Embedding time granularity in a logical specification language for synchronous real-time systems*

E. Ciapessoni and E. Corsetti
CISE, Tecnologie Innovative spa, Via Reggio Emilia 39, 20090 Segrate (MI), Italy

A. Montanari
Dipartimento di Matematica e Informatica, Università di Udine, Via Zanon 6, 33100 Udine, Italy

P. San Pietro
Dipartimento di Elettronica e Informazione, Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milano, Italy

Abstract

Formal methods have proved to be highly beneficial in the requirements specification phase of software production and are particularly valuable in the development of real-time applications (the most critical software systems). Unfortunately, most common specification languages are inadequate for real-time applications because they lack a quantitative representation of time. In this paper, we define a logical language to specify the temporal constraints of the wide-ranging class of real-time systems whose components have dynamic behaviours regulated by very different time constants. We motivate the need for allowing the consistent treatment of different time scales in formal specifications of these systems with the purpose of enhancing the naturalness and practical usability of the notation. The logical specification language is based on a revised version of the specification language TRIO. We first present the features of the basic logical language; then, we semantically and axiomatically define its granularity extension in a topological logic framework. Finally, we show some examples of its application.

1. Introduction

Formal methods have proved to be highly beneficial in the requirements specification phase of a software production process [14, 38]. They are particularly...
valuable in the development of real-time applications, which are among the most critical software systems. Plants or weapon control devices, “fly by wire” aircraft, time-critical information systems, and embedded applications are only some examples of the important family of real-time systems. Unfortunately, most common specification languages [3, 11, 16, 17, 20] are inadequate for real-time applications: they cannot deal with temporal properties in a simple and satisfactory way, because they lack an explicit and quantitative representation of time. A few remarkable exceptions, however, do exist. They are extensions of Petri nets [15, 24, 32] or versions of temporal logic [13, 30], which support direct and quantitative specifications of temporal properties and relevant validation activities.

There are, however, systems whose temporal specification is far from being simple even with timed Petri nets or metric temporal logic. In this paper we focus on a wide-ranging class of such systems: the systems whose components have dynamic behaviours regulated by very different—even by orders of magnitude—time constants (hereinafter granular systems). For instance, a pondage power station consists of a reservoir, with filling and emptying times of days or weeks, generator units, possibly changing state in a few seconds, and electronic control devices, evolving in milliseconds or even less. A complete specification of the power station must include the description of these components and of their interactions. A natural description of the temporal evolution of the reservoir state will probably use days: “During rainy weeks, the level of the reservoir increases 1 meter a day”. The description of the control devices behaviour may use microseconds: “When an alarm comes from the level sensors, send an acknowledge signal in 50 microseconds”. We say that systems of such a type have different time scales. It is somewhat unnatural to compel the specifier of these systems to use a unique time scale, microseconds in the previous example, to describe the behaviour of all the components. For instance, the specifier of the requirements for a pondage power plant should not be compelled to write sentences like “the filling of the reservoir must be completed within n microseconds”. A good language must allow the specifier to easily describe all simple and intuitively clear facts. A major issue of specification languages is in fact the naturalness of the notation. Then, different time granularities must be a feature of a specification language for granular systems.

Despite the widespread recognition of its relevance in the fields of formal specifications, knowledge representation and reasoning, and temporal databases, there is a lack of a systematic framework for time granularity. At the best of our knowledge, time granularity or related concepts have been discussed in [5, 10, 12, 18, 27, 35, 36, 39]. Hobbs [18] proposes a formal characterization of the general notion of granularity, but gives no special attention to time granularity. He only sketches out a rather restrictive mapping of continuous time into discrete times using the situation calculus formalism. Clifford et al. [5] provide a set-theoretic formalization of time granularity, but they do not attempt to relate the truth value of assertions to time granularity. Galton [12] and Shoham [36] give significant categorizations of assertions based on their temporal properties that are strictly
related to the concept of time granularity even if it is not explicitly considered. Finally, extensions to existing languages for formal specifications, knowledge representation, and temporal databases to support a limited concept of time granularity are proposed by Roman [35], Evans [10] and Montanari et al. [27], and Wiederhold et al. [39], respectively.

In this paper, we define a logical specification language embedding the notion of time granularity that allows the user to build synchronous, granular system specifications by referring to the "natural" time scale in any component of the specification, even if these are quite different from each other. At the same time, we preserve the full rigor of formal languages that allow us to associate a precise semantics with any formula.

The rationale of the introduction of time granularity in the specification of granular systems, together with the identification of the main representational requirements it imposes, are presented in [6, 7]. A first attempt of extending logical specification languages for incorporating time granularity is reported in [8, 9, 25]. It basically consists of translation mechanisms that map a formula associated with a given time scale into a corresponding formula associated with a finer one. In such a way, a model of a specification involving different granularities can be built by translating everything to the finest granularity. In this paper, we substantially revise such an approach. We extend the basic logical language with contextual and projection operators that deal with time granularity, and provide the resulting language with a model-theoretic semantics. We also give a sound axiomatization of the extended language. The proposed semantics expresses more general and complete properties of time granularity than the transformational semantics given before. Besides, the axiomatic system provides a better clarification of the meaning of time granularity and gives the possibility of doing inferences from a granular specification.

The paper is organized as follows. Section 2 presents the syntax and the semantics of the basic logical language, together with its axiomatization. The basic language is a revised, axiomatic version of TRIO, a logical language for executable specifications of real-time systems [13, 28]. Section 3 discusses time granularity issues in detail and points out the steps required to extend the basic language with time granularity. Section 4 formally defines syntax and semantics of the extended language, together with its axiomatization. Section 5 gives some examples of temporally layered specifications. Conclusions provide an assessment of the proposed approach, discuss open issues, and outline possible extensions. Montanari [26] and Ciapessoni [4] collect formal definitions and proofs of stated results.

2. The basic logical formalism

The basic logical formalism is a revised, many-sorted version of the logical specification language TRIO, a first-order logic language augmented with temporal operators and a metric on time. Similarly to standard temporal logics, e.g. [31, 34],
it is provided with a *temporal operator* that allows one to talk about truth and falsity of formulae at time instants different from the current one that is left implicit. Each formula is interpreted over a totally ordered temporal domain, and its truth value depends on its assertion time (*chronologically undefined formula*). In contrast to standard formalisms the formulae of the language may include *explicit quantifications* over time and *metric* temporal constraints. The last feature enables one to express quantitative and qualitative temporal properties over both discrete and dense time structures, including maximal, exact, and minimal temporal distances between events, periodicity, bounded response time, etc. In this respect, the language is quite similar to topological (metric) temporal logics [22, 33].

2.1. **Basic syntactic features**

In this section, we first briefly introduce alphabet, terms, and formulae of the language; then, we define the basic concepts of specification and history, and give an example of real-time system specifications.

2.1.1. **Alphabet**

The alphabet of the language includes *sorts, variables, constants, functions, predicates,* and *logical constants.* The sorts denote the domains over which variables, constants, functions, and predicates take value. The set of sorts includes a particular sort $s_T$, called the temporal sort, which is numerical in nature and denotes the set of values of temporal displacements. Depending on the specified system, $s_T$ can be either the set of integers, or the set of rational numbers, or the set of real numbers, or a subinterval of them. All constant, function, and predicate symbols are typed as well as variables. The type of an $n$-ary function is a pair $(t, s)$, where $t$ is the $n$-tuple of domain sorts and $s$ is the codomain sort. The type $t$ of an $n$-ary predicate is an $n$-tuple of sorts. We assume that the function symbols $-, \oplus, \ast, \ldots,$ the equality symbol $=$, and the usual relational symbols $<, >, \leq, \geq, \ldots$ are predefined for the temporal sort $s_T$ and, more generally, for each numerical sort.$^1$

The set of logical constants includes the usual propositional connectives $\neg$ and $\Rightarrow$, the quantifier $\forall$ and the parametrized temporal displacement operator $\forall_\alpha$, where $\alpha$ is of sort $s_T$.

2.1.2. **Terms and formulae**

The syntax of the language is given as usual by inductively defining its terms and formulae. Terms are defined in a mutually recursive fashion. Let $\mathcal{STerm}_s$ denotes the set of terms of sort $s$ and $\mathcal{STerm}$ the set of all terms. $\mathcal{STerm}_s$ includes all variables $x$, all constants $c$, and all functions of type $(t, s)$ applied to $n$ terms of the proper sorts. $\mathcal{STerm}$ is the union set of all $\mathcal{STerm}_s$. Terms of sort $s_T$ are called temporal terms. The set of formulae includes all predicates of type $t$ applied to $n$ terms of the proper sorts; all equalities between terms of the same sort, e.g. $t_1 = t_2$, where

$^1$ The properties of $\oplus$ are given in Section 2.3.
Let us now formalize the notion of *specification* of a real-time system and the related notion of *history*.

First of all, we assume that constants, functions, and predicates are time-dependent, while variables are time-independent. However, it can be useful to constrain
a subset of the set of constants, functions, and predicates to be time-independent. Time-independent constants and functions represent values unrelated with time, i.e., values that are not subject to change in time. Time-independent predicates represent properties which can be assumed not to change in time. This is the case, for example, of the equality and of the usual ordering relations. The axiomatization of time independency conditions for constants, functions, and predicates is given in Section 2.3.

On the basis of the notions of time dependency and time independency, we define the closure of a formula. We say that a formula is classically closed if and only if all its variables are quantified, and that it is temporally closed if and only if it does not include time-dependent constants, functions, or predicates, or it has either Always or Sometimes as its outermost operator, or it results from propositional compositions of temporally closed formulae, or it is the classical closure of a temporally closed formula.

