), so they are full-TRIO formulas, *not* $^{\mathbb{R}}_{\mathbb{Z}}$TRIO.

3 Sampling Invariance and Integration

The overall goal of integrating formulas that are interpreted over different time domains basically requires to define a suitable notion of *invariance*. Whenever a formula achieves this invariance it means that its "intended meaning" is preserved by changes of the temporal domain. Thus, the formula can be equivalently interpreted under any of the time domains, putting it on a common ground with the other formulas, effectively integrating it with them.

Our notion of invariance is named *sampling invariance*, and relates continuous- and discrete-time behaviors of a formula through what we call the *sampling* of a behavior. Precisely, 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 $\delta \in \mathbb{R}_{>0}$ from a basic offset $z \in \mathbb{R}$. We call δ the *sampling period* and z the *origin* of the sampling. In formulas, we have the following definition:

$$\forall k \in \mathbb{Z}: \quad \sigma_{\delta,z}[b](k) \;\equiv\; b(z + k\delta)$$

3.1 Sampling Invariance

Let us now give a sensible definition of *sampling invariance*. Ideally, we would like to be able to say that, for a given formula ϕ, sampling period δ, and origin z: the sampling of any of its continuous-time behaviors is one of its discrete-time behaviors (i.e., $b \in [\![\phi]\!]_{\mathbb{R}} \Rightarrow \sigma_{\delta,z}[b] \in [\![\phi]\!]_{\mathbb{Z}}$); and, conversely, any continuous-time behavior whose sampling is a discrete-time behavior of ϕ is also a valid continuous-time behavior of ϕ (i.e., $b \in [\![\phi]\!]_{\mathbb{Z}} \Rightarrow \forall b' : (\sigma_{\delta,z}[b'] = b \Rightarrow b' \in [\![\phi]\!]_{\mathbb{R}})$).

This ideal notion of sampling invariance is not achievable, as it is, by any non-trivial temporal logic language (and $^{\mathbb{R}}_{\mathbb{Z}}$TRIO in particular), as the continuous-time and the discrete-time behaviors of a formula are in general only loosely related. In particular, let us point out two basic reasons why the above definitions must be relaxed to be pursuable.

Regularity of behaviors. The first and most relevant issue concerns the fact that the density of the time domain $\mathbb{R}$ allows for behaviors that may change an unbounded number of times between any two sampling instants; therefore, the information about these "intermediate changes" is completely lost by sampling the behavior. Consider, for instance, Figure 2(a). There, a Boolean-valued item

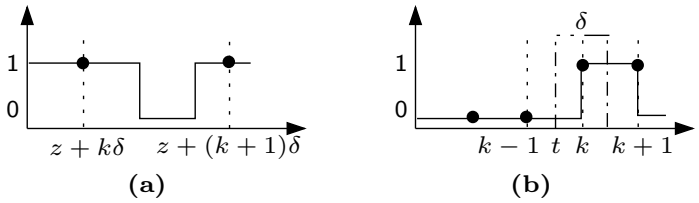

Fig. 2. (a) Change detection failure; (b) Moving interval.

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 "detects" two consecutive true values. Instead, we would like that the "rate of change" of the items is slow-paced enough that every change 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 conformant 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 3.2. 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*.

Discrete steps and continuous units. The second obstacle to achieving sampling invariance is instead a technical issue concerning measurement units. Let us illustrate it with the $\frac{\mathbb{R}}{\mathbb{Z}}$TRIO formula $\text{Lasts}(\mathsf{F}, 1/2)$. 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 $\text{Lasts}(\mathsf{F}, 1/2\delta)$ 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 $\text{Lasts}_{\text{ii}}(\mathsf{F}, 1)$ in discrete time: F holds for two consecutive time steps (including the current one). In continuous time, when evaluating the corresponding $\text{Lasts}_{\text{ii}}(\mathsf{F}, \delta)$ between two consecutive sampling instants (cf. instant t in Figure 2(b)), we can only state a *weaker* property about F holding, namely that F 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.

