# A Compositional Approach on Modal Specifications for Timed Systems*

Nathalie Bertrand[1], Axel Legay[1],
Sophie Pinchinat[2], and Jean-Baptiste Raclet[3]

[1] INRIA Rennes, France
[2] IRISA & Université Rennes 1, France
[3] INRIA Rhône-Alpes, France

**Abstract.** On the one hand, modal specifications are classic, convenient, and expressive mathematical objects to represent interfaces of component-based systems. On the other hand, time is a crucial aspect of systems for practical applications, e.g. in the area of embedded systems. And yet, only few results exist on the design of timed component-based systems. In this paper, we propose a timed extension of modal specifications, together with fundamental operations (conjunction, product, and quotient) that enable to reason in a compositional way about timed system. The specifications are given as modal event-clock automata, where clock resets are easy to handle. We develop an entire theory that promotes efficient incremental design techniques.

## 1 Introduction

Nowadays, systems are tremendously big and complex, resulting from the assembling of several components. These many components are in general designed by teams, working independently but with a common agreement on what the interface of each component should be. As a consequence, mathematical foundations that allow to reason at the abstract level of interfaces, in order to infer properties of the global implementation, and to design or to advisedly (re)use components is a very active research area, known as *compositional reasoning* [16]. In a logical interpretation, interfaces are specifications and components that implement an interface are understood as models. Aiming at practical applications as the final goal, the software engineering point of view naturally leads to the following requirements for a good theory of interfaces.

1. *Satisfiability/Consistency and Satisfaction.* It should be decidable whether a specification admits a model, and whether a given component implements a given interface. Moreover, for the synthesis of components to be effective, satisfiable interfaces should always have finitely presentable models.

* This work was funded by the European project COMBEST, IST-STREP 215543.

2. *Refinement and shared refinement. Refinement* of specifications [20, 23] expresses inclusion of sets of models, and therefore allows to compare interfaces. Related to this implication-like concept, the intersection, or *greatest lower bound*, is an optimal interface refining two given interfaces.

3. *Compositionality of the abstraction.* The interface theory should also provide a combination operator on interfaces, reflecting the standard composition of models by, *e.g.* parallel product.

4. *Quotient.* Last but not least, a quotienting operation, dual to composition is crucial to perform incremental design. Intuitively, the quotient enables to describe a part of a global specification assuming another part is already realized by some component. Together with the composition $\otimes$ the quotient operator $\oslash$ enjoys the following fundamental property at the component level:
$$\mathcal{C}_2 \models \mathcal{S} \oslash \mathcal{S}_1 \Leftrightarrow \forall \mathcal{C}_1 \; [\mathcal{C}_1 \models \mathcal{S}_1 \Rightarrow \mathcal{C}_1 \otimes \mathcal{C}_2 \models \mathcal{S}] \qquad (\star)$$
where $\mathcal{S}, \mathcal{S}_i$ are interfaces, $\mathcal{C}, \mathcal{C}_i$ components, and $\models$ is the satisfaction relation.

Building good interface theories is the subject of intensive studies which have led to theories based on models such as interface automata [12, 14], modal automata or specifications [19, 22–24, 5], and their respective timed extension [13, 9]. Modal specifications are deterministic automata equipped with transitions of the following two types: *may* and *must*. The components that implement such interfaces are deterministic labeled transition systems; an alternative language-based semantics can therefore be considered, as presented in [22, 23]. Informally, a must transition is available in every component that implements the modal specification, while a may transition needs not to be. Modal specifications are interpreted as logical specifications matching the conjunctive nu-calculus fragment of the mu-calculus [15]. As a corollary, but also proved directly in [22], satisfaction and consistency of modal specifications are decidable, and the finite model property holds. Refinement between modal specifications coincides with a standard notion of alternating simulation. Since components can be seen as specifications where all transitions are typed must (all possible implementation choices have been made), satisfaction is also expressed via alternating simulation. Shared refinement is effectively computed via a product-like construction. Combination of modal specifications, handling synchronization products *à la* Arnold and Nivat [6], and the dual quotient combinators can be efficiently handled in this setting [23].

Recently, a timed extension of the theory of modal specifications has been introduced [9], motivated by the fact that time can be a crucial parameter in practice, *e.g.* in embedded–system applications. In this piece of work, components are timed automata as defined in [1], and naturally, an effective and expressive region-based semantics allows to combine modalities and timing constraints.