A specification \( \Sigma \) of a real-time system is a classically and temporally closed formula of the language.

As an example, assuming a closed temporal domain, a communication channel that outputs each message with a delay \( t \) with respect to its input time and that neither generates nor loses messages can be specified as follows:

\[
\forall \text{msg} (\text{Always}(\text{out} = \text{msg}) \equiv \nabla \text{-}(\text{in} = \text{msg}))
\]

where \( \text{out} \) and \( \text{in} \) are time-dependent constants.

A history \( H \) models a temporal evolution of the specified system by constraining the temporal relations between atomic formulae representing occurring events or system states. Formally, a history \( H \) is a formula of the form:

\[
\text{Sometimes} \left( \bigwedge_i \nabla_\alpha_i F_i \right)
\]

where, for each \( i \), \( F_i \) is an atomic formula of the type:

1. time-dependent predicate applied to time-independent ground terms;
2. equality of the form \( c = t \), where \( c \) is a time-dependent constant of sort \( s \) and \( t \) is a time-independent ground term of the same sort;

and \( \alpha_i \) is a time-independent ground temporal term.

2.2. Basic semantic features

The semantics of the language is based on the concept of temporal structure that allows us to derive the notions of state and valuation function. The state is an assignment of suitable values to constants, functions, and predicates at each time

\[^3\text{It is easy to generalize time independency from constants and functions to terms. We say that a term is time-independent if and only if it is a variable, or it is a time-independent constant, or it is an } n\text{-ary time-independent function applied to } n\text{ time-independent terms.}\]
instant. The *valuation function* is an assignment of a value to terms and formulae at each time instant. Formally, a *temporal structure* $S$ is a triplet:

$$S = (D, \mathcal{F}, \mathcal{I})$$

where

- $D$ is a family of non-empty sets called the *domain of interpretation*: $D = \{D_s : s \in St\}$, where $St$ denotes the sets of sorts. It includes the metric domain $D_{\mathcal{F}}$ over which temporal terms are interpreted.
- $\mathcal{F}$ is the temporal domain, numerical in nature, over which is defined a partial function $\mathcal{+}$. Such a function maps each time instant $t$ of $\mathcal{F}$ and a relevant temporal displacement $d$ of $D_{\mathcal{F}}$ into the time instant $t_d$ of $\mathcal{F}$ that is $d$ time units away from $t$ (if any).\(^4\)
- $\mathcal{I}$ is the interpretation function that assigns a value to variables $x$, constants $c$, functions $f$, and predicates $p$ such that:

  for each variable $x \in S\text{Var}$,
  $$x \mapsto \mathcal{I}(x) \in D_x,$$

  for each constant symbol $c \in S\text{C}$,
  $$c \mapsto \mathcal{I}(c) : \mathcal{F} \to D_x,$$

  for each function symbol $f \in S\text{Fn}_{(s,t)}$ and $t = (s_1, \ldots, s_n)$
  $$f \mapsto \mathcal{I}(f) : \mathcal{F} \to (D_{s_1} \times \cdots \times D_{s_n}) \to D_x,$$

  for each predicate symbol $p \in S\text{Pr}$ and $t = (s_1, \ldots, s_n)$
  $$p \mapsto \mathcal{I}(p) : \mathcal{F} \to 2^{D_{s_1} \times \cdots \times D_{s_n}},$$

where $S\text{Var}$, $S\text{C}$, $S\text{Fn}_{(s,t)}$, and $S\text{Pr}$ denote the set of variables of type $s$, the set of constants of type $s$, the set of functions of type $\langle t, s \rangle$, and the set of predicates of type $t$, respectively. The value of a constant $c$, a function $f$, and a predicate $p$ at a time instant $i$ of $\mathcal{F}$ are denoted by $\mathcal{I}_i(c)$, $\mathcal{I}_i(f)$, and $\mathcal{I}_i(p)$, respectively.

$S$ defines a set of interpretations that differ from each other in the time instant of the temporal domain they assign to the implicit current instant. On the basis of $\mathcal{I}$, we can give a value to each term and formula of the language at each time instant $i$ of $\mathcal{F}$:

- if $x$ is a variable then $\mathcal{I}_i(x) = \mathcal{I}(x)$;

- if $f$ is an $n$-ary function and $t_1, \ldots, t_n$ are terms then
  $$\mathcal{I}_i(f(t_1, \ldots, t_n)) = \mathcal{I}_i(f)(\mathcal{I}_i(t_1), \ldots, \mathcal{I}_i(t_n));$$

- if $p$ is an $n$-ary predicate and $t_1, \ldots, t_n$ are terms then
  $$\mathcal{I}_i(p(t_1, \ldots, t_n)) = \text{true} \iff (\mathcal{I}_i(t_1), \ldots, \mathcal{I}_i(t_n)) \in \mathcal{I}_i(p);$$

- $\mathcal{I}_i(t_1 = t_2) = \text{true} \iff \mathcal{I}_i(t_1) = \mathcal{I}_i(t_2)$

\(^4\) In the semantic clause for $\forall \mathcal{F} \mathcal{I}$ we implicitly refer to a superset $\mathcal{F}'$ of $\mathcal{F}$ over which the function $\mathcal{+}$ is total to verify if it is also defined over $\mathcal{F}$. 

---

*Embedding time granularity in a logical specification language* 147

---
where \(=\) is the identity relation in \(D,\);

\[\mathcal{I}_i(\neg \mathcal{F}) = \text{true} \iff \text{not } \mathcal{I}_i(\mathcal{F}) = \text{true};\]

\[\mathcal{I}_i(\mathcal{F} \supset \mathcal{F'}) = \text{true} \iff \text{not } \mathcal{I}_i(\mathcal{F}) = \text{true} \text{ or } \mathcal{I}_i(\mathcal{F'}) = \text{true};\]

\[\mathcal{I}_i(\forall x \mathcal{F}) = \text{true} \iff \mathcal{I}'_i(\mathcal{F}) = \text{true}\]

for each \(\mathcal{I}'_i\) that differs from \(\mathcal{I}_i\) at most in the value it assigns to \(x\);

\[\mathcal{I}_i(\nabla_\alpha \mathcal{F}) = \text{true} \iff \text{if } i + \mathcal{I}_i(\alpha) \in \mathcal{T} \text{ then } \mathcal{I}_{i + \mathcal{I}_i(\alpha)}(\mathcal{F}) = \text{true}.\]

Notice that from the previous interpretation it follows that \(\nabla_\alpha \mathcal{F}\) is true when \(i + \mathcal{I}_i(\alpha)\) does not belong to \(\mathcal{T}\). From the definition of \(\Delta_\alpha\), it follows that:

\[\mathcal{I}_i(\Delta_\alpha \mathcal{F}) = \text{true} \iff i + \mathcal{I}_i(\alpha) \in \mathcal{T} \text{ and } \mathcal{I}_{i + \mathcal{I}_i(\alpha)}(\mathcal{F}) = \text{true}.\]

As anticipated, this clause states that \(\Delta_\alpha \mathcal{F}\) is false if a time instant \(t\) at distance \(\alpha\) from \(i\) does not exist. It is easy to see that the two displacement operators are equivalent for closed domains, e.g. cyclic domains, while they differ for open domains, e.g. finite, acyclic domains.

Let us now define the notions of temporal satisfiability, validity, and invariance of formulae with respect to a temporal structure. A formula \(\mathcal{F}\) is said to be **temporally satisfiable** with respect to a temporal structure \(S\) if and only if it evaluates to true in at least one instant of the temporal domain. In such a case, we say that the temporal structure provides a *model* for the formula. A formula \(\mathcal{F}\) is said to be **temporally valid** with respect to a temporal structure \(S\) if and only if it evaluates to true in every instant of the temporal domain. Finally, a formula \(\mathcal{F}\) is said to be **temporally invariant** with respect to a temporal structure \(S\) if and only if it is temporally unsatisfiable or temporally valid.

It is possible to prove that each temporally closed formula (and, then, each specification) is temporally invariant [28]. This can be intuitively understood in the case of formulae having *Always* (or *Sometimes*) as their outermost operators by considering that these operators provide a way to universally (or existentially) quantify the current time left implicit in the formulæ. The main consequence of this theorem is that to prove the temporal validity of a temporally invariant formula it is sufficient to prove its temporal satisfiability. Notice that, however, a temporally closed formula can be temporally valid with respect to a given temporal structure and temporally unsatisfiable with respect to another.

On the basis of these concepts, we define the notions of **satisfiability** and **validity** of formulae. A formula \(\mathcal{F}\) is said to be **satisfiable** if and only if there exists a temporal structure with respect to which \(\mathcal{F}\) is temporally satisfiable, while it is said to be **valid** if and only if it is temporally valid with respect to every temporal structure.