In Section 3.2 we will 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 $\eta_\delta^{\mathbb{Z}}\{\cdot\}$.

Definition of Sampling Invariance. We are finally able to give a formal definition of sampling invariance which, by employing the above outlined ingredients, is achievable for $\frac{\mathbb{R}}{\mathbb{Z}}$TRIO formulas (as it will be shown in Section 3.3).

Definition 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 $b \in \mathcal{B}_{\mathbb{Z}}$:*

$$b \in [\![\phi]\!]_{\mathbb{Z}} \quad \Rightarrow \quad \forall b' \in [\![\chi]\!]_{\mathbb{R}} : \big(\sigma_{\delta,z}\,[b'] = b \;\Rightarrow\; b' \in [\![\eta_\delta^{\mathbb{Z}}\{\phi\} \wedge \chi]\!]_{\mathbb{R}}\big)$$

- *ϕ is* sampling invariant *iff it is closed under sampling (when interpreted in the continuous-time domain) and closed under inverse sampling (when interpreted in the discrete-time domain).*

3.2 Adaptation, Conditions, and Behavior Constraints

The proof that $^{\mathbb{R}}_{\mathbb{Z}}$TRIO is sampling invariant is given for formulas of the language that are in a *normal form*, where there is no nesting of temporal operators, and all negations are pushed inside. Consequently, negations on Since and Until are replaceable by the Releases and Released operators. As it is fully shown in [9], any $^{\mathbb{R}}_{\mathbb{Z}}$TRIO can be put in normal form, possibly introducing auxiliary time-dependent items. Therefore, in the remainder of this section we will assume to deal with formulas in normal form.

Adaptation Function. Let us define the adaptation function $\eta^{\mathbb{R}}_{\delta}\{\cdot\}$ from continuous to discrete time as follows, where l' is $\lfloor l/\delta \rfloor$ if $\langle$ is (, and $\lceil l/\delta \rceil$ if $\langle$ is [, and u' is $\lceil u/\delta \rceil$ if $\rangle$ is), and $\lfloor u/\delta \rfloor$ if $\rangle$ is].[4]

$$
\begin{array}{lcl}
\eta^{\mathbb{R}}_{\delta}\{\xi\} & \equiv & \xi \\
\eta^{\mathbb{R}}_{\delta}\{\mathrm{Until}_{\langle l,u\rangle\rangle}(\xi_1,\xi_2)\} & \equiv & \mathrm{Until}_{[\lfloor l/\delta\rfloor,\lceil u/\delta\rceil])}(\xi_1,\xi_2) \\
\eta^{\mathbb{R}}_{\delta}\{\mathrm{Since}_{\langle l,u\rangle\langle}(\xi_1,\xi_2)\} & \equiv & \mathrm{Since}_{[\lfloor l/\delta\rfloor,\lceil u/\delta\rceil](}(\xi_1,\xi_2) \\
\eta^{\mathbb{R}}_{\delta}\{\mathrm{Releases}_{\langle l,u\rangle\rangle}(\xi_1,\xi_2)\} & \equiv & \mathrm{Releases}_{\langle l',u'\rangle]}(\xi_1,\xi_2) \\
\eta^{\mathbb{R}}_{\delta}\{\mathrm{Released}_{\langle l,u\rangle\langle}(\xi_1,\xi_2)\} & \equiv & \mathrm{Released}_{\langle l',u'\rangle[}(\xi_1,\xi_2) \\
\eta^{\mathbb{R}}_{\delta}\{\phi_1 \wedge \phi_2\} & \equiv & \eta^{\mathbb{R}}_{\delta}\{\phi_1\} \wedge \eta^{\mathbb{R}}_{\delta}\{\phi_2\} \\
\eta^{\mathbb{R}}_{\delta}\{\phi_1 \vee \phi_2\} & \equiv & \eta^{\mathbb{R}}_{\delta}\{\phi_1\} \vee \eta^{\mathbb{R}}_{\delta}\{\phi_2\}
\end{array}
$$

