Automated Compositional Proofs for Real-Time Systems*

Carlo A. Furia, Matteo Rossi, Dino Mandrioli, and Angelo Morzenti

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

Abstract. We present a framework for formally proving that the composition of the behaviors of the different parts of a complex, real-time system ensures a desired global specification of the overall system. The framework is based on a simple compositional rely/guarantee circular inference rule, plus a small set of conditions concerning the integration of the different parts into a whole system. The reference specification language is the TRIO metric linear temporal logic.
The novelty of our approach with respect to existing compositional frameworks — most of which do not deal explicitly with real-time requirements — consists mainly in its generality and abstraction from any assumptions about the underlying computational model and from any semantic characterizations of the temporal logic language used in the specification. Moreover, the framework deals equally well with continuous and discrete time. It is supported by a tool, implemented on top of the proof-checker PVS, to perform deduction-based verification through theorem-proving of modular real-time axiom systems.
As an example of application, we show the verification of a real-time version of the old-fashioned but still relevant "benchmark" of the dining philosophers problem.

Keywords: Formal verification, modular systems, real-time, compositionality, rely/guarantee, axiom systems

1 Introduction

Formal methods are more and more recognized to be a useful tool for the development of applications, as they allow their users to precisely verify the correctness of systems in their early development phases, before uncaught mistakes become overly costly to fix. One drawback often attributed to formal methods, however, is that they do not "scale up", i.e. when the system grows in complexity, they are too cumbersome and unwieldy to be used effectively. A natural solution

* Work supported by the MIUR project: "Quack: Piattaforma per la qualità di sistemi embedded integrati di nuova generazione."

to this problem is to apply well-known software engineering principles such as modularity and separation of concerns to the verification of formal models. A compositional framework can help in this regard, in that it allows one to focus on the single parts of the system at first, and then analyze their mutual interactions at a later moment with a smaller effort than it would be required if all aspects (local and global) of the application were taken into account at once.

This paper presents a compositional inference rule for the TRIO language [5] that is suitable to formally prove the correctness of the behavior of a modular system from the behavior of its components. TRIO is a metric temporal logic for modeling and analysis of time-critical systems, and has been used in a number of industrial projects. Its advanced modular features are useful in writing specifications of complex systems. Our framework combines these features with the compositional inference rule through some application methodology that facilitates its practical use in structured specifications.

The approach followed in this paper belongs to the general framework of axiom systems, where a specification consists of a set of logic formulas and the verification consists in formally demonstrating that certain desired properties follow deductively from the specification formulas. This framework is indeed very general and abstract, as it does not rely on specific semantic assumptions and is independent of any notion of underlying computational model (to be considered when moving from specification towards implementation). In fact, proofs in our framework are to be intended as in classic logic deduction [15], and are supported and semi-automated by the theorem proving tool PVS [18]. Also, even if the reference language is TRIO, the results can be easily extended to any logic formalisms for the description of real-time axiom systems.

The paper is structured as follows: Section 2 shortly introduces the TRIO language; Section 3 presents a proof-oriented compositional framework for TRIO; Section 4 applies the framework to a timed version of Dijkstra's dining philosophers problem [8]; Section 5 reviews the most important compositional rules and frameworks in the literature, and points out where our approach differs from previous works on this subject; Section 6 draws conclusions and outlines future research.

For reasons of space, some of the formulas and proofs discussed in the paper have been omitted. The interested reader can find these details in an extended version of the paper, available online [11].

2 TRIO

TRIO [5] is a general-purpose specification language suitable to describe real-time systems. It is a first-order linear temporal logic that supports a metric on time. In addition to the usual propositional operators and quantifiers, it has a basic modal operator, called *Dist*, that relates the *current time*, which is left implicit in the formula, to another time instant: given a time-dependent formula F (i.e. a term representing a mapping from the time domain to truth values) and a term t indicating a time distance, the formula $Dist(F, t)$ specifies that F holds

at a time instant whose distance is exactly t time units from the current instant. Notice that, in this paper, we deliberately do not formally specify a semantics for the interpretation of TRIO formulas: in fact all the discussion is independent of how the modal operator $Dist$ is interpreted, and of which computational model is chosen, since it involves only syntactic manipulation of formulas.

A number of *derived* temporal operators can be defined from the basic $Dist$ operator through propositional composition and first-order logic quantification. Table 1 shows those used in this paper. Notice that TRIO operators predicat-

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

Table 1. TRIO derived temporal operators

ing on intervals by default do not include the interval boundaries. We define variations that may or may not include such boundaries by using the subscripts i (included) or e (excluded). E.g. $AlwP_i(F) \equiv \forall d \geq 0 : Past(F,d)$, $AlwP_e(F) \equiv AlwP(F)$, $WithinF_{ei} \equiv \exists d \in (0,t] : Futr(F,d)$, $WithinF_{ii} \equiv \exists d \in [0,t] : Futr(F,d)$, etc.

TRIO is well-suited to deal with both dense and discrete time. For specifying 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. The basic encapsulation unit is the class, which is a collection of parameters, basic items, formulas and an interface.

Items are the primitive elements of the specification, such as predicates, time-dependent variables, functions, etc. *States* and *events* are time-dependent items with a particular temporal behavior: events are predicates that are true only at

isolated time instants; states are predicates which are true on non-empty time intervals (see [12] for a more precise definition).

We illustrate TRIO's features by introducing the specification of the *dining philosophers problem* [8]. Our solution is based on a philosopher class and assumes a continuous time model. Continuous time introduces some peculiar difficulties in the specification and verification phases, which we will handle exploiting an axiomatic-deductive approach. The basic items of the class are the event item start for the system initialization, and the items take(s) and release(s), with $s \in \{l, r\}$, indicating, for each philosopher, the action of taking or releasing the left or right fork. Other items are the state eating, which is true 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). 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 $T_e > t_e > 0$.

For each TRIO class, formulas are divided into three categories: axioms, assumptions and theorems. Axioms postulate the basic behavior of the system, assumptions express constraints we must discharge by means of other parts of the system (external to the current class) and theorems describe properties that are derived from other formulas. Any TRIO formula is implicitly universally quantified with the *Alw* operator.

We formalize the basic behavior of a philosopher through axiom formulas. For the sake of space limit, we do not present explicitly all the formulas of the example, but just give an informal presentation of some of them. The interested reader can find the complete example in the extended version of this paper available online [11]. 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.[1] A philosopher becomes "hungry" when he/she has not eaten for a period longer than T_t. If he/she 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 hungry.

Axiom 1 (hungry) $Lasted(\neg\text{eating}, T_t) \wedge UpToNow(\text{available}(l) \wedge \text{available}(r)) \Rightarrow (\text{take}(l) \wedge \text{take}(r)) \vee NowOn(\neg\text{available}(l) \vee \neg\text{available}(r))$

Axioms eating_def and eating_duration state that, when a philosopher succeeds in acquiring both forks, 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. Whenever a philosopher holds both forks we consider him/her eating. When he/she is not

[1] We introduce this simplification with respect to the traditional formulation because our example aims at proving a *real-time, non starvation property*, rather than the absence of deadlocks.

eating we say he/she is thinking (axiom thinking). Finally, a thinking session which has just begun lasts T_t time units (axiom thinking_duration).

Each TRIO class has an interface, defined as the set of items and formulas that are externally *visible*. The user can declare each item to be visible or non-visible; henceforth all the formulas predicating on visible items *only* are considered as visible, while all the other formulas are considered as non-visible outside the class. The interface is synthetically represented with a graphical notation: Figure 1 illustrates the interface of the philosopher class.