### 2.3. Language axiomatization

The basic properties of the language are expressed by axioms and inference rules of first-order predicate calculus with equality together with the following axiom schemata (hereinafter axioms):
Embedding time granularity in a logical specification language

**Ax1.** $\forall_{\alpha}(\mathcal{T} \supset \mathcal{G}) \supset (\forall_{\alpha}\mathcal{T} \supset \forall_{\alpha}\mathcal{G})$ (normality of $\forall_{\alpha}$)

**Ax2.** $\forall_{\alpha}\forall_{\alpha}\mathcal{T} \supset \forall_{\alpha}\forall_{\alpha}\mathcal{T}$

if $x$ is not in $\alpha$ (Barcan's formula for $\forall_{\alpha}$)

**Ax3.** $\forall_{\alpha}\forall_{\alpha}\mathcal{T} \supset \forall_{\alpha}\forall_{\alpha}\mathcal{T}$

if $x$ is not in $\alpha$ (Barcan's formula for $\Delta_{\alpha}$)

**Ax4.** $\forall_{\alpha}\forall_{\alpha}\mathcal{T} \supset \forall_{\alpha}\forall_{\alpha}\mathcal{T}$ (quasi-functionality)

and the following inference rule:

**IR1.** $\vdash \mathcal{T} \rightarrow \forall_{\alpha}\mathcal{T}$ (necessitation rule for $\forall_{\alpha}$)

Axioms Ax2 and Ax3 state that the interpretation domain does not change under temporal displacement, i.e., it is time-independent. These axioms should be weakened to deal with the creation and deletion of objects. This problem is addressed in its full extent in the free logic literature, e.g. [23].

Axiom Ax4 states that if a time instant $t$ at distance $\mathcal{D}_{i}(\alpha)$ from the current instant $i$ exists, then such an instant is unique.

Let us report now a number of interesting theorem schemata (hereinafter theorems).

First of all, given the definition of $\Delta_{\alpha}$, it is immediate to prove that:

$\Delta_{\alpha}\mathcal{T} = \neg\forall_{\alpha}\neg\mathcal{T}$

and its corollaries:

$\forall_{\alpha}\neg\mathcal{T} = \neg\Delta_{\alpha}\mathcal{T}$

$\neg\forall_{\alpha}\mathcal{T} = \Delta_{\alpha}\neg\mathcal{T}$

Such theorems, together with the usual substitution rule of equivalents, allow us to replace $\forall_{\alpha}$ with $\neg\Delta_{\alpha}\neg$, and vice versa, in any formula.

Axiom Ax1, together with the inference rule IR1, allows us to deduce the distributivity of $\forall_{\alpha}$ with respect to $\wedge$ and then, by duality, the distributivity of $\Delta_{\alpha}$ with respect to $\vee$. Then, from Ax1, IR1, the distributivity of $\forall_{\alpha}$ with respect to $\vee$, and the duality of $\forall_{\alpha}$ and $\Delta_{\alpha}$, it follows that:

$\forall_{\alpha}\mathcal{T} \supset (\Delta_{\alpha}\mathcal{T} \equiv \Delta_{\alpha}\forall)$

Furthermore, Ax4, together with the distributivity of $\Delta_{\alpha}$ with respect to $\vee$ and the duality of $\forall_{\alpha}$ and $\Delta_{\alpha}$, allows us to derive that:

$\Delta_{\alpha}\mathcal{T} \equiv (\Delta_{\alpha}\forall \wedge \forall_{\alpha}\mathcal{T})$. 

From this last theorem, it is easy to prove the distributivity of $\Delta_\alpha$ with respect to $\land$ and then, by duality, the distributivity of $\nabla_\alpha$ with respect to $\lor$.

Besides the basic axioms $Ax1-Ax4$, we require the existence of a zero displacement that does not change the current time instant and express the compositional properties of the displacement operators in terms of $\oplus$ function properties.

We first require that the application of a zero displacement to a formula does not change it:

$$Ax5. \quad \nabla_0 \mathcal{F} = \mathcal{F}$$

(existence of a zero element)

This implies [33]:

$$\forall x \nabla_x \mathcal{F} \supset \mathcal{F}$$

provided that $x$ does not occur in $\mathcal{F}$. Then we state that temporal displacements are compositional provided that there exists a time instant $\alpha$ units from the current one:

$$Ax6. \quad \Delta_\alpha \lor \supset \nabla_\alpha \nabla_\beta \mathcal{F} \equiv \nabla_{\alpha \oplus \beta} \mathcal{F}$$

(vector addition for $\nabla_\alpha$)

where $\alpha$ and $\beta$ are time-independent terms of sort $s_T$ and $\oplus$ is a function whose properties are specified by the following axioms:

$$Ax7. \quad \alpha \oplus \beta = \beta \oplus \alpha$$

(commutativity)

$$Ax8. \quad \alpha \oplus (\beta \oplus \gamma) = (\alpha \oplus \beta) \oplus \gamma$$

(associativity)

$$Ax9. \quad \alpha \oplus 0 = \alpha$$

(identity)

$$Ax10. \quad \alpha \oplus -\alpha = 0$$

(inverse)

The vector addition for $\Delta_\alpha$ can be easily derived from axiom $Ax6$ by duality. The closure of the temporal domain with respect to temporal displacements\(^5\) is obtained by replacing axioms $Ax4$ and $Ax6$ with

$$Ax4'. \quad \Delta_\alpha \mathcal{F} = \nabla_\alpha \mathcal{F},$$

$$Ax6'. \quad \nabla_\alpha \nabla_\beta \mathcal{F} = \nabla_{\alpha \oplus \beta} \mathcal{F}$$

Combined with the previous axioms, $Ax4'$ allows us to conclude that:

$$\nabla_\alpha (\neg \mathcal{F}) \equiv \neg \nabla_\alpha \mathcal{F}$$

\(^5\) Such a closure is implicitly assumed in [33].
where $\alpha$ is any temporal term, and then, given the distributivity of $\forall_\alpha$ with respect to $\land$, that $\forall_\alpha$ distributes itself over all truth functional connectives.

Finally, specific domain axioms have to be added to impose time independency to constants, functions, and predicates.

They have the following form:

- a constant $c$ is time-independent iff
  \[ \exists x \text{Always}(c = x); \]
- an $n$-ary function $f$ is time-independent iff
  \[ \forall x_1, \ldots, x_n \exists y \text{Always}(f(x_1, \ldots, x_n) = y); \]
- an $n$-ary predicate $p$ is time-independent iff
  \[ \forall x_1, \ldots, x_n (\text{Always}(p(x_1, \ldots, x_n)) \lor \text{Always}(\neg p(x_1, \ldots, x_n))); \]

3. Embedding time granularity in the language

The main problems we have to solve to give a formal meaning to the use of different time granularities are the qualification of assertions with respect to time granularity and the definition of the links between assertions associated with a given time granularity, like "days", and the assertions associated with another granularity, like "microseconds".

Sometimes, this problem has an obvious solution that consists in using different time units—say, months and minutes—to measure time quantities in a unique dynamic model. For instance the problem of specifying a pondage power plant through a set of states and transitions requires the definition of the temporal constraints of the system. A description of the plant could include states such as empty_reservoir, full_reservoir, open_sluice_gate, closed_sluice_gate, together with the transitions between these states. A numeric value is associated with each transition, which is the time needed for its completion. We can easily state that moving from empty_reservoir to full_reservoir by applying a given input of water per second takes 2 months, whereas moving from open_sluice_gate to closed_sluice_gate, when applying the command close_sluice_gate, takes 2 minutes. All that is needed is that, syntactically, the user may attach a suitable label to temporal terms specifying the unit for them. Semantically, a possible interpreter for such a language could easily build a global state of the system bound to a time instant that is measured in the finest time unit. Simple multiplications would be needed when executing transitions measured in a coarser scale. At most, some level of nondeterminism could arise from the fact that, generally, when we say that "a reservoir is filled within 2 months" we do not mean that it is filled in exactly $2 \times 30 \times 24 \times 60 \times 60$ seconds (assuming that every month has exactly 30 days), but in an approximation of such a number whose bounds could be either explicitly stated by the user—say, 5 days—or stated a priori on the
basis of the adopted time unit—more than 1 month and less than 3 months. In this case, therefore, a model of the system using different time granularities is just an abbreviation for a model on the finest time unit.

In most granular systems, however, the treatment of different time granularities involves more difficult semantic problems. Consider, for instance, the sentence: “Every month, if an employee works, then he gets his salary”. It could be formalized, in a first-order language, by a formula such as:

$$\forall t_m, \text{emp} (\text{work}(\text{emp}, t_m) \supset \text{get_salary}(\text{emp}, t_m))$$

with an obvious meaning of the used symbols, once it is stated that the subscript “$m$” denotes the fact that $t$ is measured by the time unit of “months”.

Another requirement can be expressed by the sentence: “An employee must complete every received job within 3 days”. It is formalized by the formula:

$$\forall t_d, \text{emp, job} (\text{get_job}(\text{emp, job, t_d}) \supset \text{job_done}(\text{emp, job, t_d + 3}))$$

where the subscript “$d$” denotes that $t$ is measured by the time unit of “days”.

Assume now that the two formulae are part of the specification of the same office system. We need a common model for both formulae. As done before, we could choose the finest temporal domain, i.e., the set of (times measured by) days, as the common domain. Then, a term labeled by “$m$” would be translated into a term labelled “$d$” by multiplying its value by 30. However, clearly the statement “Every month, if an employee works, then he gets his salary” is different from “Every day, if an employee works, then he gets his salary”. In fact, working for a month means that one works for 22 days in the month, whereas getting a monthly salary means that there is one day when one gets the salary for the month. Similarly, stating that “Every day of a given month it rains” does not mean, in general, that it rains for all seconds of all days of the month.

Further difficulties arise from the so-called alignment problem of temporal domains [9]. It can be illustrated by the following examples. Consider the sentence “tomorrow I will eat”. If one interprets it in the domain of hours, its meaning is that there will be several hours, starting from the next midnight until the following one, when it will be true that I eat, no matter in which hour of the present day this sentence is claimed.