The adaptation function $\eta^{\mathbb{Z}}_{\delta}\{\cdot\}$ from discrete to continuous time is instead defined as follows, where the adapted interval $\langle (l-1)\delta, (u-1)\delta\rangle$ for the Until and Since operators can indifferently be taken to include or exclude their endpoints.[5]

$$
\begin{array}{lcl}
\eta^{\mathbb{Z}}_{\delta}\{\xi\} & \equiv & \xi \\
\eta^{\mathbb{Z}}_{\delta}\{\mathrm{Until}_{[l,u]}(\xi_1,\xi_2)\} & \equiv & \mathrm{Until}_{\langle (l-1)\delta,(u+1)\delta\rangle]}(\xi_1,\xi_2) \\
\eta^{\mathbb{Z}}_{\delta}\{\mathrm{Since}_{[l,u][}(\xi_1,\xi_2)\} & \equiv & \mathrm{Since}_{\langle (l-1)\delta,(u+1)\delta\rangle[}(\xi_1,\xi_2) \\
\eta^{\mathbb{Z}}_{\delta}\{\mathrm{Releases}_{[l,u])}(\xi_1,\xi_2)\} & \equiv & \mathrm{Releases}_{[(l+1)\delta,(u-1)\delta])}(\xi_1,\xi_2) \\
\eta^{\mathbb{Z}}_{\delta}\{\mathrm{Released}_{[l,u](}(\xi_1,\xi_2)\} & \equiv & \mathrm{Released}_{[(l+1)\delta,(u-1)\delta](}(\xi_1,\xi_2) \\
\eta^{\mathbb{Z}}_{\delta}\{\phi_1 \wedge \phi_2\} & \equiv & \eta^{\mathbb{Z}}_{\delta}\{\phi_1\} \wedge \eta^{\mathbb{Z}}_{\delta}\{\phi_2\} \\
\eta^{\mathbb{Z}}_{\delta}\{\phi_1 \vee \phi_2\} & \equiv & \eta^{\mathbb{Z}}_{\delta}\{\phi_1\} \vee \eta^{\mathbb{Z}}_{\delta}\{\phi_2\}
\end{array}
$$

Conditions. The definitions of the primitive conditions ξ appearing in $^{\mathbb{R}}_{\mathbb{Z}}$TRIO formulas depend on whether we deal with primitive time-dependent items take values to discrete or dense domains. This should not be confused with the density of the time domain: for behaviors mapping $\mathbb{T}$ to a generic domain D, we now distinguish whether D is a discrete or a dense set. Note that when discrete- and dense-valued items coexist in the same specification, we just have to consider them separately.

[4] As it is customary, the brackets $\lfloor\cdot\rfloor$ and $\lceil\cdot\rceil$ denote the floor and ceiling functions [11].

[5] Note that in discrete time we need only consider closed intervals.

Discrete-valued items. If all primitive time-dependent items in $\Psi = \{\psi_1, \ldots \psi_n\}$ have discrete codomains, then $D \equiv D_1 \times \cdots \times D_n$ where each D_i is a discrete set. Let Λ be a set of constants from the sets D_i's, and Γ be a set of functions having domains in subsets of D and codomains 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:

$$\xi \ ::= \ f(\psi_1, \ldots, \psi_n) \bowtie \lambda \mid f_1(\psi_1, \ldots, \psi_n) \bowtie f_2(\psi_1, \ldots, \psi_n) \mid \neg\xi \mid \xi_1 \wedge \xi_2$$