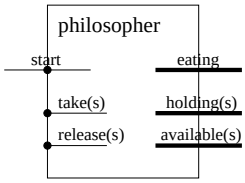

Fig. 1. Interface of the philosopher class

A structured TRIO specification is a collection of *modules*, i.e. instances of TRIO classes. The behavior of the overall composite system is given by the combination of the behaviors of its modules (i.e. it is defined by the logical conjunction of the axioms). 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.

3 A Compositional Framework in TRIO

In this section, we introduce a compositional framework for axiom systems, and the TRIO language in particular. The rationale of our approach is the following. The specification of a complex system is structured into classes. The fundamental behavior of each class is captured by axiom formulas. The derived behavior of each class can be expressed by theorem formulas. In general, according to the rely/guarantee paradigm, we want to relate the derived behavior of a class with certain properties of the (external) environment we assume to hold. When we compose the class with other classes constituting its actual environment, we have to discharge (i.e. prove) the assumption formulas by means of formulas of other classes. If the assumptions on the environment are temporally closed formulas (i.e. they express time-invariant properties), we may use TRIO assumption formulas to represent them; then, when discharging them, it is important to avoid circularities, in order to guarantee the soundness of the composite specification. If, on the other hand, we have to express assumptions on the environment that

are directly temporally related with the derived behavior they guarantee,[2] we need a new operator: the $\twoheadrightarrow$ operator (called *time progression*) for the TRIO language, introduced in Section 3.2. When we compose the class with other classes, rely/guarantee formulas expressed using the $\twoheadrightarrow$ operator are composed using an *ad hoc* inference rule which handles temporal circularities correctly, according to the semantics of the operator. Finally, once the local assumptions have been discharged, we can infer the global specification from the logical composition of the (valid) local specifications.

Notice that the $\twoheadrightarrow$ operator, while similar to the operators presented in [1, 16], differs from them in that it does not impose any constraints nor conditions on the properties involved and the underlying computational model of the formalism. Section 5 discusses this issue in more detail.

3.1 Rely/Guarantee Specifications

Let us consider the rely/guarantee specification of a TRIO class C written according to the following guideline. 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 wish to derive a number of remarkable properties of the class as theorems. These theorems often depend on assumptions about the behavior of the environment. In TRIO, these assumptions can be stated using the language construct of the *assumption* formula. Let us name $\mathcal{AX}_C$, $\mathcal{AS}_C$ and $\mathcal{TH}_C$ the set of all axioms, assumptions and theorems of class C, respectively, and let $\mathcal{F} = \mathcal{AX}_C \cup \mathcal{AS}_C \cup \mathcal{TH}_C$ be the set of all formulas of C. Furthermore, for each set of formulas $\mathcal{F}$, we define $\mathcal{F}^{\vee} \subseteq \mathcal{F}$ to be the set of *visible* formulas in $\mathcal{F}$, i.e. the formulas predicating over visible items only (see Section 2). Therefore, the complete specification of C is represented by the formula $\mathcal{AX}_C \wedge \mathcal{AS}_C \Rightarrow \mathcal{TH}_C$.

Let us map these ideas on the philosophers example. Section 2 showed some axioms of the philosopher class. A derived property of the class we want to state is that there is always a time interval in which both forks are available to the philosopher. In our axiomatization, it suffices to show that this time interval ends at a time distant at most $T_t + 2T_e$ time units. The above property is expressed by the following TRIO formula.

Theorem 2 (fork_availability)
$WithinF_{ei}(UpToNow(\text{available}(l) \wedge \text{available}(r)), T_t + 2T_e)$

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 2. First of all, each fork has to become available within $T_t + T_e$ time units or be already available and remain so for a sufficiently long (i.e. $\geq T_t$) amount of time. Second, we want each fork to be available, for a non-empty time interval, within T_e time units. This is basically

[2] Notice that this is likely to happen when specifying real-time systems.

like assuming that the adjacent philosophers eat no longer than T_e time units. Finally, when a fork becomes available, we assume it to stay so for (at least) T_t time units, i.e. the thinking time of the neighbor philosophers is no shorter than T_t. These three assumptions are formalized by three formulas named availability, availability_2 and lasting_availability (not shown here for brevity).

Let us now formalize what happens when composing TRIO classes. Let us consider n modules $C_1, \ldots, C_n$. For all $i = 1, \ldots, n$, module C_i has a rely/guarantee specification expressed synthetically by the formula $\mathcal{AX}_i \wedge \mathcal{AS}_i \Rightarrow \mathcal{TH}_i$. Let C_{glob} be the class obtained by composing the n instances 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 classes 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} has its own axioms, assumptions and theorems, besides those of its modules, to allow the recursive application of the method. Hence, C_{glob} is described by the formula $\mathcal{AX}_{glob} \wedge \mathcal{AS}_{glob} \Rightarrow \mathcal{TH}_{glob}$.

We seek a way to prove:

$$\mathcal{AX}_{glob} \wedge \mathcal{AS}_{glob} \wedge \bigwedge_{j=1,\ldots,n} (\mathcal{AX}_j \wedge \mathcal{AS}_j \Rightarrow \mathcal{TH}_j) \Rightarrow \mathcal{TH}_{glob}$$

Hence, we want to find a way to *discharge* the local assumptions of each class by means of visible formulas of other classes, so that we can in turn use the validity of the local visible theorems to deduce global results. As it is simple to realize, this kind of reasoning involves a circularity between assumptions and guarantees of the modules, so that a naïve rule does not guarantee soundness, in general. We want to rule out these invalid reasonings in order to obtain a valid rule for composition. To this end we introduce a new temporal operator suitable to express rely/guarantee compositional specifications, and different conditions on the initial validity of the environment assumptions and on how to discharge them. Through these elements, we finally provide a valid compositional rule for rely/guarantee reasoning in TRIO.

3.2 A Rely/Guarantee Inference Rule

Let us introduce the $\twoheadrightarrow$ "time progression" operator for the TRIO language, suitable for expressing temporal relationships between assumptions and guarantees. Let P and Q be two time-dependent formulas. We define the $\twoheadrightarrow$ operator as a shorthand for the formula:

$$P \twoheadrightarrow Q \quad \triangleq \quad \begin{cases} \text{if dense :} AlwP_e(P) \Rightarrow AlwP_i(Q) \wedge NowOn(Q) \\ \text{if discrete :} AlwP_e(P) \Rightarrow AlwP_i(Q) \end{cases}$$

Informally speaking, $P \twoheadrightarrow Q$ means that Q lasts at least as long as P does and even "a bit longer".

Now, we consider rely/guarantee specifications whose semantics is given in terms of the $\twoheadrightarrow$ operator. Therefore, if E is the environment assumption and M

is the guarantee, the rely/guarantee specification is written as $E \twoheadrightarrow M$. Notice that we now admit temporally open formulas to be assumptions and guarantees.

Let us state, without proof, the following property of the $\twoheadrightarrow$ operator; the interested reader can find a proof in [11, Appendix A]. Note that the lemma holds both in dense and in discrete time models.

Lemma 1. *For any formulas P, Q and R, if:*

1. $Som(AlwP_e(P))$
2. $Alw(Q \wedge R \Rightarrow P)$

then:

$$Alw(P \twoheadrightarrow Q) \Rightarrow Alw(R \twoheadrightarrow Q)$$

The following proposition states a sound inference rule[3] for the $\twoheadrightarrow$ operator.