In this paper, we build on this preliminary paper and develop a complete compositional approach for modal specifications of timed systems. This framework favors methodologies for an incremental design process: Assume a global system implementing specification $\mathcal{S}$ has to be synthesized, and assume a component implements interface $\mathcal{S}_1$. Computing $\mathcal{S} \oslash \mathcal{S}_1$ and synthesizing a model of $\mathcal{S} \oslash \mathcal{S}_1$ yields a component that, in $\otimes$-combination with the component for $\mathcal{S}_1$, will yield a model for the global interface $\mathcal{S}$, thanks to property ($\star$). As a consequence, low complexity algorithms are needed for computing product and quotient, as well as for the satisfiability decision procedure.

The synchronous product of timed objects requires a tight control on clocks [1], and so should its dual quotient. Actually, developing the theory in the general framework where components can reset their clocks in an arbitrary manner is a difficult question. Indeed, computing the resets of clocks of a product or of a quotient depends on how the control of clocks is distributed among the components. This information has to be provided *a priori*, which requires an extra formalism. We therefore restrict the presentation to the class of components definable by event-clock automata [2]: in these timed automata, resets are fully determined by the actions. Interfaces whose models are event-clock automata are called *modal event-clock specifications* (MECS).

Inheriting from the region-based semantics of timed modal specifications [9], we study the satisfiability as well as the consistency problems for MECS. Satisfiability is PSPACE-complete, hence no harder than traditional decision problems in the class of timed automata. Refinement serves as a theoretical basis to develop the product and the quotient of MECS. We propose two equivalent characterizations of these operations. Not surprisingly according to the semantics, inefficient EXPTIME constructions via the region graphs of the MECS (seen as untimed specifications) are provided. More interestingly, we present alternative direct and efficient PTIME constructions.

The rest of the paper is organized as follows. In Section 2, we introduce the timed modal specification setting, with preliminaries on untimed modal specifications and the definition of modal event-clock specifications. Section 3 focuses on MECS and exposes effective techniques to compute the binary operations of greatest lower bound, product, and quotient. In Section 4, we compare our framework with the existing literature. Section 5 concludes the paper. A long version of the current paper, including proofs is available as a research report [8].

## 2 Timed modal specifications

In this section we recall the framework of *modal specifications* originaly defined in [18] twenty years ago (see [5] for a survey), and its timed extension, recently proposed in [9]: We discuss the semantic, the preorder refinement and the satisfiability problem for untimed and timed modal specifications.

## 2.1 Preliminaries on untimed specifications

A modal specification is an automaton equipped with two types of transitions: *must*-transitions, that are required and *may*-transitions, that are allowed. We fix $\Sigma$ a finite set of actions.

**Definition 2.1 (Modal specification).** *A modal specification (*MS*) is a tuple $\mathcal{R} = (P^{\perp\!\!\!\perp}, \lambda^0, \Delta^m, \Delta^M)$ where $P^{\perp\!\!\!\perp} = P \cup \perp\!\!\!\perp$ is a finite set of states with $\perp\!\!\!\perp \cap P = \emptyset$, $\lambda^0 \in P^{\perp\!\!\!\perp}$ is the unique initial state, and $\Delta^M \subseteq \Delta^m \subseteq P \times \Sigma \times P^{\perp\!\!\!\perp}$. $\Delta^M$ and $\Delta^m$ correspond respectively to* must*-transitions and* may*-transitions. We additionally assume that $\Delta^m$ is* deterministic *(hence so is $\Delta^M$) and* complete*, that is, for every state $p \in P$ and every action $a \in \Sigma$, there is exactly one state $\lambda \in P^{\perp\!\!\!\perp}$ such that $(p, a, \lambda) \in \Delta^m$.*

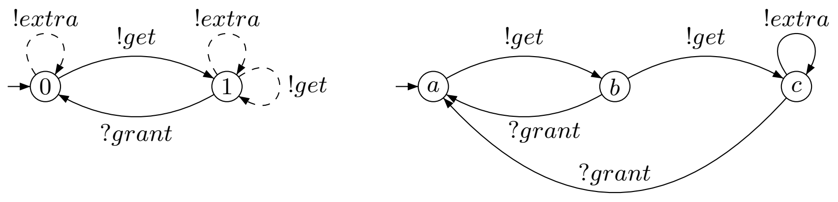
We use $p$ (resp. $\lambda$) as typical element of $P$ (resp. $P^{\perp\!\!\!\perp}$). Note that completeness is not a restriction since from any incomplete specification, one can derive a complete one by adding may-transitions to a possibly new state $\perp \in \perp\!\!\!\perp$. Intuitively, in state $p \in P$ a-may transition to some state $\lambda \in \perp\!\!\!\perp$ labelled by action $a$ means that action $a$ is forbidden in $p$. This interpretation will become clearer when we define the set of models of a modal specification.