Dense-valued items. If all time-dependent items in Ψ have dense codomains, then $D \equiv D_1 \times \cdots \times D_n$ where each D_i is a dense set. 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 $i = 1, \ldots, n$. Then, if $\lambda \in \Lambda$, conditions ξ are defined simply as follows:

$$\xi \ ::= \ \langle \psi_1, \ldots, \psi_n \rangle \in \lambda \mid \neg\xi \mid \xi_1 \wedge \xi_2$$

Then, for a behavior $b : \mathbb{T} \to D$, the evaluation of conditions on b at $t \in \mathbb{T}$ is defined as obvious; in particular — for discrete-valued items — a function application $f(\psi_1, \ldots, \psi_n)$ is interpreted by taking $f(\psi_1|_{b(t)}, \ldots, \psi_n|_{b(t)})$, and — for dense-valued items — $\langle \psi_1, \ldots, \psi_n \rangle|_{b(t)} \in \lambda$ is interpreted as $\forall i = 1, \ldots, n : \psi_i(t) \in \langle^i l_i, u_i\rangle^i$ (i.e., all items are in their respective ranges at time t).

Behavior Constraints. According to whether we are considering discrete- or dense-valued items in our specification, we have two different definitions for the behavior constraint χ. Again, if discrete and dense items coexist in the same specification, we just consider both conditions for their respective items.

Discrete-valued items. The constraint on behaviors for discrete-valued items is given in Formula $\chi_\circ$ and basically requires that the value of each item in Ψ is held for δ time units, at least.[6]

$$\chi_\circ \triangleq \forall v \in D : \ (\langle \psi_1, \ldots, \psi_n \rangle = v \ \Rightarrow \ \mathrm{WithinP_{ii}}(\mathrm{Lasts_{ii}}(\langle \psi_1, \ldots, \psi_n \rangle = v, \delta), \delta))$$

Dense-valued items. The constraint on behaviors for dense-valued items is expressed by Formula $\chi_\bullet^\phi$ depends on the actual conditions ξ that we introduced in the formula ϕ. Let $\widetilde{\Xi}$ be the set of conditions that appear in ϕ; $\chi_\bullet^\phi$ requires that the value of every item in Ψ is such that its changes with respect to the conditions in $\widetilde{\Xi}$ occur at most once every δ time units. 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 just a shorthand for the explicit enumeration of all the conditions in (the finite set) $\widetilde{\Xi}$.

$$\chi_\bullet^\phi \triangleq \forall\widetilde{\xi} \in \widetilde{\Xi} : \ \mathrm{WithinP_{ii}}\Big(\mathrm{Lasts_{ii}}\Big(\widetilde{\xi}, \delta\Big), \delta\Big) \vee \mathrm{WithinP_{ii}}\Big(\mathrm{Lasts_{ii}}\Big(\neg\widetilde{\xi}, \delta\Big), \delta\Big)$$

[6] Actually, while $^{\mathbb{R}}_{\mathbb{Z}}$TRIO, as we defined it above, is purely propositional, $\chi_\circ$ uses a universal quantification on non-temporal variables. However, it is not difficult to understand that this generalization does not impact on sampling-invariance.

3.3 Sampling Invariance of $^{\mathbb{R}}_{\mathbb{Z}}$TRIO

We finally state the sampling invariance of $^{\mathbb{R}}_{\mathbb{Z}}$TRIO formulas.

Theorem 1 (Sampling Invariance). *$^{\mathbb{R}}_{\mathbb{Z}}$TRIO is sampling invariant with respect to the behavior constraints $\chi_\circ$ and $\chi^{\phi}_{\bullet}$, the adaptation functions $\eta^{\mathbb{R}}_{\delta}\{\cdot\}$ and $\eta^{\mathbb{Z}}_{\delta}\{\cdot\}$, for any sampling period δ and origin z.*

Proof (sketch). For the sake of space we just sketch the outline of the proof and refer to [9] for the details.