Proposition 1 (Rely/Guarantee Compositional Inference Rule). *If, for $i = 1, \ldots, m$ ($m \in \mathbb{N}^+$):*

1. $Som(AlwP_e(E_i))$ *(that is E_i is initialized)*
2. $E \wedge \bigwedge_{j=1,\ldots,m} M_j \Rightarrow E_i$
3. $\bigwedge_{j=1,\ldots,m} M_j \Rightarrow M$

then: $Alw\left(\bigwedge_{j=1,\ldots,m}(E_j \twoheadrightarrow M_j)\right) \Rightarrow Alw(E \twoheadrightarrow M)$.

Proof. Assume $Alw(\bigwedge_{j=1,\ldots,m}(E_j \twoheadrightarrow M_j))$. It is simple to realize, by considering the definition of the time progression operator, that this implies $Alw\left(\left(\bigwedge_{j=1,\ldots,m} E_j\right) \twoheadrightarrow \left(\bigwedge_{j=1,\ldots,m} M_j\right)\right)$. Moreover, hypothesis 2 implies that $Alw(E \wedge \bigwedge_{j=1,\ldots,m} M_j \Rightarrow \bigwedge_{j=1,\ldots,m} E_j)$, since it holds for every $i = 1, \ldots, m$. Finally, hypothesis 1 implies that $Som(AlwP_e(\bigwedge_{=1,\ldots,m} E_i))$, since there exists a base interval such that the conjunction of the E_i's is true on it[4].

Therefore, we can apply Lemma 1 by substituting $\bigwedge_{j=1,\ldots,m} E_j$ for P, $\bigwedge_{j=1,\ldots,m} M_j$ for Q and E for R. We get:
$Alw\left(\left(\bigwedge_{j=1,\ldots,m} E_j\right) \twoheadrightarrow \left(\bigwedge_{j=1,\ldots,m} M_j\right)\right) \Rightarrow Alw\left(E \twoheadrightarrow \left(\bigwedge_{j=1,\ldots,m} M_j\right)\right)$.

Finally, by combining it with hypothesis 3 and with the definition of the time progression operator, we get the desired result. □

[3] As it is often the case with compositional rules (and formal languages in general), there is a trade off between (relative) completeness and simplicity and ease of use [16, 14]. In this work, we have chosen to privilege the latter over the former, so that the inference rule in Proposition 1 is *incomplete*, as several other compositional rules in the literature [16, 4]. For the sake of space limit, we do not discuss this issue in depth, leaving it for future extensions of this work.

[4] Consider the intersection of the intervals on which each of the E_i's is individually true; this intersection is non-empty, since all intersected intervals are unbounded on the left, because of the $AlwP$ operator.

3.3 Integrating TRIO Modules

Let us consider the composition of n modules $C_1, \ldots, C_n$. We want to show briefly how the inference rule of Proposition 1 can be used in a large TRIO specification, integrating it with generic TRIO formulas. In general, each of the n modules we compose may have one or more rely/guarantee formulas of the form $E \twoheadrightarrow M$ among its theorems. For each module $j = 1, \ldots, n$, let $\mathcal{TH}_j^{rg} \subseteq \mathcal{TH}_j$ be the set of theorems in the form $E \twoheadrightarrow M$ of class j. More formally, for each theorem formula $F \in \mathcal{TH}_j$, $F \in \mathcal{TH}_j^{rg}$ if and only if F can be written as $F \equiv E \twoheadrightarrow M$ for some formulas E and M. Let us also define m to be the number of such formulas over all classes: $m = \sum_{j=1,\ldots,n} |\mathcal{TH}_j^{rg}|$. Moreover, $\mathcal{TH}_j^{nrg}$ is defined as the complement set $\mathcal{TH}_j \setminus \mathcal{TH}_j^{rg}$ for all $j = 1, \ldots, n$. The composite class C_{glob} also has its own rely/guarantee formula $E_{glob} \twoheadrightarrow M_{glob}$ among its theorems $\mathcal{TH}_{glob}$.

Now, we have to 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, [15]). 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 $\phi \overset{+}{\rightsquigarrow}_\pi \chi$ 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 $\pi \in \Pi$ such that $\phi \overset{+}{\rightsquigarrow}_\pi \chi$.

Finally, we can formulate a "checklist" to follow when verifying our composite specification. The rationale is that we avoid circularities in discharging temporally closed assumptions, and we resolve possible circularities between $\twoheadrightarrow$ specifications by using the inference rule of Proposition 1. More precisely, one should proceed as follows.

1. Verify each local specification, that is prove that for all $k = 1, \ldots, n$: $\mathcal{AX}_k \wedge \mathcal{AS}_k \Rightarrow \mathcal{TH}_k$. 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, local axioms and theorems, and visible formulas of other classes. In formulas, this corresponds to proving that for all $k = 1, \ldots, n$: $\mathcal{F}_{glob} \wedge \mathcal{AX}_k \wedge \mathcal{TH}_k \wedge \bigwedge_{j=1,\ldots,n \wedge j \neq k} \mathcal{F}_j^{\vee} \Rightarrow \mathcal{AS}_k$.
3. Prove that the global non-rely/guarantee theorems (i.e. not involving the $\twoheadrightarrow$ 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 $\mathcal{AX}_{glob} \wedge \mathcal{AS}_{glob} \wedge \mathcal{TH}_{glob}^{rg} \wedge \bigwedge_{j=1,\ldots,n} \mathcal{F}_j^{\vee} \Rightarrow \mathcal{TH}_{glob}^{nrg}$.
4. Show that each local rely/guarantee formula has an assumption which satisfies the initialization condition (as in hypothesis 1 of Proposition 1). In order to prove the initialization condition, we can use global and local formulas, plus any visible formula of any other class of the system. In formulas, this corresponds to proving that for all $k = 1, \ldots, m$: for all $j = 1, \ldots, n$: if $(E_k \twoheadrightarrow M_k) \in \mathcal{TH}_j^{rg}$ then $\mathcal{F}_{glob} \wedge \mathcal{F}_j \wedge \bigwedge_{i=1,\ldots,n} \mathcal{F}_i^{\vee} \Rightarrow Som(AlwP_e(E_k))$.

5. Show that each local rely/guarantee formula has an assumption 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 hypothesis 2 in Proposition 1. In formulas, this is proving that for all $k = 1, \ldots, m$: for all $j = 1, \ldots, n$: if $(E_k \twoheadrightarrow M_k) \in \mathcal{TH}_j^{rg}$ then: $\mathcal{F}_{glob} \wedge \mathcal{F}_j \wedge \bigwedge_{i=1,\ldots,n} \mathcal{F}_i^{\veebar} \Rightarrow Alw(E_{glob} \wedge \bigwedge_{i=1,\ldots,m} M_i \Rightarrow E_k)$.
6. Show that the global guarantee follows from the local guarantees of all modules and from global formulas and local visible formulas of any class. This corresponds to hypothesis 3 of Proposition 1. In formulas, this is proving $\mathcal{F}_{glob} \wedge \bigwedge_{j=1,\ldots,n} \mathcal{F}_j^{\veebar} \Rightarrow Alw(\bigwedge_{j=1,\ldots,m} M_j \Rightarrow M_{glob})$.
7. Be sure that in all the above proofs there are no circular dependencies among any two closed formulas. Formally, this corresponds to checking that in the set Π of all the above proofs, for all formulas $\phi \in \bigcup_{k=1,\ldots,n}(\mathcal{AS}_k \cup \mathcal{TH}_k) \cup \mathcal{TH}_{glob}$: $\neg(\phi \overset{+}{\rightsquigarrow}_{\Pi} \phi)$.