Thus, if the sentence is claimed at 1 a.m., it will be true that “I eat” in times $t$ whose distance $d$ from the current instant is such that $23 \leq d < 47$. Instead, if the same sentence is claimed at 10 p.m. of the same day, $d$ will be such that $2 \leq d < 26$. Consider now the sentence “dinner will be ready in one hour”. If it is interpreted in the domain of minutes, its meaning is that dinner will be ready in 60 minutes starting from the minute when it is claimed. Thus, if the sentence is claimed at minute, say, 10, or 55, of a given hour, always it will be true that “dinner is ready” at time $t$ whose distance $d$ from such a minute is exactly 60 minutes. Clearly, the two examples require two different semantics. We call cases of the first and second type synchronous and asynchronous, respectively. In this paper, we confine our analysis to the synchronous case.
Embedding time granularity in the basic language to support the specification of synchronous granular systems involves three main steps:

(i) replacing the unique temporal domain of the basic language with a finite set of disjoint and differently grained temporal domains whose union constitutes the temporal universe $\mathcal{F}$ of the granular specification;

(ii) qualifying formulae with respect to the temporal universe;

(iii) defining the link between the formulae associated to different temporal domains.

The temporal universe identifies the temporal domains relevant to the granular system and defines the relations between differently grained instants. It decomposes instants of coarser domains into intervals of finer domains, and abstracts intervals of finer domains into intervals or points of coarser domains.

Then, to identify the domains a given formula refers to and to specify the links between differently grained formulae, the extended language provides a contextualization and a projection operator, respectively. They allow one to build the specification of a synchronous granular system by properly connecting a set of differently grained formulae. In the simplest case, the specification consists of the logical composition of a number of temporally closed formulae referring to different temporal domains. In more complex cases, composition of differently grained formulae may require to switch from a given domain to another one. The projection operator can be used to deal with nested quantifications of differently grained temporal displacements, e.g. to model the temporal condition “Every day there exist some hours . . .”. Furthermore it can be used to specify the composition of differently grained temporal displacements, e.g., to model the temporal condition “In twenty seconds five minutes will have passed from . . .”.

Finally, we need to define some rules that, given the truth value of a formula with respect to the domain it refers to, allow us to constrain its truth value with respect to any other domain. Then, to relate the truth values of a formula, we define some default projection rules that allow us to switch it across domains. We distinguish between projections from coarser to finer domains (downward temporal projection) and projections from finer to coarser ones (upward temporal projection).

### 3.1. The notion of temporal universe

The temporal universe $\mathcal{F}$ of a specification is the union of a finite set of disjoint temporal domains, that is, $\mathcal{F} = \bigcup_{i=1, \ldots, n} T_i$. The set of domains $\{T_1, \ldots, T_n\}$ is totally ordered on the basis of the degree of fineness (coarseness) of its elements. Let $< \text{ be such a granularity relation. For each } i, \text{ with } 1 \leq i < n, T_i < T_{i+1}, \text{ and the granularity of } T_{i+1} \text{ is said to be finer than the granularity of } T_i. \text{ As an example, consider the temporal universe including years, months, weeks, and days. The domains are ordered by granularity as follows: } years < months < weeks < days. \text{ We also introduce a finer relation on the set of domains of a temporal universe, namely the disjointedness}
relation \( \trianglerighteq \). It is a partial ordering relation modeling a natural notion of inclusion between domains. It allows us to rule out domains like weeks which can overlap coarser domains like years and months. With respect to the previous example, the domains are ordered by disjointedness as follows: years \( \trianglerighteq \) months, months \( \trianglerighteq \) days, and weeks \( \trianglerighteq \) days.

Each domain is discrete with the possible exception of the finest domain(s) that may be dense. The reason is that each dense domain is already at the finest level of granularity, since it allows any degree of precision in measuring time displacements. As a consequence, for dense domains we must distinguish granularity from metric, while for discrete domains we can define granularity in terms of set cardinality and assimilate it to a natural notion of metric [9]. For simplicity, we assume that each domain is discrete.

For each ordered pair \( T_i \) and \( T_j \), with \( T_i < T_j \), a mapping is defined that maps each element \( t_i \) of \( T_i \) into an interval of contiguous elements of \( T_j \), whose width is called the conversion factor between \( T_i \) and \( T_j \) with respect to \( t_i \). In general, the value of the conversion factors of elements belonging to the same domain may be different. This dependency on time instants is introduced to deal with pairs of domains like real months and days for which a different number of instants of the finer domains (28 or 29, 30 and 31 days) corresponds to different instants of the coarser one (months). Furthermore, such a decomposition function maps contiguous instants into contiguous intervals and preserves the ordering of domains. If \( T_i \trianglerighteq T_j \), then the intervals are disjoint, e.g. in the case of the mapping from minutes to seconds, otherwise the intervals can meet at their endpoints, e.g. in the case of the mapping from months to weeks. It is worth noting that this general definition of decomposition functions allows us to deal with pairs of temporal domains in which an instant of the finer domain is astride two instants of coarser one. Finally, the union set of the intervals of \( T_j \) belonging to the range of the decomposition function is equal to \( T_j \). For each \( i, j, \) and \( k \), we also require that if \( T_i \trianglerighteq T_k \trianglerighteq T_j \) then the decomposition function from \( T_i \) to \( T_j \) is equal to the composition of the decomposition functions from \( T_i \) to \( T_k \) and from \( T_k \) to \( T_j \). For certain classes of temporal universes, we assume that for each pair of temporal domains \( T_i \) and \( T_j \) the conversion factor is constant. In such a case, conversion factors provide a relative measurement of the granularity of each ordered pair of domains \( T_i \) and \( T_j \). This assumption is useful, for instance, to deal with legal months.

In general, there are several ways to define these mappings, each one satisfying the required properties. According to the intended meaning of the mappings as decomposition functions, each element of \( T_i \) is mapped into the set of elements of \( T_j \) that compose it.

For each pair \( T_i \) and \( T_j \), with \( T_i < T_j \), we also define a coarse grain equivalent function that maps each element \( t_j \) of \( T_j \) into an interval \( I_i \) of contiguous elements of \( T_i \) such that \( t_j \) belongs to the intersection of the intervals of \( T_i \) resulting from the application of the decomposition function to the elements of \( I_j \). The uniqueness of the coarse grain equivalents can be easily deduced from the definition of the
decomposition functions. If $T_i \equiv T_j$, each interval $I_i$ is a singleton and the coarse grain equivalent function can be easily redefined as a mapping from $T_j$ on $T_i$.

### 3.2. Temporal universe formalization

In this section, the concept of temporal universe is formally characterized. First of all, we require that the set of domains is a partition of the temporal universe, which is partially ordered with respect to the disjointedness relation $\equiv$, and that each individual domain is linearly ordered. Then, we formalize the properties of conversion factors. Finally, to embed the decomposition and abstraction functions in a temporal logic setting, we define a projection relation $\rightarrow$ over the domain $\mathcal{F} \times \mathcal{F} = \bigcup_{i,j=1,...,n} T_i \times T_j$.

The requirement that the set of domains is a partition of the temporal universe is expressed by requiring that each time instant belongs to one (domains cover the temporal universe) and only one (domains are disjoint) domain, and that for each domain there exists at least one instant belonging to it (domains are not empty). Furthermore, we require that the set of domains is partially ordered with respect to $\equiv$. The linear order of each domain is obtained by requiring that each pair $(T_i, <)$, with $i = 1, \ldots, n$, is a poset and that it satisfies the backward and forward linearity axioms.

For each ordered pair of domains $T_i$ and $T_j$ and each $t_i$ in $T_i$ we also require that a conversion factor exists that expresses the numerical relationship between the granularities of $T_i$ and $T_j$ with respect to $t_i$. Let $C_F$ be the function that for each ordered pair $T_i$ and $T_j$ and each $t_i$ in $T_i$ returns the relevant conversion factor. Formally, we define a function $C_F : T_i \times \mathcal{F} \rightarrow Q$ which satisfies the following properties:

(a) conversion factors from each domain into itself are equal to 1:

$$\forall T_i, t_i(t_i \in T_i \Rightarrow C_F(t_i, T_i, T_i) = 1);$$

(b) conversion factors from coarser to (strictly) finer domains are greater than 1:

$$\forall T_i, T_j, t_i(t_i \in T_i \wedge T_i < T_j \Rightarrow C_F(t_i, T_i, T_j) > 1);$$

(c) conversion factors of symmetrical and disjoint pairs of domains are reciprocal:

$$\forall T_i, T_j, t_i(t_i \in T_i \wedge t_j \in T_j \wedge T_i \equiv T_j \wedge t_i \rightarrow t_j \Rightarrow$$

$$C_F(t_i, T_i, T_j) \cdot C_F(t_j, T_j, T_i) = 1);$$

(d) conversion factors of disjoint domains are compositional:

$$\forall T_i, T_j, T_k, t_i(t_i \in T_i \wedge t_j \in T_j \wedge T_i \equiv T_j \wedge t_i \rightarrow t_j \Rightarrow$$

$$C_F(t_i, T_i, T_j) - \sum_{t \in \{t_j : t_j \in T_j \land t_i \rightarrow t_j\}} C_F(t, T_j, T_k).$$
Let us assume \( T_k \) to be equal to \( T_j \) in (d). From (a), it follows that:

(e) the conversion factor between \( T_i \) and \( T_j \), with \( T \supseteq T_j \), with respect to \( t_i \in T_i \)

is equal to the cardinality of the set of \( t_j \in T_j \) such that \( t_i \rightarrow t_j \):

\[
\forall T_i, T_j, t_i (T_i \supseteq T_j \land t_i \in T_i \supseteq)
\]

\[
C_F(t_i, T_i, T_j) = \#(t_j : t_j \in T_j \land t_i \rightarrow t_j).
\]