First of all, let us notice that the proof for both dense- and discrete-valued items is the same, and it relies on the basic fact that the truth value of any condition ξ cannot change more than once between any two consecutive sampling instants, because of the behavior constraints $\chi_\circ$ and $\chi^{\phi}_{\bullet}$. Then, let ϕ be any $^{\mathbb{R}}_{\mathbb{Z}}$TRIO formula.

To prove *closure under sampling*, let b be a continuous-time behavior in $[\![\chi_\circ]\!]_{\mathbb{R}}$, $\phi' = \eta^{\mathbb{R}}_{\delta}\{\phi\}$, 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 $\chi^{\phi}_{\bullet}$) 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 $\eta^{\mathbb{R}}_{\delta}\{\cdot\}$, 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, $\phi' = \eta^{\mathbb{Z}}_{\delta}\{\phi\}$, 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 $\chi^{\phi}_{\bullet}$). Afterwards, 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. □

4 Example

Let us now develop controlled reservoir whose specification was introduced in Section 2.3. Let us consider the following invariance formula, asserting that the

level of liquid never goes below l.

$$\mathrm{Alw}(\mathsf{l} \geq \mathrm{l}) \tag{7}$$

We would like to prove that 7 follows from the axioms, that is (1–6)⇒(7). Notice, however, that the axioms do not refer to a uniform time domain, and thus they need to be integrated first.

A first possible strategy would first notice that Formula 5 does not use temporal operators at all and so it could be easily interpreted over the reals. Then, one would proceed to prove (1–6)⇒(7) assuming continuous time. Such a proof is possible although not very simple — at least with respect to the simplicity of the involved formulas — as it requires to deal with regularity properties of functions with real domain (such as continuity and non-Zenoness); the reader can check out [8], where a similar proof was carried out in continuous time with the aid of a compositional inference rule.

An alternative approach to achieve the verification goal is therefore to exploit the results about integration. First of all we need to identify some simpler formulas, expressible in $\frac{\mathbb{R}}{\mathbb{Z}}$TRIO, which can be easily derived from the TRIO axioms (1–4) assuming $\mathbb{R}$ as time domain. Thus, let us pick a sufficiently small sampling period δ, i.e., one such that $\mathrm{r_l}\delta < \mathrm{t_l} - \mathrm{l}$. Then, we derive three formulas from the axioms (1–4) that state basic properties about the invariance of the reservoir level *over a time interval of length* δ. We omit their simple proofs.

$$\mathrm{Lasts}(\mathsf{F}, \delta) \wedge \mathsf{l} \geq \mathrm{l} \;\Rightarrow\; \mathrm{Futr}(\mathsf{l} \geq \mathrm{l}, \delta) \tag{8}$$

$$\mathsf{l} \geq \mathrm{t_l} \;\Rightarrow\; \mathrm{Futr}(\mathsf{l} \geq \mathsf{l}, \delta) \tag{9}$$

$$\mathrm{Lasts}(\neg\mathsf{F} \wedge \neg\mathsf{L}, \delta) \wedge \mathsf{l} \geq \mathrm{l} \;\Rightarrow\; \mathrm{Futr}(\mathsf{l} \geq \mathrm{l}, \delta) \tag{10}$$

Now, Formulas (8–10,5,6,7) are all well-formed $\frac{\mathbb{R}}{\mathbb{Z}}$TRIO formulas, and are therefore invariant under sampling and inverse sampling. Therefore, let us interpret all of them in discrete time. While Formulas (5) and (7) do not require any adaptation, the other formulas (8–10,6) must be adapted. After translating them in normal form, applying the adaptation function $\eta_\delta^{\mathbb{R}}\{\cdot\}$, and writing them back with derived operators, one gets the following formulas.