From the application of the above steps, thanks to the inference rule of Proposition 1 and the absence of circularities, we conclude the validity of the global specification $\mathcal{AX}_{glob} \wedge \mathcal{AS}_{glob} \Rightarrow \mathcal{TH}_{glob}$

4 Compositional Dining Philosophers

This section illustrates an analysis of the *dining philosophers problem*, with the compositional proofs of some relevant properties, as an example of compositional specification and verification in TRIO using the rely/guarantee paradigm. Even if the example does not constitute an "industrial-strength in-the-large" case study, we believe that, after several decades of successful application, it is still an insightful and thought-provoking example to assess the validity of — not only — our compositional rule in the short space of a conference paper. All details of the proofs have been checked with the encoding of the TRIO language in the PVS proof checker [18] (see [12, 10] for some details of this encoding), even if we present them succinctly (due to space limit) and in human-readable form.

4.1 One Rely/Guarantee Philosopher

Assumptions availability, availability_2 and lasting_availability of Section 3.1 express the suppositions 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 two theorems, taking_turns and taking_turns_2 (not explicitly shown), that are analogous to the assumptions availability and availability_2, while assumption lasting_availability corresponds to axiom thinking_duration (see Section 2). For the sake of brevity, the proofs of these two theorems, which are directly derivable from the axioms of the philosopher class only, are not discussed here.

The *local* non-starvation property requires that, assuming a regular availability of the forks, we can guarantee that, after the system starts, the philosopher

eats regularly. This requirement can be formalized using the $\twoheadrightarrow$ operator, relating the availability of the forks in the past to the occurrence of the eating sessions in the immediate future. In our case, $E_k = WithinF_{ei}(UpToNow(\text{available}(l) \wedge \text{available}(r)), T_t + 2T_e)$ and $M_k = SomP_i(\text{start}) \Rightarrow (\exists t > t_e : Within_{ii}(Lasts(\text{eating}, t), T_t + 2T_e))$. The following theorem expresses the local non-starvation property, whose proof we omit for brevity. Notice that the proof assumes that the thinking time of each philosopher is larger than twice the eating time: $T_t > 2T_e$. After all, they are philosophers, not gourmands! (Unless they are Epicureans, one may argue...). This condition allows to avoid the race conditions.

Theorem 3 (regular_eatings_rg)
$WithinF_{ei}(UpToNow(\text{available}(l) \wedge \text{available}(r)), T_t + 2T_e)$
$\twoheadrightarrow \quad (SomP_i(\text{start}) \Rightarrow (\exists t > t_e : Within_{ii}(Lasts(\text{eating}, t), T_t + 2T_e)))$

Up to this point, we have proved that $\mathcal{AX}_{phil} \wedge \mathcal{AS}_{phil} \Rightarrow \mathcal{TH}_{phil}$, which corresponds to step 1 of Section 3.3.

4.2 A Table of Philosophers

The *global* non-starvation property is expressed by theorem liveness_rg below. It simply states that each philosopher in the array eats regularly, unless he/she has not started yet. Notice that in our example $E_{glob} = \text{true}$ since the composite system is closed, and M_{glob} coincides with the following liveness_rg theorem.

Theorem 4 (liveness_rg) $SomP_i(\text{Philosophers[i].start}) \Rightarrow$
$(\exists t > t_e : Within_{ii}(Lasts(\text{Philosophers[i].eating}, t), T_t + 2T_e))$

In Section 4.1 above, we completed step 1 of Section 3.3. Now, let us consider step 2. Each local assumption is discharged by either a global assumption or by a visible theorem or axiom of the modules adjacent to the current philosopher. 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.

Step 3 is empty in our example, since the only global theorem we want to prove is theorem liveness_rg in $\mathcal{TH}^{rg}_{dining\_N}$. Let us consider step 4, where we have to prove that each environment assumption E_k is initialized, that is we have to show that hypothesis 1 of Proposition 1 holds. Since the local theorems have already been proved without circularities, we can use fork_availability to complete this step. The theorem simply states that the desired property $E_k = WithinF_{ei}(UpToNow(\text{available}(l) \wedge \text{available}(r)), T_t + 2T_e)$ always holds, which subsumes the initialization condition. Step 5 requires to discharge the E_k's by means of other formulas, in order to fulfill hypothesis 2 of Proposition 1. Once again, the theorem fork_availability for class 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 $M = \forall k \in \{1, \ldots, n\} : M_k$, so that the implication of this step holds trivially. As a consequence, hypothesis 3 of Proposition 1 is shown to hold.

As discussed above, no circularities arise in proving the local formulas, so we conclude that theorem liveness_rg holds as a consequence of the inference rule of Proposition 1 and following the steps in Section 3.3.

4.3 Complexity of the Proofs

Let us briefly evaluate the complexity of the global proof outlined above, using the proof checker PVS. The PVS proofs of the dining_N system required a total of 949 prover commands; approximately half of them where devoted to the proof of the theorem regular_eatings_rg. Let us compare the cost of this 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 parametrize 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 $\twoheadrightarrow$ operator by temporally closed formulas only. This unstructured solution has been implemented; the result has been a proof of length comparable to 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 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 where 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.

5 Related Works

A compositional analysis technique applies some, possibly formal, method to infer global properties of a large, complex system through a hierarchical and iterative process that exploits the system's modular structure. A general (and historical) introduction to compositional methods can be found in [6, 7]. Without aiming at exhaustiveness, this section briefly reviews some of the most important contributions about compositional reasoning and shows how the approach of this paper differs from them.

An issue still largely unexplored in the present literature on compositionality is the consideration of hard real-time aspects, which require a metric modeling of time. A first noticeable exception is Ostroff in [17], where the metric temporal logic RTTL is embedded in a compositional framework. Nonetheless, the approach is rather different from ours, being focused on refinement aspects rather than on *a posteriori* composition that exploits reuse. Furthermore, time is treated as a separate variable and is discrete, while in our approach time is an implicit item of the language and can be either continuous or discrete.

The need for compositionality has become indisputable in the formal methods community, so that almost every newly introduced formalism encompasses some sort of compositional technique or permits compositional specifications. However, to the best of our knowledge, all proposed compositional frameworks are deeply rooted on some particular, often restrictive semantic assumptions, and depend explicitly on the underlying computational model. In this regard, formalisms typically assume either an interleaving semantics (e.g. [1, 9, 16]) or a synchronous semantics (e.g. [3]) for the concurrent components of the system.

A rather different compositional framework to support the top-down development of real-time systems based on logical formulas at the semantic level is studied by Hooman [13]. In a sense, Hooman's framework is independent of semantic assumptions, even if its set-theoretic model of semantic primitives naturally relates to interleaving semantics models. However, the framework is focused on the refinement (i.e. decomposition) aspect and basically consists of an inference rule that permits to deduce that the decomposition of a module into its refined parts correctly implements the original (unrefined) module. Another important difference between Hooman's framework and ours is that the former does not adopt the rely/guarantee paradigm, and is therefore suitable only to write specifications of modules who do not rely on a constrained behavior of the environment to function correctly.

More typical solutions to the problem of formulating a sound rely/guarantee compositional rule involve the use of an *ad hoc* operator to write rely/guarantee specifications so that they satisfy certain specific characterizations.