Finally, we state basic and derived properties of the relation \( \rightarrow \):

- **Reflexivity.** Every time instant projects on itself:

\[
\forall t (t \rightarrow t).
\]

- **Symmetry.** If \( t_i \) downward (upward) projects on \( t_j \), then \( t_j \) upward (downward) projects on \( t_i \):

\[
\forall t_i, t_j (t_i \rightarrow t_j \supseteq t_j \rightarrow t_i).
\]

- **Downward transitivity.** If \( T_i \supseteq T_j \supseteq T_k \) and \( t_i \) of \( T_i \) projects on \( t_j \) of \( T_j \) and \( t_j \) projects on \( t_k \) of \( T_k \), then \( t_i \) projects on \( t_k \):

\[
\forall T_i, T_j, T_k, t_i, t_j, t_k (T_i \supseteq T_j \supseteq T_k \land t_i \in T_i \land t_j \in T_j \land
\]

\[
t_k \in T_k \land t_i \rightarrow t_j \land t_j \rightarrow t_k \supseteq t_i \rightarrow t_k).
\]

- **Downward/upward transitivity (case 1).** If \( T_i \supseteq T_k \supseteq T_j \) and \( t_i \) of \( T_i \) projects on \( t_j \) of \( T_j \) and \( t_j \) projects on \( t_k \) of \( T_k \), then \( t_i \) projects on \( t_k \):

\[
\forall T_i, T_j, T_k, t_i, t_j, t_k (T_i \supseteq T_k \supseteq T_j \land t_i \in T_i \land t_j \in T_j \land
\]

\[
t_k \in T_k \land t_i \rightarrow t_j \land t_j \rightarrow t_k \supseteq t_i \rightarrow t_k).
\]

- **Order preservation.** The linear order of domains is preserved by the projection relation. For each \( T_i \) and \( T_j \) we require that the projection intervals are ordered but possibly meet:

\[
\forall T_i, T_j, t_i, t_i', t_j, t_j' (t_i \in T_i \land t_j \in T_j \land t_i \in T_i \land t_j \in T_j \land
\]

\[
t_i \rightarrow t_j \land t_i' \rightarrow t_j' \land \exists \alpha (\alpha > 0 \land t_i' = t_i + \alpha) \supseteq
\]

\[
\exists \beta (\beta > 0 \land t_j' = t_j + \beta).
\]

For pairs of domains ordered by disjointedness, we require the stronger property that projection intervals are disjoint

\[
\forall T_i, T_j, t_i, t_i', t_j, t_j'
\]

\[
(T_i \supseteq T_j \land t_i \in T_i \land t_i' \in T_i \land t_j \in T_j \land t_j' \in T_j \land
\]

\[
t_i \rightarrow t_j \land t_i' \rightarrow t_j' \land \exists \alpha (\alpha > 0 \land t_i' = t_i + \alpha) \supseteq
\]

\[
\exists \beta (\beta > 0 \land t_j' = t_j + \beta).
\]
Strong order preservation and symmetry properties allow us to prove uniqueness of coarse grain equivalents for disjoint domains

$$\forall T_j, T_i, t_j, t_i, t'_i$$

$$\left( T_i \equiv T_j \land t_j \in T_j \land t_i \in T_i \land t'_i \in T_i \land t_i \rightarrow t_j \land t'_i \rightarrow t_j \implies t_i = t'_i \right)$$

Together with properties (b) and (c) of conversion factors, it allows us to generalize property (e) to the property:

$$\forall T_i, T_j, t_i(t_i \in T_i \Rightarrow [C_F(t_i, T_i, T_j)] = \#\{t_j : t_j \in T_j \land t_i \rightarrow t_j\})$$

It states that, for each pair of disjoint domains $T_i$ and $T_j$, and each $t_i \in T_i$, the $[ \ ]$ of the value of the relevant conversion factor is exactly the number of $t_j \in T_j$ such that $t_i \rightarrow t_j$.

- **Contiguity.** The projection relation maps an instant into an interval of contiguous instants on a given domain, i.e., there exist at least $[C_F(t_i, T_i, T_j)]$ contiguous instants of $T_j$ related to each instant $t_i$ of $T_i$:

$$\forall T_i, T_j, t_i(t_i \in T_i \Rightarrow$$

$$\exists t_j(t_j \in T_j \land \forall k(0 \leq k < [C_F(t_i, T_i, T_j)] \Rightarrow t_i \rightarrow t_j + k)))$$

and there exist at most $[C_F(t_i, T_i, T_j)]$ contiguous instants of $T_j$ related to $t_i$:

$$\forall T_i, T_j, t_i(t_i \in T_i \Rightarrow$$

$$\exists t'_j(t'_j \in T_j \land \forall t'_j(t'_j \rightarrow t'_j \land t'_j \in T_j \Rightarrow$$

$$\exists k(0 \leq k < [C_F(t_i, T_i, T_j)] \land t'_j = t_j + k))))$$

where $t$, $t_i$, $t_j$, and $t_k$ are quantified over the domain $T$ (if not further constrained).

For particular kinds of temporal universe, we can also require that the projection satisfies the property of homogeneity.

- **Homogeneity.** For each pair of disjoint domains of the temporal universe, the homogeneity property requires that there exists a constant conversion factor expressing the numerical relationship between their granularities:

$$\forall T_i, T_j((T_i \equiv T_j \lor T_j \equiv T_i) \Rightarrow$$

$$\exists C_{i,j} \forall t_i(t_i \in T_i \Rightarrow C_F(t_i, T_i, T_j) = C_{i,j})).$$

Clearly, such a property precludes us to deal with domains like real months.

Pairing the contiguity and the homogeneity properties we obtain that, for each pair $T_i$ and $T_j$, there exist exactly $C_{i,j}$ contiguous instants of $T_j$ related to each instant of $T_i$.

Many other relevant properties can be derived from the given ones including:
• **Totality (seriality).** The projection relation is defined for each instant of every domain of the temporal universe:

$$\forall t_i, T_j \exists t_j (t_j \in T_j \land t_i \rightarrow t_j).$$

• **Coverage.** For each instant $t_j$ and each domain $T_i$ there exist a displacement $\alpha$ and an instant $t_i$ belonging to $T_i$ such that $t_i + \alpha$ belongs to the temporal universe and projects on $t_i$, and $t_i$ projects on $t_j$:

$$\forall t_j, T_i \exists \alpha, t_i (t_j + \alpha \in T_i \land t_i \rightarrow t_j \land t_i \in T_i \land t_i \rightarrow t_j).$$

• **Upward transitivity.** If $T_k \supseteq T_j \supseteq T_i$ and $t_i$ of $T_i$ projects on $t_j$ of $T_j$ and $t_j$ projects on $t_k$ of $T_k$, then $t_i$ projects on $t_k$:

$$\forall T_i, T_j, T_k, t_i, t_j, t_k (T_k \supseteq T_j \supseteq T_i \land t_i \in T_i \land t_j \in T_j \land t_k \in T_k \land t_i \rightarrow t_j \land t_j \rightarrow t_k \supseteq t_i \rightarrow t_k).$$

• **Downward/upward transitivity (case 2).** If $T_k \supseteq T_j \supseteq T_i$ and $t_i$ of $T_i$ projects on $t_j$ of $T_j$ and $t_j$ projects on $t_k$ of $T_k$, then $t_i$ projects on $t_k$:

$$\forall T_i, T_j, T_k, t_i, t_j, t_k (T_k \supseteq T_j \supseteq T_i \land t_i \in T_i \land t_j \in T_j \land t_k \in T_k \land t_i \rightarrow t_j \land t_j \rightarrow t_k \supseteq t_i \rightarrow t_k).$$

4. **The extended logical formalism**

4.1. **The syntax of time granularity**

The alphabet of the extended language is the alphabet of the basic one plus a context sort $S_C$ denoting the set of domains into which the temporal universe is partitioned. At the same time, we introduce quantifiable context variables, context constants, and context functions, but we exclude the possibility of having predicate arguments of context sort, except for the binary predicates $\supseteq$ and $\subset$. Moreover, the extended language is provided with two other operators, namely the contextual operator $\nabla^A$, where $A$ is a context, and the projection operator $\Box$.

The set of terms $STerm$ is extended with $STerm_{S_C}$, which is defined according to the usual formation rules. For simplicity, we assume that context terms are time-independent.

The formulae of the extended language are the formulae of the basic one plus $\nabla^A \mathcal{F}$ and $\Box \mathcal{F}$, where $\mathcal{F}$ is a formula and $A$ is a context term.

The contextual operator $\nabla^A$ restricts the evaluation of $\mathcal{F}$ to time instants belonging to the context $A$ only. Moreover, $\nabla^A \mathcal{F}$ conventionally evaluates to true outside the context $A$. 
The dual operator $\Delta^A$ is defined as follows:

$$\Delta^A \mathcal{F} \overset{\text{def}}{=} \neg \nabla^A \neg \mathcal{F}.$$  

In contrast to $\nabla^A$, $\Delta^A$ conventionally evaluates to false outside the context $A$.

The projection operator $\Box$ allows us to evaluate $\mathcal{F}$ at time instants related to the current one by the projection relation. The formula $\Box \neg \mathcal{F}$ evaluates to true if $\mathcal{F}$ is true at all related instants.

The dual operator $\Diamond$ is defined as follows:

$$\Diamond \mathcal{F} \overset{\text{def}}{=} \neg \Box \neg \mathcal{F}.$$  

It evaluates to true if $\mathcal{F}$ is true in at least one related instant.