$$\mathrm{Lasts_{ii}}(\mathsf{F}, 1) \wedge \mathsf{l} \geq \mathrm{l} \;\Rightarrow\; \mathrm{Futr}(\mathsf{l} \geq \mathrm{l}, 1) \tag{11}$$

$$\mathsf{l} \geq \mathrm{t_l} \;\Rightarrow\; \mathrm{Futr}(\mathsf{l} \geq \mathrm{l}, 1) \tag{12}$$

$$\mathrm{Lasts_{ii}}(\neg\mathsf{F} \wedge \neg\mathsf{L}, 1) \wedge \mathsf{l} \geq \mathrm{l} \;\Rightarrow\; \mathrm{Futr}(\mathsf{l} \geq \mathrm{l}, 1) \tag{13}$$

$$\mathrm{Som}(\mathrm{AlwP_i}(\mathsf{l} \geq \mathrm{t_l})) \tag{14}$$

All in all, we have reduced the verification goal to proving that (11–13,5,14)⇒(7), in discrete time. This is a sufficiently simple task to be totally automated; for instance, we proved it using model-checking techniques for TRIO and the SPIN model checker [16]. Finally, the invariance property (7) that has been proved in discrete time, holds in continuous time as well, for all behaviors satisfying the constraint χ. A further, complementary, analysis could then consider behaviors violating the constraints χ, if they are deemed physically meaningful; we leave this (interesting) problem to future work.

5 Related Works

The problem of formally describing in a uniform manner systems that encompass both discrete-time and continuous-time modules is particularly motivated by the recently growing interest in *hybrid systems* [3]. Although this paper does not deal with hybrid systems in the sense of *hybrid automata* [2], we also allow one to describe systems where both discrete- and continuous-time dynamics occur. However, with respect to the hybrid automata work, our approach considers descriptive models, i.e., entirely based on temporal logic formulas, which are well suited for very high-level descriptions of a system. Moreover, our idea of integration stresses separation of concerns in the development of a specification, as the modules describing different parts of the systems, that obey different models, can be developed independently and then joined together to have a global description of the system. Thus, no *ad hoc* formalism has to be introduced, since the analysis can exploit the ideas and formalisms of temporal logics.

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 [7], 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 complexity, as continuous time often introduces some peculiar difficulties that render reasoning more difficult. On the contrary, our approach aims at achieving a notion of equivalence between discrete-time and continuous-time descriptions, so that one can resort to discrete time when verifying properties, while still being able to describe naturally physical processes using a full continuous-time model.

Another approach to the definition of an equivalence between continuous- and discrete-time behaviors of a specification formula is that based on the notion of *digitization*. Henzinger et al. [12] were the first to introduce and study this notion in the context of temporal logic. 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 major aspects. First, [12] 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 [12]

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 weaker 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. A formal comparison of the two notions of discretization belongs to future work (see also [4]).

We end the presentation of related works by mentioning that several different papers on discretization and continuous/discrete equivalence have exploited similar notions of invariance as that in [12]. For the sake of space we limit ourselves to citing the work by Hung and Giang [13], where a sampling semantics for Duration Calculus (DC) is defined; the paper by Ouaknine [17], where digitization for CSP is discussed; and the paper by Ouaknine and Worrell [18], which discusses the problem of digitization for timed automata, and offers several other references to related works.

6 Conclusions

This paper introduced the notion of *sampling invariance*, which is a physically motivated way to relate the continuous-time and discrete-time behaviors of a system. Sampling invariance means that a temporal logic formula can be interpreted with a continuous-time or discrete-time model consistently and in a uniform manner. Therefore, writing sampling invariant formulas permits to achieve integration of discrete-time and continuous-time formalisms in the same specification, thus being able to verify the system in the discrete-time models, while still describing naturally physical processes with the true continuity permitted by continuous-time models.

We demonstrated how to achieve sampling invariance with $^{\mathbb{R}}_{\mathbb{Z}}$TRIO, a subset of the TRIO language. The approach involved the introduction of some suitable constraints on the possible behaviors of a system, which limit the dynamics in an appropriate way, as well as simple translation rules that adapt the meaning of time units in the two time models. We are confident that the approach can be applied to other metric temporal logics as well (and MTL in particular).