For example, the above approach is followed by Abadi and Lamport [1], who analyze the rely/guarantee compositional paradigm using TLA as the reference specification language. The authors introduce the TLA operator $\stackrel{+}{\triangleright}$ to write rely/guarantee specifications that can be soundly composed. Notice that our paper also introduced a suitable operator (the *time progression* operator $\twoheadrightarrow$) to write rely/guarantee specifications. The crucial difference is that our time progression operator is applied in inference rules independently of any assumption on the semantics of processes and also of any semantic characterization of formulas (its application does not need notions such as safety, closure, etc., which are instead integral part of (among others) Abadi and Lamport's framework). This renders our framework purely *syntactic* and very general. In particular, even if the inference rule of [1] is usable for general properties, the conditions of the rule are hard to prove if they are not safety properties; such a distinction does not apply to a syntactic rule such as ours.

Abadi and Merz [2] propose an abstract generalization of rely/guarantee inference rules, in an attempt to treat compositionality syntactically. To this extent, a modal operator to write rely/guarantee inference rules is introduced with minimal semantic assumptions. However, the use of the operator in inference rules and the consequent soundness proofs are possible only *after* the abstract framework is specialized by choosing a semantic model and a computational model. On the contrary, in our framework the soundness of the inference rule is completely proved without assumptions of this kind.

Amla et al. [4] present an abstract compositional framework which can be considered as a generalization of several concrete compositional frameworks in the literature. In particular, they succeed in formulating an inference rule which does not rely on an *ad hoc* operator to be sound, and is therefore simpler than others. However, their framework still relies heavily on semantic assumptions, such as downward closure, on the set of behaviors describing a process. Therefore, our framework does not fit the models in [4], since it pursues the alternative (and new) approach of using a rely/guarantee operator, but independently of any semantic assumptions on the behavior of the components of the system, according to the axiomatic approach.

6 Conclusions

We presented a compositional framework for the TRIO specification language that supports verification through automated theorem proving. The framework is based on a formal notion of composition of TRIO modules, which is used to prove that the mutual interactions between components of a complex system guarantee some property for the global application, after the components are integrated into the system. The compositional rule has been proved sound and has been applied to the classic example of Dijkstra's dining philosophers as a simple, but not simplistic, example. The compositional framework has been encoded into the logic of the PVS theorem prover.

With respect to other approaches to compositionality in formal methods, our own emerges as more suitable for real-time modeling, it encompasses both continuous and discrete time to better model physical processes, and it is conceived for axiom systems and deductive verification. Therefore, the approach is very general and abstracts away from specific assumptions about process semantics and the underlying computational model.

Future work in this line of research will follow three main directions. First, the framework presented here is being applied to several real-life industrial case studies to experimentally evaluate its effectiveness. Second, alternative weaker — or stronger — inference rules will be investigated. In particular, we are exploring variations and generalizations of the $\twoheadrightarrow$ operator, better suited to be applied on certain classes of systems, different inference rules which do not use a time progression operator at all, and complete inference rules (which sacrifice some simplicity of application). Third, automated support for the framework will be improved and extended.

Acknowledgements

The authors thank the anonymous reviewers for their remarks.

References

1. M. Abadi and L. Lamport. Conjoining specifications. *ACM TOPLAS*, 17(3):507–535, 1995.
2. M. Abadi and S. Merz. An abstract account of composition. In *MFCS'95*, volume 969 of *LNCS*, pages 499–508, 1995.
3. R. Alur and T. A. Henzinger. Reactive modules. *Formal Methods in System Design*, 15(1):7–48, 1999.
4. N. Amla, E. A. Emerson, K. S. Namjoshi, and R. J. Trefler. Abstract patterns of compositional reasoning. In *CONCUR'03*, volume 2761 of *LNCS*, pages 431–445, 2003.
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. W.-P. de Roever. The need for compositional proof systems: a survey. In *COMPOS'97*, volume 1536 of *LNCS*, pages 1–22, 1998.
7. W.-P. de Roever, F. de Boer, U. Hannemann, J. Hooman, Y. Lakhnech, M. Poel, and J. Zwiers. *Concurrency Verification: Introduction to Compositional and Noncompositional Methods*. Cambridge University Press, 2001.
8. E. W. Dijkstra. Hierarchical ordering of sequential processes. In *Operating Sys. Tech.*, pages 72–93. 1972.
9. B. Finkbeiner, Z. Manna, and H. B. Sipma. Deductive verification of modular systems. In *COMPOS'97*, volume 1536 of *LNCS*, pages 239–275, 1998.
10. C. A. Furia. Compositional proofs for real-time modular systems. Laurea degree thesis, Politecnico di Milano, 2003.
11. C. A. Furia, M. Rossi, D. Mandrioli, and A. Morzenti. Automated compositional proofs for real-time systems. Full version with appendices available online from `http://www.elet.polimi.it/upload/furia`, 2005.
12. A. Gargantini and A. Morzenti. Automated deductive requirement analysis of critical systems. *ACM TOSEM*, 10(3):255–307, July 2001.
13. J. Hooman. Compositional verification of real-time applications. In *COMPOS'97*, volume 1536 of *LNCS*, pages 276–300, 1998.
14. P. Maier. Compositional circular assume-guarantee rules cannot be sound and complete. In *FOSSACS'03*, volume 2620 of *LNCS*, pages 343–357, 2003.
15. E. Mendelson. *Introduction to Mathematical Logic*. Chapman & Hall, 1997.
16. K. S. Namjoshi and R. J. Trefler. On the completeness of compositional reasoning. In *CAV'00*, volume 1855 of *LNCS*, pages 139–153, 2000.
17. J. S. Ostroff. Composition and refinement of discrete real-time systems. *ACM TOSEM*, 8(1):1–48, 1999.
18. S. Owre, J. M. Rushby, and N. Shankar. PVS: A Prototype Verification System. In *CADE-11*, volume 607 of *LNCS*, pages 748–752, 1992.

Appendix A: Proof of Lemma 1

Lemma 1. *For any formulae P, Q and R, if:*

1. $Som(AlwP_e(P))$
2. $Alw(Q \wedge R \Rightarrow P)$

then: $Alw(P \twoheadrightarrow Q) \Rightarrow Alw(R \twoheadrightarrow Q)$

Proof (Dense time models). By the definition of the $\twoheadrightarrow$ operator, we assume $Alw(P \twoheadrightarrow Q)$ and $AlwP_e(R)$ true at a generic time instant t. We want to prove that $AlwP_i(Q) \wedge NowOn(Q)$ is true at t.

Since $Som(AlwP_e(P))$, let u_0 be the time instant at which $AlwP_e(P)$ holds. If $u_0 \geq t$, by considering $Alw(P \twoheadrightarrow Q)$ we deduce that $AlwP_i(Q) \wedge NowOn(Q)$ is true for all time instants less than or equal to u_0. This simply concludes this branch of the proof.

Let us now assume $u_0 < t$. We know that P is true for all instants less than u_0. Moreover, since $Alw(P \twoheadrightarrow Q)$, there exists some quantity $\epsilon_0 > 0$ such that Q is true for all instants less than $u_1 = u_0 + \epsilon_0$. Now, it is either $u_1 > t$ or $u_1 \leq t$. In the first case the proof is concluded, since we have shown that Q lasts longer than R does. Instead, if $u_1 \leq t$ we can iterate the reasoning to new instants $u_2 = u_1 + \epsilon_1, u_3, \ldots$ until we obtain a $u_n > t$.