To make it possible to contextualize displacements, we also introduce the derived operator $\nabla^A_\alpha$ defined as follows:

$$\nabla^A_\alpha \mathcal{F} \overset{\text{def}}{=} \nabla^A \nabla_\alpha \mathcal{F}$$

together with the dual one $\Delta^A_\alpha$

$$\Delta^A_\alpha \mathcal{F} \overset{\text{def}}{=} \nabla^A \Delta_\alpha \mathcal{F}.$$  

They allow us to view the context term $A$ as the sort of the temporal term $\alpha$ (multisorted temporal terms). In such a way, the composition of contextual and displacement operators can be seen as new typed operators, the contextual displacements $\nabla^A_\alpha$ and $\Delta^A_\alpha$.

The following examples illustrate the main kinds of relations that can exist between different components of a layered specification.

**Example 1.** In the simplest cases, layered specifications are obtained by contextualizing formulae and composing them by means of logical connectives. For instance, the sentence:

"Men work every month and eat every day"

is specified by the formula:

$$\forall \alpha \nabla_{\alpha \text{month}} \text{work}(x_{\text{man}}) \land \forall \beta \nabla_{\beta \text{day}} \text{eat}(x_{\text{man}}).$$

**Example 2.** The projection operator is needed when displacements over different temporal domains have to be composed. For instance, the sentence:

"In twenty seconds five minutes will have passed from the occurrence of the fault"

is specified by the formula:

$$\Delta^{\text{second}}_{20} \Diamond \Delta^{\text{minute}}_{5} \text{fault}.$$  

It is possible to give a stronger interpretation of the sentence, which is expressed by the formula:

$$\Delta^{\text{second}}_{20} \Diamond \Delta^{\text{minute}}_{5} \text{fault} \land \forall x(0 \leq x < 20 \Rightarrow \neg \Delta^{\text{second}}_{x} \Diamond \Delta^{\text{minute}}_{5} \text{fault}).$$
Contextual and projection operators are also paired to specify nested quantifications. Some typical situations, together with their formalization are given by the following examples.

**Example 3.** The sentence:

“There exist some days during which the plant works every hour”

is specified by the formula:

$$\exists \alpha \Delta^\text{day}_\alpha \Box \bigtriangledown^\text{hour} \text{work(plant)}.$$

**Example 4.** The sentence:

“There exist some days during which the plant remains inactive for several hours”

is specified by the formula:

$$\exists \alpha \Delta^\text{day}_\alpha \Diamond \bigtriangleup^\text{hour} \text{inactive(plant)}.$$

**Example 5.** The sentence:

“Every day there exist some hours during which the plant is in production”

is specified by the formula:

$$\forall \alpha \bigtriangledown^\text{day}_\alpha \Diamond \Delta^\text{hour} \text{in\_production(plant)}.$$

**Example 6.** The sentence:

“The plant is monitored by the remote system each minute of every hour”

is specified by the formula:

$$\forall \alpha \bigtriangledown^\text{hour}_\alpha \Box \bigtriangleup^\text{minute} \text{monitor(remote\_system, plant)}.$$

### 4.2. The semantics of time granularity

The semantics of the language extended with time granularity is based on a concept of generalized temporal structure that still allows us to derive the notions of state and valuation function.

Formally, a **generalized temporal structure** $S$ is a triplet:

$$S = (\mathcal{D}, \mathcal{F}, \mathcal{I})$$

where

- $\mathcal{D}$ is a family of non-empty sets called the **domain of interpretation**: $\mathcal{D} = \{D_s : s \in St\}$. It extends the domain of interpretation of the basic language by replacing its metric domain with $n$ metric domains $\mathcal{D}_1, \ldots, \mathcal{D}_n$ and by adding the set of domains of the temporal universe $\{T_1, \ldots, T_n\}$ over which context terms are interpreted.
Embedding time granularity in a logical specification language

- $T$ is the temporal universe over which are defined a projection relation $\rightarrow$ and $n$ partial functions $+$ (as many temporal domains as there are);
- $\mathcal{I}$ is the interpretation function that assigns a value to variables $x$, constants $c$, functions $f$, and predicates $p$. It is the extension of the interpretation function of the basic language to the temporal universe.

On the basis of $\mathcal{I}$, we give a value to each term and formula of the language at a time instant $i$ of $T$.

The interpretation rules are the same of the basic language augmented with the following ones:

\[ \mathcal{I}_i(\forall^A \mathcal{F}) = true \iff i \in \mathcal{I}_i(A) \text{ then } \mathcal{I}_i(\mathcal{F}) = true; \]
\[ \mathcal{I}_i(\forall_a \mathcal{F}) = true \iff i + \mathcal{I}_i(\alpha) \in T (\text{then } \mathcal{I}_{i+\mathcal{I}_i(\alpha)}(\mathcal{F}) = true); \]
\[ \mathcal{I}_i(\Box \mathcal{F}) = true \iff \mathcal{I}_j(\mathcal{F}) = true \text{ for each } j \text{ such that } i \to j. \]

From duality it follows that:

\[ \mathcal{I}_i(\Delta^A \mathcal{F}) = true \iff i \in \mathcal{I}_i(A) \text{ and } \mathcal{I}_i(\mathcal{F}) = true; \]
\[ \mathcal{I}_i(\Diamond \mathcal{F}) = true \iff \mathcal{I}_j(\mathcal{F}) = true \text{ for at least one } j \text{ such that } i \to j. \]

Furthermore, it is easy to see that:

\[ \mathcal{I}_i(\forall^A_a \mathcal{F}) = true \iff (i \in \mathcal{I}_i(A) \text{ and } i + \mathcal{I}_i(\alpha) \in \mathcal{I}_i(A) \text{)) then } \mathcal{I}_{i+\mathcal{I}_i(\alpha)}(\mathcal{F}) = true. \]

Let us now redefine the notions of temporal satisfiability and temporal validity of formulae with respect to a generalized temporal structure. A formula $\mathcal{F}$ is said to be locally temporally satisfiable with respect to a temporal domain $T_i$ of a temporal structure $S$ if and only if $\mathcal{F}$ evaluates to true in at least one instant of $T_i$. A formula $\mathcal{F}$ is said to be locally temporally valid with respect to a temporal domain $T_i$ of a temporal structure $S$ if and only if $\mathcal{F}$ evaluates to true in every instant of $T_i$. A formula $\mathcal{F}$ is said to be locally temporally invariant if and only if it is locally temporally unsatisfiable or locally temporally valid.

On the basis of the concepts of local temporal satisfiability and validity, we define the notions of satisfiability and validity of formulae. A formula $\mathcal{F}$ is said to be satisfiable if and only if there exists a temporal structure with respect to which it is locally temporally satisfiable. A formula $\mathcal{F}$ is said to be valid if and only if it is locally temporally valid with respect to each temporal domain of every temporal structure.

4.3. Time granularity axiomatization

The fundamental properties of the contextual and the projection operators are given by the following axioms:

Ax11. $\forall^A(\mathcal{F} \supset \mathcal{G}) \supset (\forall^A \mathcal{F} \supset \forall^A \mathcal{G})$ (normality of $\forall^A$)
Ax12. \( \forall x \Delta^A \mathcal{F} \supset \Delta^A \forall x \mathcal{F} \)
if \( x \) is not in \( A \) (Barcan's formula for \( \Delta^A \))

Ax13. \( \forall x \Delta^A \mathcal{F} \supset \Delta^A \forall x \mathcal{F} \)
if \( x \) is not in \( A \) (Barcan's formula for \( \Delta^A \))

Ax14. \( \Delta^A \mathcal{F} \supset \mathcal{F} \)
("necessity" for \( \Delta^A \))

Ax15. \( \nabla^A \nabla^A \mathcal{F} \equiv \nabla^A \mathcal{F} \)
(idempotency of \( \nabla^A \))

Ax16. \( \nabla^A \nabla_\alpha \mathcal{F} = \nabla_\alpha \nabla^A \mathcal{F} \)
(commutativity of \( \nabla^A \) and \( \nabla_\alpha \))

Ax17. \( \square (\mathcal{F} \supset \mathcal{G}) \supset (\square \mathcal{F} \supset \square \mathcal{G}) \)
(normality of \( \square \))

where \( x \) is a variable, \( \alpha \) is a temporal term, \( A \) is a temporal sort and \( \mathcal{F} \) and \( \mathcal{G} \) are formulae, and by the inference rules:

IR2. \( \vdash \mathcal{F} \supset \vdash \nabla^A \mathcal{F} \)
(necessitation rule for \( \nabla^A \))

IR3. \( \vdash \mathcal{F} \supset \vdash \square \mathcal{F} \)
(necessitation rule for \( \square \))

First of all, it is worth noting that neither the contextual operator nor the projection operator distribute themselves over all truth functional connectives.

Axioms Ax12 and Ax13 state that the interpretation domain does not change under temporal contextualization, i.e. it is context-independent. Again, to deal with visibility and invisibility of objects in the different contexts, these axioms should be weakened.

Finally, axiom Ax15 provides us with a reduction rule for contextual operators.

As in the case of the basic language, let us report now a number of interesting theorems.

First of all, given the definition of \( \Delta^A \), it is immediate to prove that:

\[ \Delta^A \mathcal{F} = \neg \nabla^A \neg \mathcal{F} \]

and its corollaries:

\[ \nabla^A \neg \mathcal{F} = \neg \Delta^A \mathcal{F} , \]
\[ \neg \nabla^A \mathcal{F} = \Delta^A \neg \mathcal{F} . \]

Such theorems, together with the usual substitution rule of equivalents, allows us to replace \( \nabla^A \) with \( \neg \Delta^A \neg \), and vice versa, in any formula.