Future work will follow three main directions. Firstly, we will compare the expressiveness of $^{\mathbb{R}}_{\mathbb{Z}}$TRIO with that of other formalisms for the description of real-time and hybrid systems that admit discretization under certain constraints; in particular, we will consider timed and hybrid automata [2], logics such as MITL [1], and the AASAP semantics [6]. Secondly, we will study how to possibly extend and generalize our approach, both in terms of enriching the expressiveness of the logic language, and by attempting variants of the notion of sampling invariance. Thirdly, we will try to apply our results about sampling invariance to a

refinement approach, as we hinted at in the Introduction. This effort should be methodological as well as technical, and it will aim at tackling, from a new perspective, the important problem of refining an abstract high-level specification to an implementable low-level one.

References

1. R. Alur, T. Feder, and T. A. Henzinger. The benefits of relaxing punctuality. *JACM*, 43(1):116–146, 1996.
2. R. Alur, T. A. Henzinger, G. Lafferriere, and G. J. Pappas. Discrete abstractions of hybrid systems. In Antsaklis [3], pages 971–984.
3. P. J. Antsaklis, editor. *Special issue on hybrid systems: theory and applications*, volume 88 of *Proceedings of the IEEE*, 2000.
4. E. Asarin, O. Maler, and A. Pnueli. On discretization of delays in timed automata and digital circuits. In *Proceedings of CONCUR'98*, volume 1466 of *LNCS*, pages 470–484, 1998.
5. E. Ciapessoni, A. Coen-Porisini, E. Crivelli, D. Mandrioli, P. Mirandola, and A. Morzenti. From formal models to formally-based methods: an industrial experience. *ACM TOSEM*, 8(1):79–113, 1999.
6. M. De Wulf, L. Doyen, and J.-F. Raskin. Almost ASAP semantics. *Formal Aspects of Computing*, 17(3):319–341, 2005.
7. C. J. Fidge. Modelling discrete behaviour in a continuous-time formalism. In *Proceedings of IFM'99*, pages 170–188. Springer, 1999.
8. C. A. Furia and M. Rossi. A compositional framework for formally verifying modular systems. volume 116 of *ENTCS*, pages 185–198. Elsevier, 2004.
9. C. A. Furia and M. Rossi. When discrete met continuous. Technical Report 2005.44, DEI, Politecnico di Milano, 2005. Available from `http://www.elet.polimi.it/upload/furia/`.
10. C. Ghezzi, D. Mandrioli, and A. Morzenti. TRIO: A logic language for executable specifications of real-time systems. *JSS*, 12(2):107–123, 1990.
11. R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete Mathematics: A foundation for computer science*. Addison-Wesley, 1994.
12. T. A. Henzinger, Z. Manna, and A. Pnueli. What good are digital clocks? In *Proceedings of ICALP'92*, volume 623 of *LNCS*, pages 545–558, 1992.
13. D. V. Hung and P. H. Giang. Sampling semantics of duration calculus. In *Proceedings of FTRTFT'96*, volume 1135 of *LNCS*, pages 188–207, 1996.
14. R. Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.
15. A. Morzenti, D. Mandrioli, and C. Ghezzi. A model parametric real-time logic. *ACM TOPLAS*, 14(4):521–573, 1992.
16. A. Morzenti, M. Pradella, P. San Pietro, and P. Spoletini. Model-checking TRIO specifications in SPIN. In *Proceedings of FME'03*, volume 2805 of *LNCS*, pages 542–561, 2003.
17. J. Ouaknine. Digisation and full abstraction for dense-time model checking. In *Proceedings of TACAS'02*, volume 2280 of *LNCS*, pages 37–51, 2002.
18. J. Ouaknine and J. Worrell. Revisiting digitization, robustness, and decidability for timed automata. In *Proceedings of LICS'03*, pages 198–207. IEEE Computer Society, 2003.