We still have to prove that the sequence of points u_i cannot accumulate before time t, but we must eventually get to a u_n such that $u_n > t$. We show this by contradiction: assume that Q holds up to time $\delta < t$ only (note that the following reasoning holds regardless of whether Q holds *exactly* at δ or not, i.e. whether the interval of validity of Q is open or closed to the right). Surely R also holds till δ, since it holds till time t. Now, by hypothesis 2, we realize that P is also true for all time instants less than δ. Since $Alw(P \twoheadrightarrow Q)$, Q lasts at least a bit more than P, contradicting the hypothesis that Q holds till δ only.

Since Q lasts for all instants less than $u_n > t$, the proof is concluded, since it is true that $AlwP_i(Q) \wedge NowOn(Q)$ holds at t. □

Proof (Discrete time models). By the definition of the $\twoheadrightarrow$ operator, we assume $Alw(P \twoheadrightarrow Q)$ and $AlwP_e(R)$ true at a generic time instant t. We want to prove that $AlwP_i(Q)$ is true at t.

Since $Som(AlwP_e(P))$, let u_0 be the time instant at which $AlwP_e(P)$ holds. Let us consider the sequence of points $u_i = u_0 + i$, for all $i = 0, \ldots, (t - u_0)$. By induction on i, we show that $AlwP_i(Q)$ holds at all time instants u_i.

The base case $i = 0$ is subsumed by the hypotheses, since $AlwP_e(P)$ is true at u_0 and $P \twoheadrightarrow Q$ also holds at u_0.

The inductive step requires us to prove that $AlwP_e(Q)$ holds at time u_{i+1}, by assuming that it holds a u_i, for $u_i < t$ (otherwise u_{i+1} is undefined). Since $u_i < t$, it is also $AlwP_i(R)$ at u_i by hypothesis. Then, by hypothesis 2, $AlwP_i(P)$ holds at the same time u_i. From the definition of the $AlwP_i$ operator, and by the discreteness of time, this last fact can be rewritten equivalently as $AlwP_e(P)$ at time $u_i + 1 = u_{i+1}$. Since $Alw(P \twoheadrightarrow Q)$, it follows immediately that $AlwP_i(Q)$ also holds at time u_{i+1}, thus proving the inductive step.

Finally, since $u_{t-u_0} = u_0 + (t - u_0) = t$, we have successfully shown that $AlwP_i(Q)$ holds at t. □

Appendix B: The Philosophers Example

To deal in a uniform and consistent way with some subtle semantic features that are not discussed here for the sake of brevity (see [12]), we assume that all state predicates of the example hold on right-continuous intervals, i.e., for any state S, $S \Leftrightarrow NowOn(S)$.

Note that, in the remainder, the numbering of the formulas does not match the corresponding formulas in the main sections of the paper.

Axioms of the philosopher class

In this section we list the complete axioms of the philosopher class, that is the set $\mathcal{AX}_{\text{phil}}$. We also provide a brief informal explanation for all axioms not mentioned in the main body of the paper.

Axiom 5 (holding_synch) $\text{holding}(l) \Leftrightarrow \text{holding}(r)$

Axiom 6 (hungry) $Lasted(\neg\text{eating}, T_t) \wedge UpToNow(\text{available}(l) \wedge \text{available}(r)) \Rightarrow (\text{take}(l) \wedge \text{take}(r)) \vee NowOn(\neg\text{available}(l) \vee \neg\text{available}(r))$

Axiom 7 (eating_def) $Becomes(\text{holding}(l) \wedge \text{holding}(r)) \Rightarrow (\exists t > t_e : Lasts(\text{eating}, t) \wedge Futr(\text{release}(l) \wedge \text{release}(r), t))$

Axiom 8 (eating_duration) $Lasted(\text{eating}, t) \Rightarrow t < T_e$

Axiom 9 (thinking) $\text{holding}(l) \wedge \text{holding}(r) \Leftrightarrow \text{eating}$

Axiom 10 (thinking_duration) $Becomes(\neg\text{holding}(s)) \Rightarrow Lasts(\neg\text{holding}(s), T_t)$

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

Axiom 11 (taking) $\text{take}(s) \wedge UpToNow(\neg\text{holding}(s)) \wedge UpToNow(\neg\text{available}(s)) \Rightarrow Until(\text{holding}(s), \text{release}(s))$

Axiom 12 (putting) $\text{release}(s) \wedge UpToNow(\text{holding}(s)) \Rightarrow Until(\neg\text{holding}(s), \text{take}(s))$

Axioms start_uniqueness and init respectively state that start is a unique event (i.e. it happens exactly once) and that when a philosopher is "started", he/she is hungry (i.e. he/she has not eaten for more than T_e time units). Although we do not directly use these axioms in the proofs, they complete the specification.

Axiom 13 (start_uniqueness) $Som(\text{start} \wedge AlwP(\neg\text{start}) \wedge AlwF(\neg\text{start}))$

Axiom 14 (init) $\text{start} \Rightarrow Lasted(\neg\text{eating}, T_e)$

Assumptions of the philosopher class

In this section we list the complete assumptions of the philosopher class, that is the set $\mathcal{AS}_{\text{phil}}$.

Assumption 15 (availability) $(\exists t \geq T_t : Lasts(\text{available}(s), t))$
$\vee\, WithinF_{ei}(Becomes(\text{available}(s)), T_t + T_e)$

Assumption 16 (availability_2) $WithinF(UpToNow(\text{available}(s)), T_e)$

Assumption 17 (lasting_availability) $Becomes(\text{available}(s)) \Rightarrow Lasts(\text{available}(s), T_t)$

The requirement that the thinking time is larger than twice the eating time is formalized by assumption TCCs (for "Type Correctness Constraints", *à la* PVS).

Assumption 18 (TCCs) $T_t > 2T_e > 0$

Theorems of the philosopher class

In this section we list the complete theorems of the philosopher class, that is the set $\mathcal{TH}_{\text{phil}}$. We also provide a brief informal explanation for all theorems not mentioned in the main body of the paper, together with their proofs. The interested reader can retrieve the mechanized PVS proofs from `http://www.elet.polimi.it/upload/furia/`.

Theorems taking_turns and taking_turns_2 correspond to assumptions availability and availability_2, respectively. In other words, each philosopher guarantees the requirements stated in the following two theorems, thus satisfying the corresponding assumptions that his/her neighbors make on him/her (i.e. discharging them, by means of the equivalences stated by the connections of the classes).

Theorem 19 (taking_turns) $(\exists t \geq T_t : Lasts(\neg\text{holding}(s), t))$
$\vee\, WithinF_{ei}(Becomes(\neg\text{holding}(s)), T_t + T_e)$

Proof. Let us take $t = T_t$ and show that $Lasts(\neg\text{holding}(s), T_t) \vee WithinF_{ei}($ $Becomes(\neg\text{holding}(s)), T_t + T_e)$ for a generic s. The proof procedes by contradiction: assume the negation of the goal, that is, after some simple rewriting, $WithinF(\text{holding}(s), T_t)$ and $Lasts_{ei}(\neg Becomes(\neg\text{holding}(s)), T_t + T_e)$.