Axiom Ax11, together with inference rule IR2, allows us to deduce the distributivity of \( \nabla^A \) with respect to \( \land \) and then, by duality, the distributivity of \( \Delta^A \) with respect to \( \lor \).


Then, from Ax11, IR2, the distributivity of $\Delta^A$ with respect to $\vee$, and the duality of $\forall_n$ and $\Delta_n$, it follows that:

$$\forall^A \mathcal{F} \supset (\Delta^A \mathcal{F} = \Delta^\forall \mathcal{F})$$.

From the duality of $\forall^A$ and $\Delta^A$ and axiom Ax14, it follows that:

$$\mathcal{F} \supset \forall^A \mathcal{F}$$

and then:

$$\Delta^A \mathcal{F} \supset \forall^A \mathcal{F}$$.

This last theorem, together with the distributivity of $\Delta^A$ with respect to $\vee$ and the duality of $\forall^A$ and $\Delta^A$, allows us to derive that:

$$\Delta^A \mathcal{F} = (\Delta^\forall \wedge \forall^A \mathcal{F})$$.

From this theorem, it is easy to prove the distributivity of $\Delta^A$ with respect to $\wedge$ and then, by duality, the distributivity of $\forall^A$ with respect to $\vee$. Moreover, together with axiom Ax14 and the distributivity of $\Delta^A$ with respect to $\wedge$, it allows us to deduce that:

$$\Delta^A (\mathcal{F} \wedge \mathcal{G}) = \Delta^A \mathcal{F} \wedge \mathcal{G}$$,

together with the dual one:

$$\forall^A (\mathcal{F} \lor \mathcal{G}) = \forall^A \mathcal{F} \lor \mathcal{G}$$.

These formulae can be generalized to conjunction and disjunction of formulae.

Together with axiom Ax15, it also allows us to obtain the reduction rule:

$$\forall^A \Delta^A \mathcal{F} \equiv \forall^A \mathcal{F}$$.

The formula expressing the idempotency of the dual operator $\Delta^A$:

$$\Delta^A \Delta^A \mathcal{F} = \Delta^A \mathcal{F}$$

and the dual reduction rule

$$\Delta^A \forall^A \mathcal{F} \equiv \Delta^A \mathcal{F}$$

can be easily derived by duality.

In a similar way, the commutativity of $\Delta^A$ and $\Delta_n$ can be derived from axiom Ax16 by duality.

Finally, from axiom Ax17, it is possible to derive the distributivity of $\Box$ with respect to $\wedge$ using inference rule IR3 and then, by duality, the distributivity of $\Diamond$ with respect to $\vee$. It is also easy to show that from axiom Ax17 it follows that:

$$\Box \forall x \mathcal{F} \supset \forall x \Box \mathcal{F}$$.

Beside the fundamental logical properties of contextual and projection operators, we can axiomatize the properties of the temporal universe (temporal universe partition, properties of conversion factors, properties of the projection relation).

Let us report here the properties of the projection relation:

**Ax18.** $\Box \mathcal{F} \supset \mathcal{F}$  
(reflexivity)
Ax19. \( \mathcal{F} \supseteq \Box \Diamond \mathcal{F} \)  
(symmetry)

Ax20. \( \forall A, B, C \left( (A \supseteq B \supseteq C \land \nabla^A \Box \nabla^C \mathcal{F} \supset \nabla^A \Box \nabla^B \Box \nabla^C \mathcal{F} \right) \)  
(downward transitivity)

Ax21. \( \forall A, B, C \left( (A \supseteq C \supseteq B \land \nabla^A \Box \nabla^C \mathcal{F} \supset \nabla^A \Box \nabla^B \Box \nabla^C \mathcal{F} \right) \)  
(downward/upward transitivity (case 1))

Ax22. \( \forall A, B \left( \exists x(x > 0 \land \Delta^A \Diamond \Delta^B \mathcal{F} \supset \nabla^A \Box \nabla^B \exists y(y \geq 0 \land \nabla_y \mathcal{F}) \right) \)  
(weak order preserving)

Ax22'. \( \forall A, B \left( (A \equiv B \land \exists x(x > 0 \land \Delta^A \Diamond \Delta^B \mathcal{F}) \supset \nabla^A \Box \nabla^B \exists y(y > 0 \land \nabla_y \mathcal{F}) \right) \)  
(strong order preserving)

Ax23. \( \forall A, B \nabla^A \exists z(z = \left[ C_F(B) \right] \land (\Diamond \Delta^B \forall x(0 \leq x < z \supset \nabla_x \mathcal{F}) \equiv \exists y(0 \leq y < z \land \nabla^B \nabla_y \mathcal{F})) \)  
(contiguity)

Ax24. \( \forall A, B \exists z((A \equiv B \lor B \equiv A) \supset \nabla^A (C_F(B) = z)) \)  
(homogeneity)

where \( C_F(B) \) is a time-dependent function denoting the conversion factors.\(^6\)

Given the axiom for symmetry, it follows that:

\( \forall x \Box \mathcal{F} \supset \Box \forall x \mathcal{F} \)  
(Barcan’s formula for \( \Box \))

and then:

\( \forall x \Box \mathcal{F} \equiv \Box \forall x \mathcal{F} \).

From the given axioms it is also possible to derive the following theorems:

\( \forall B (\Box \nabla^B \mathcal{F} \supset \Diamond \Delta^B \mathcal{F}) \)  
(totality)

\( \forall A, B, C \left( (C \equiv B \equiv A \land \nabla^A \Box \nabla^C \mathcal{F} \supset \nabla^A \Box \nabla^B \Box \nabla^C \mathcal{F} \right) \)  
(upward transitivity)

\(^6\) It is worth noting that the contiguity axiom assumes the basic properties of displacements. For instance, given an interval \([x, y]\) of width \(d\), such properties allow us to conclude that the intervals \([x + d_1, x + d_1 + d]\) and \([x + d_1, y + d_1]\) coincide. Furthermore, notice that the restriction of the function \(C_F(B)\) to any domain \(A\) becomes time-independent when homogeneity is assumed.
Finally, we introduce upward and downward projection rules. For each pair of domains \( T_i \) and \( T_j \), with \( T_i \) coarser than \( T_j \), the downward projection rule states that if a property \( P \) holds at a time instant \( t_i \) of \( T_i \), then there exists at least one time instant \( t_j \) of \( T_j \), belonging to its decomposition, such that \( P \) holds at \( t_j \). For each pair of domains \( T_i \) and \( T_j \), with \( T_i \) finer than \( T_j \), the upward projection rule states that if a property \( P \) holds at each time \( t_i \) of \( T_i \) such that \( t_i + t_j \), then \( P \) holds at time \( t_j \).

Formally, downward projection is defined by the following axiom:

\[
\forall A, B, C ((C \equiv A \equiv B \land \forall^A \square \forall^C \square) \supset \forall^A \square \forall^B \square \forall^C \square)
\]

(downward/upward transitivity (case 2))

\[
\forall A (\square \exists x \Delta_x \Diamond \Delta^A \Diamond \square)
\]

(coverage)

This allows us to conclude that the axioms defining downward and upward projection are interdeducible.

The downward projection rule provides the weakest semantics that can be attached to an assertion in a domain finer than the original one, provided that such an assertion is not wholistic.\(^7\) Most often it is too weak so that user qualifications are needed. In general, it is possible to provide domain-specific categorizations of assertions according to their behaviour under downward temporal projection. Such categorizations allow us to introduce and characterize primitive ontological concepts as event, property, fact, and process in terms of their temporal projection.\(^8\) It allows us to distinguish assertions that hold at one and only one instant of the finer domain (punctual), assertions that hold at each instant \( t_i \) of the finer domain such that \( t_i \to t_j \) (continuous and pervasive), assertions that hold over a scattered sequence of intervals of the finer domain whose elements \( t_j \) all satisfy the condition \( t_i \to t_j \) (bounded sequence), and so on [25].

---

\(^7\) Wholistic assertions relate to the structure of the interval over which they hold as a whole, and they do not hold over any proper subinterval of it. Such assertions cannot be projected across domains.

\(^8\) These kinds of categorization present some similarities with the classification of temporal propositions given in [36] and with the characterization of assertions proposed in [35] to deal with time and space granularities.
In the rest of the section, we give a brief survey of the soundness and completeness proofs for time granularity.

The soundness of the logical language for time granularity is proved by checking that each axiom is a valid formula and that each inference rule deduces a true formula from a true formula. Axioms and inference rules of first-order predicate calculus, included into time granularity, are assumed sound (see for instance [1]). Proofs of soundness for axioms and inference rules related to time granularity can be found in [26] and here they are only sketched. The proof of the soundness of each axiom referring to a temporal operator can be easily deduced from the semantic definition of the language. The proof of soundness of inference rules related to displacement, contextual, and projection operators is easily derived by the notion of validity. The soundness of the axioms expressing properties of temporal universe is proved negating each property in the temporal structure and then checking that no interpretation satisfies the corresponding axiom.

About the proof of completeness for time granularity, we sketch out a schema quite similar to [34] that allows us to get a relative completeness result according to the well-known Gödel incompleteness results on arithmetic axiomatization theories. Such a proof provides a correspondence between time granularity (TG) and a pure quantification theory (FO), and derives TG completeness by means of FO completeness. Firstly, a translation function $\tau$ is defined in such a way that for each valid formula $\mathcal{F}$ in TG there exists a valid formula $\mathcal{F}^*$ in FO. Further, the deduction of $\mathcal{F}^*$ can be obtained by means of FO completeness. Finally, the completeness of TG is obtained putting into correspondence the deduction of $\mathcal{F}^*$ in FO, and a deduction of $\mathcal{F}$ in TG, by means of a translation function $\tau^{-1}$.