Let us consider the first term $WithinF(\text{holding}(s), T_t)$; by the definition of the $WithinF$ operator, this is the same as saying that there is a time instant $0 < u < T_t$ in the future in which holding(s) is true. It is simple to infer, by axioms holding_synch and thinking, 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 $S \wedge Lasts(\neg Becomes(\neg S), d)$ is true at a certain time, then it follows that $Lasts(S, d)$ 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 $Lasts_{ei}(\neg Becomes(\neg\text{holding}(s)), T_t + T_e)$ implies

that $Lasts(\neg Becomes(\neg \text{holding}(s)), T_e)$ at time u, being $T_e + u < T_e + T_t$. Again, by combining axioms holding_synch and thinking, this last statement implies $Lasts(\neg Becomes(\neg \text{eating}), T_e)$. Hence, we deduce that $Lasts(\text{eating}, T_e)$ holds at time u.

Equivalently, this last fact can be stated at time $u+T_e$ as $Lasted(\text{eating}, T_e)$. It is immediate to derive a contradiction with the statement of axiom eating_duration. □

Theorem 20 (taking_turns_2) $WithinF(UpToNow(\neg \text{holding}(s)), T_e)$

Proof. By axiom eating_duration, we can assume that it is false that $Lasted(\text{eating}, T_e)$ at T_e time instants in the future or, equivalently, that $Lasts(\text{eating}, T_e)$ is false at the current time. By the definition of the $Lasts$ operator, this is equivalent to assuming that $WithinF(\neg eating, T_e)$ is *true* at the current time.

Let $0 < u < T_e$ be the time instant in the future at which eating is false. Since eating is a right-continuous state, it follows that $NowOn(\neg \text{eating})$ is true at u. More explicitly, these facts can be stated as $Lasts_{ie}(\neg eating, v)$ 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 $w = u + \min(T_e - u, v)/2$ (clearly, $w < u + v$ and $w < T_e$ by construction). We realize that $UpToNow(\neg eating)$ holds at w, since it is true that $Lasted(\neg eating, \min(T_e - u, v)/2)$ holds at w.

Finally, by considering axioms holding_synch and thinking, we deduce the statement $UpToNow(\neg \text{holding}(s))$ at $w < T_e$, which concludes the proof. □

Theorem 21 (fork_availability)
$WithinF_{ei}(UpToNow(\text{available}(l) \wedge \text{available}(r)), T_t + 2T_e)$

Proof. Let us consider two instances of assumption availability, one for the left fork and one for the right fork, and fix $t = T_t$. According to the disjunction, the proof can be split into four cases, whether:

1. $Lasts(\text{available}(l), T_t)$ and $Lasts(\text{available}(r), T_t)$
2. $Lasts(\text{available}(l), T_t)$ and $WithinF_{ei}(Becomes(\text{available}(r)), T_t + T_e)$
3. $Lasts(\text{available}(r), T_t)$ and $WithinF_{ei}(Becomes(\text{available}(l)), T_t + T_e)$
4. $WithinF_{ei}(Becomes(\text{available}(l)), T_t + T_e)$ and $WithinF_{ei}(Becomes(\text{available}(r)), T_t + T_e)$,

all formulas holding at the current time.

Let us first consider the simpler case 1. Let us consider the time instant $u = T_t/2$ in the future. We claim that $UpToNow(\text{available}(l) \wedge \text{available}(r))$ holds at u; since $u < T_t + 2T_e$ this would conclude this part of the proof. In fact, obviously $Lasts(\text{available}(l) \wedge \text{available}(r), T_t/2)$ holds, being an immediate consequence of the hypotheses. Equivalently, this can be stated as $Lasted(\text{available}(l) \wedge \text{available}(r), T_t/2)$ at time u. By considering the definition of the $UpToNow$ operator, this implies $UpToNow(\text{available}(l) \wedge \text{available}(r))$ holds at u, what we just claimed.

Let us now pass to case 2 and just assume $WithinF_{ei}(Becomes(\text{available}(r)), T_t+T_e)$ at the current time. Now, let $0 < u < T_t+T_e$ be the time instant at which $Becomes(\text{available}(r))$ holds. By assumption lasting_availability, it follows that $Lasts(\text{available}(r), T_t)$ 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 $WithinF(UpToNow(\text{available}(l)), T_e)$ holds at u. Let $u + v < u + T_e$ be the time instant at which $UpToNow(\text{available}(l))$ holds. Since $T_e < T_t/2$, it follows that $u + v < u + T_t$ and therefore *a fortiori* $Lasts(\text{available}(r), v)$ holds at u. This last formula implies that $UpToNow(\text{available}(r))$ holds at $u + v$. All in all, $UpToNow(\text{available}(l) \wedge \text{available}(r))$ holds at $u + v$. Now, since $u + v < (T_t + T_e) + T_e = T_t + 2T_e$, this branch of the proof is successfully concluded.

Case 3 has a proof which is all similar to case 2, because of the simmetry 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 $Lasts(\text{available}(s), T_t)$ in the proof. □

The following theorem is an intermediate result, needed in the proof of the theorem regular_eatings_rg. Its informal meaning is the following: the philosopher is always in one of these two situations: he/she is hungry (i.e. T_t time units without eating have passed), or no more than $T_t + T_e$ time units have elapsed since the last time he/she ate or he/she began eating.

Theorem 22 (always_eating_or_not) $Lasted(\neg\text{eating}, T_t) \vee WithinP_{ii}(Becomes(\text{eating}) \vee (\exists t > t_e : Lasted(\text{eating}, t)), T_t + T_e)$

Proof. In order to prove the disjunction, we assume the negation of the first term as hypothesis. Therefore, we assume $\neg Lasted(\neg\text{eating}, T_t)$ at the current time or, equivalently, $WithinP(\text{eating}, T_t)$. Writing the $WithinP$ out explicitly, this means that there exists a time distance $0 < u < T_t$ such that eating holds at $-u$.

Being eating a right-continuous state, we also have that $NowOn(\text{eating})$ at $-u$, that is $Lasts(\text{eating}, v)$ 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 $UpToNow(\text{eating})$ and $NowOn(\text{eating})$ at $-w$.

Now, for any right-continuous state predicate S such that $NowOn(S)$, it can be proved[5] 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. $\exists p > 0 : Lasts(S, p) \wedge Futr(NowOn(\neg S), p)$). 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. $\exists p > 0 : Lasted(S, p) \wedge Past(NowOn(\neg S) \vee UpToNow(\neg S), p)$).[6] Therefore, for

[5] A weaker but similar result holds for generic time-dependent predicates, but we do not need it here. We do not discuss neither proof.

[6] The asymmetry with the future case is due to the fact that S is *right*-continuous.

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, $T_e)$ at $-w$. Since $T_e > t_e$ and $w < u < T_t < T_t+T_e$, clearly $WithinP_{ii}(\exists t >$ $t_e : Lasted$(eating, $t), T_t + T_e)$ 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 $p > T_e$, it is true that $Lasted$(eating, T_e) at $-w > -(T_t + T_e)$, so the goal $WithinP_{ii}(\exists t > t_e : Lasted$(eating, $t), T_t + T_e)$ is satisfied. On the contrary, if $p \leq T_e$, 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 $p \leq T_e$. Therefore $UpToNow(\neg$eating$) \wedge NowOn$(eating) at $-w-p > -(u+T_e) > -(T_t+T_e)$, 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). □

Theorem 23 (regular_eatings_rg)
$WithinF_{ei}(UpToNow($available$(l) \wedge$ available$(r)), T_t + 2T_e)$
$\twoheadrightarrow \quad (SomP_i($start$) \Rightarrow (\exists t > t_e : Within_{ii}(Lasts($eating, $t), T_t + 2T_e)))$

Proof. Because of the definition of the $\twoheadrightarrow$ operator, we assume $AlwP_e(WithinF_{ei}$ $(UpToNow($available$(l) \wedge$ available$(r)), T_t + 2T_e))$ as hypothesis and set our goal to proving $AlwP_i(F)$ and $NowOn(F)$ separately, where F is the formula $(\exists t > t_e : Within_{ii}(Lasts($eating, $t), T_t + 2T_e)) \vee AlwP_i(\neg$start), an equivalent statement of the implication. First of all, we notice that, because of theorem fork_availability, we can actually strengthen our current hypothesis to be $AlwP_i(WithinF_{ei}(UpToNow($available$(l) \wedge$ available$(r)), T_t + 2T_e))$, that is including the current instant. Now, let us first prove $AlwP_i(F)$ 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 at time u and make case discussion on it. The proof is split into two branches whether $Lasted(\neg\text{eating}, T_t)$ or $WithinP_{ii}(Becomes(\text{eating}) \vee \ldots, T_t + T_e)$ holds at u.

The first branch considers the hypothesis instantiated at time u, that is $WithinF_{ei}(UpToNow(\text{available}(l) \wedge \text{available}(r)), T_t + 2T_e)$ at u. Therefore, let us make explicit the existentially quantified time variable of the $WithinF$ operator and name it f. Notice that $0 < f \leq T_t + 2T_e$ and we can write that $UpToNow(\text{available}(l) \wedge \text{available}(r))$ at time $u+f$. Now, the proof is further split into two branches whether the state eating never becomes true for all time instants since u to $u + f$. In the first case $Lasts_{ie}(\neg Becomes(\text{eating}), f)$ 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 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 hungry at this time. We immediately conclude that both forks are taken at $u+f-\epsilon$ and therefore $Becomes(\text{holding}(l) \wedge \text{holding}(r))$ is true at $u + f - \epsilon$. Furthermore, we can consider axiom eating_def at time $u + f - \epsilon$ and deduce that there exists $r > t_e$ such that $Lasts(\text{eating}, r)$ holds at $u + f - \epsilon$. Since $|f - \epsilon| \leq T_t + 2T_e$ 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 $Becomes(\text{eating})$ is true. It is simple to understand how the remainder of this branch is concluded. Since eating becomes true at g, we can apply axiom eating_def and deduce that there exists $r > t_e$ such that $Lasts(\text{eating}, r)$ holds at g. Since $u \leq g < u + f \leq u + (T_t + 2T_e)$, this branch of the proof is also concluded.

Let us now consider the case in which $WithinP_{ii}(Becomes(\text{eating}) \vee (\exists t > t_e : Lasted(\text{eating}, t)), T_t + T_e)$ holds at u. This means that there is a generic time instant $v : u - (T_t + T_e) \leq v \leq u$ where $Becomes(\text{eating})$ or $(\exists t > t_e : Lasted(\text{eating}, t))$ holds. In the first case, axiom eating_def lets us conclude that eating lasts for a time at least as long as t_e starting from time instant v. Since $v \geq u - (T_t + T_e) \geq u - (T_t + 2T_e)$, this branch of the proof is concluded. In the second case, eating was true for $r > t_e$ 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. □

Connections Between Modules

We compose $N \geq 2$ instances of the philosopher class into the new composite class dining_N. Obviously, this class is parametric with respect to the number of philosophers N and with respect to the parameters t_e, T_e, T_t, whose actual values can be chosen freely, provided they satisfy some TCCs, $T_t > 2T_e \wedge T_e > t_e > 0$, as that of the component modules. 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(s) item of each philosopher corresponds to the negation of the holding(s) item of the philosopher on his/her left/right. The connections can be defined by the TRIO formula: (i is of type

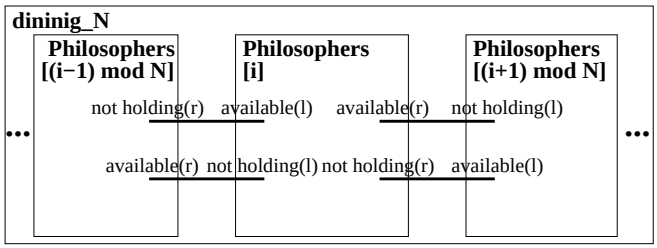

Fig. 2. Interface of the dining_N class

$[0..N-1]$, and $(i \pm 1) \mod N$ indicates the philosopher on the right/left):

```
connections:
  forall i: [0..N-1]:
  (direct Philosophers[i].available(l),
          not Philosophers[(i-1) mod N].holding(r))
  (direct Philosophers[i].available(r),
          not Philosophers[(i+1) mod N].holding(l))
```

Another fine-grain issue regards mutual exclusion between global events from different modules, in order to avoid meaningless situations where two philosophers issue a take action at the same time. This is would be expressed by a global axiom serialization, which is not listed here explicitly for the sake of brevity. Notice that this axiom is not used directly in the verification of the system, but it is nonetheless important to rule out of the model situations which do not correspond to a "real" system.

Proof Dependencies in the philosopher and dining_N Classes

The complete proof dependencies for the class philosopher are shown in Figure 3 (the weights on the arcs correspond to the number of times a formula has been instantiated in the proof of the other formula; no weight means 1).

The global dependencies in the discharging of assumption formulas in the overall system is instead shown in Figure 4.

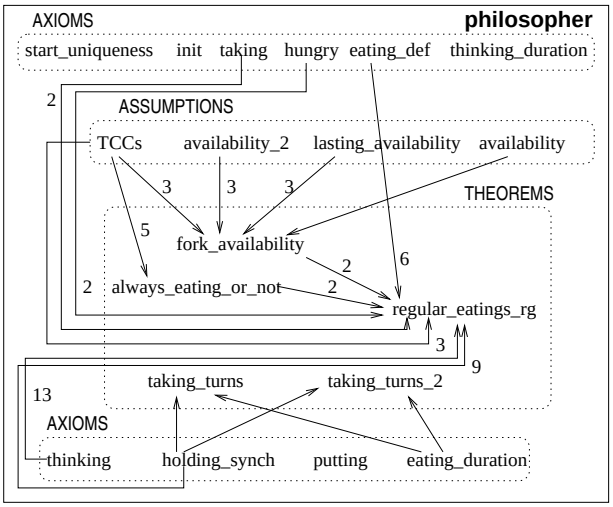

Fig. 3. Proof dependencies in class philosopher

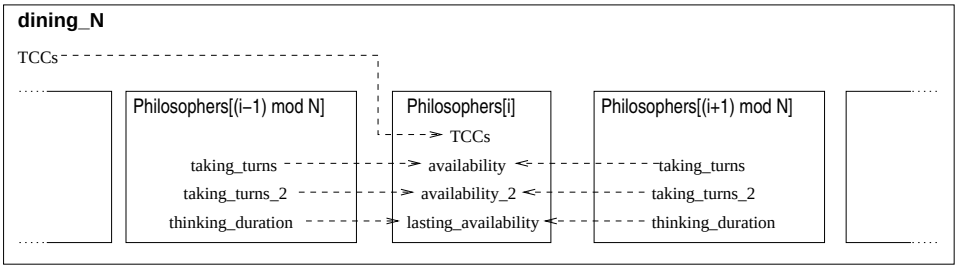

Fig. 4. Proof dependencies in global class dining_N