5. Examples of layered specifications

In this section, we show how to use the extended language to specify a monitoring system and a high voltage station.

Example 7 (Monitoring system specification). Let $S$ be a monitoring system composed of a monitor $M$ and a remote system $R$. $R$ must send a message to $M$ every hour. If in a given hour the message does not arrive, then the next hour $M$ activates a control procedure sending a control message to $R$. If $R$ gives back an answer within 5 seconds and sends the expected message to $M$ no later than 5 seconds after the answer, then the verification is successful and the system comes back to its normal state. Otherwise, $M$ declares $R$ idle 10 seconds after the control message. There is no restoration from the idle condition.

The formal specification of $S$ uses a temporal universe composed of two domains, hours and seconds. The normal monitoring activity refers to the domain of hours, while the fast control procedure refers to the domain of seconds. It consists of the logical conjunction of three different components $C_1$, $C_2$, and $C_3$. 
• C1: The control procedure starts if a message has not arrived within the given time boundary, and \( R \) has never been declared idle:
\[
\forall \alpha \forall_{\text{hour}} (\text{control} = (\forall_{-1} \square \forall_{\text{seconds}} \lnot \text{message} \land \lnot \text{SomPast}(\text{idle}))).
\]

• C2: The idle declaration:
\[
\forall \alpha \forall_{\text{seconds}} (\forall_{10} \text{idle} = (\text{control} \land (\text{Lasts}(\lnot \text{answer}, -6) \lor \\
\exists \beta (1 \leq \beta \leq 5 \land \forall_{\beta} (\text{answer} \land \text{Lasts}(\lnot \text{message})))))).
\]

• C3: An answer from \( R \) can only occur within 5 seconds from a control message:
\[
\forall \alpha \forall_{\text{seconds}} (\text{answer} \Rightarrow \exists \beta (-5 \leq \beta \leq -1 \land \forall_{\beta} \text{control})).
\]

**Example 8 (High voltage station specification).** This example is a little part of a case study provided by the Centro Ricerche in Automatica (CRA) of the Ente Nazionale per l’Energia Elettrica (ENEL). It regards the specification of a supervisor that automates the activities of a high voltage (HV) station, devoted to the end-user distribution of the energy generated by power plants. Each station is composed of bays, connecting the generation units and the distribution line. A bay consists of circuit breakers and insulators. They are both switches, but an expensive circuit breaker can interrupt current in a very short time (50 milliseconds or even less), while a cheap insulator is not able to interrupt a flowing current and has switching time of a few seconds.

Let us consider a simple HV station consisting of two bars \( b_1 \) and \( b_2 \) connected to different power units, a distribution line \( l \) and two bays, \( pb \) (parallel bay) and \( lb \) (line bay). The parallel bay shorts circuit between the two bars \( b_1 \) and \( b_2 \); it is composed of two insulators, \( ip_1 \) and \( ip_2 \), and one circuit breaker \( cb_p \). It is in the state \( \text{closed} \) if all its switches are closed, it is \( \text{open} \) otherwise. The line bay connects the distribution line either with the first or the second bar. It is composed of three insulators \( ilb_1, ilb_2, \) and \( ill \) and one circuit breaker \( cb_p \). It is in the state \( \text{closed on } b_1 \) if \( ilb_1, cbl, \) and \( ill \) are closed, while it is in the state \( \text{closed on } b_2 \) if \( ilb_2, cbl, \) and \( ill \) are closed.

We report here the specification of the change from \( b_1 \) to \( b_2 \) of the bar connected to the line. The supervisor must close the parallel bay \( pb \) first, this action taking 10 seconds, then it closes the insulator \( ilb_2 \) and opens the insulator \( ilb_1 \) in 5 seconds. Lastly, it opens the parallel bay, taking another 10 seconds.

For the formal specification, we identify for every action the time scale where it can be considered as an instantaneous event. The change of the bar takes about 30 seconds, the opening and the closing of the parallel bay 10 seconds, the switching of the insulators 5 seconds, the switching of the circuit breakers 50 milliseconds.

The predicates \( \text{change}_{-\text{bar from } b_1 \text{ to } b_2}, \text{closed}_{-\text{pb}}, \text{open}_{-\text{pb}}, \text{close}_{-\text{ilb}1}, \text{close}_{-\text{ilb}2}, \text{open}_{-\text{ilb}1}, \) etc., denote the corresponding commands sent to the various devices by the supervisor. The existential projection operator \( \Diamond \) is used to connect formulae on different domains.
The change of bar is described by the formula below, specifying the sequence of actions taken by the supervisor.

$$\forall \alpha \forall \alpha^{10}\text{sec}(\text{change\_bar\_from\_b1\_to\_b2} \Rightarrow$$

$$\Diamond (\Delta^{10}\text{sec}\text{close\_pb} \land \Delta^{5}\text{sec}\text{close\_ilb2} \land$$

$$\Delta^{5}\text{sec}\Diamond \Delta^{10}\text{sec}\text{open\_par\_bay})).$$

The effect of closing the parallel bay is specified by the following formula:

$$\forall \alpha \forall \alpha^{10}\text{sec}(\text{close\_pb} \Rightarrow \Diamond (\Delta^{5}\text{sec}\text{close\_ip1} \land$$

$$\Delta^{5}\text{sec}\text{close\_ip2} \land \Delta^{5}\text{sec}\Diamond \Delta^{50\text{mili}}\text{close\_ch})).$$

The opening of the parallel bay is symmetrical to its closing, so we do not show it here for the sake of brevity.

6. Conclusions

When building specifications for time-dependent systems—whether plant control systems, office systems, or whatever—it may happen that different components of such systems have quite different dynamic behaviours, bound to different time scales. Present formal languages impose the use of a unique time scale, which can make formal specifications of such systems quite cumbersome and unnatural.

In this paper, we presented an axiomatic approach to deal with different time granularities in real-time, granular specifications. The first step has been the definition of a temporal logic language, suitable to explicitly deal with time and therefore to cope with hard real-time systems. It is a revised version of the logical specification language TRIO. We endowed it with a new definition of syntax and semantics based on a unique basic temporal operator, and with a sound system of axioms. Then, we extended this language with operators to deal with time granularity.

We first introduced the concept of temporal universe in a more general way than in [5, 7] to fit a larger variety of structures of practical interest. For instance, it is now possible to deal with real months (and not only 30-day legal months), leap years, and weeks (which do not fit exactly in a month or in a year). Then, we defined syntax and semantics of suitable operators augmenting the temporal logic language to deal with granular specifications. The semantics of the extended language is based on the concept of generalized temporal structure. It allows us to define the notions of local temporal satisfiability and validity that make it possible to generalize the basic concepts of temporal invariance, satisfiability, and validity. Finally, we formalized the properties of the contextual and projection operators by a sound axiomatic system. Significant results have been derived from the axioms, including the equivalence between the formulae for upward and downward temporal projections.
This paper does not exhaust the time granularity problem. First, a formal proof of completeness for the logical system is needed if we want to use it to derive proofs of properties of specifications. Secondly, the proposed language is just a kernel for an effective specification language. In fact, it lacks abstraction and modularization mechanisms that make it suitable to deal with the complexity and the details of real-life cases. Such mechanisms do exist in more structured languages that have been defined on the basis of TRIO, namely TRIO+ [29] and TRIO* [7]. These are provided by exploiting object-oriented techniques and combining them with TRIO features. Clearly, the possibility of dealing with different time granularities should be extended to these languages. Since we do believe in the orthogonality of our approach, we do not expect major difficulties in such a job.

The proposed language together with other related ones, equipped with presently prototype tools, are the current result of an on-going research that aims at the construction of a complete specification environment for real-time systems whose kernel is the language TRIO. A detailed description of the main features of the environment is given in [28].

It is widely accepted belief that the effectiveness of a specification language is strongly increased by the availability of a rich and integrated environment of tools. Such tools should allow not only the editing and the managing of specification documents, but even their execution, with the purpose of early prototyping and verification [15, 16, 21]. No execution algorithms have been yet developed for the proposed language. Currently, we can only have some partial executions, based on the algorithm originally developed for TRIO [28], based on the proof method of semantic tableaux [1, 2, 37, 40] that allows us to prove the finite satisifiability of a formula by constructing a finite model for it. Such an algorithm can be used to verify the consistency of specifications and to perform both simulation and verification of histories with respect to a given specification. It also allows the specifier to prove any property of the system that can be derived from its specification by verifying if the conjunction of the specification and the negation of the property is unsatisfiable.

With respect to the extended language, parts of a specification referring to the same temporal domain can be easily translated in the basic language and executed using the existing method. So we can have various formulae to be interpreted separately on different temporal domains. This is a step to prove the consistency of a granular specification, but it is not enough to say that specifications on different time scales are really executable. New algorithms are thus needed for the proposed language to actually execute specifications involving different temporal domains.

Acknowledgements

We thank Dino Mandrioli, Angelo Morzenti, and Elena Ratto for the useful discussions and the anonymous referees for their helpful and perceptive suggestions.
for improvements of the first draft of the paper. We would also like to thank Johan van Benthem and Alberto Policriti for their valuable comments on an earlier version of the paper.

References

Embedding time granularity in a logical specification language