The condition $\Delta^M \subseteq \Delta^m$ naturally imposes that every required transition is also allowed; it guarantees the *local consistency* of the modal constraints. The set of states $\perp\!\!\!\perp$ denotes the "bad states" which carry local inconsistency. Elements of $\perp\!\!\!\perp$ are *sink states* with no outgoing transition since both $\Delta^M$ and $\Delta^m$ are subsets of $P \times \Sigma \times P^{\perp\!\!\!\perp}$. *Global inconsistency* can be derived as follows: we let $\mathcal{I}$ be the set of *inconsistent* states that must lead (that is via a sequence of *must*-transitions) to a local inconsistency; states in $P^{\perp\!\!\!\perp} \setminus \mathcal{I}$ are *consistent.* Formally $\mathcal{I} = \{\lambda_0 \mid \exists n \geq 0,\ \exists \lambda_1 \cdots \lambda_n \in P^{\perp\!\!\!\perp}\ \exists a_1 \cdots a_n \in \Sigma \text{ s.t. } \lambda_n \in \perp\!\!\!\perp \text{ and } (\lambda_i, a_{i+1}, \lambda_{i+1}) \in \Delta^M\}$. Notice that in particular $\perp\!\!\!\perp \subseteq \mathcal{I}$. We say that the modal specification $\mathcal{R}$ is *consistent* whenever its initial state is consistent, i.e. $\lambda^0 \notin \mathcal{I}$; otherwise $\mathcal{R}$ is *inconsistent.*

In the following, we write or draw $p \xrightarrow{a} \lambda$ (resp. $p \overset{a}{\dashrightarrow} \lambda$) to mean $(p, a, \lambda) \in \Delta^M$ (resp. $(p, a, \lambda) \in \Delta^m \setminus \Delta^M$); in other words, solid arrows denote *required* transition, whereas dashed arrow represent *allowed* but not required transitions.

*Example 2.2.* Consider a client for a given resource available in a system. The alphabet of actions includes: *get* when the resource is requested; *grant* in case of access to the resource; and, *extra* which occurs when a privileged access with extended time is requested.

In order to simplify the figures, states in $\perp\!\!\!\perp$ are not represented (except if they are necessary) and transitions of the form $q \overset{a}{\dashrightarrow} \perp$ are not depicted. Action names may be preceded by some "!" or "?" when the occurrence of the actions respectively stems from the designed component or by its environment.



(a) Client $Cl$ (b) Automaton $\mathcal{M}$

**Fig. 1.** The modal specification $Cl$ accepts the automaton $\mathcal{M}$

The modal specification $Cl$ for the client in Fig. 1(a) specifies that a *get* request may be sent again. Moreover every *get* request *must* be granted. Additionally the client may request extended time at any moment.

Models of MS are *deterministic automata*[4], with possibly infinitely many states, which we shortly call *automata* in the sequel. An automaton is a structure of the form $\mathcal{M} = (M, m^0, \Delta)$ where $M$ is a (possibly infinite) set of states, $m^0 \in M$ is a unique initial state, and $\Delta \subseteq M \times \Sigma \rightarrow M$ is a *partial* transition function. The model relation $\models$ defined below is a particular case of alternating simulation [4] between the model and the consistent part, if any, of the specification.

**Definition 2.3 (Model Relation).** *Let $\mathcal{R} = (P^{\perp}, \lambda^0, \Delta^m, \Delta^M)$ be a* MS. *An automaton $\mathcal{M} = (M, m^0, \Delta)$ is a model of $\mathcal{R}$, written $\mathcal{M} \models \mathcal{R}$, if there exists a binary relation $\rho \subseteq M \times (P \backslash \mathcal{I})$ such that $(m^0, \lambda^0) \in \rho$, and for all $(m, p) \in \rho$, the following hold:* (1) *for every $(p, a, \lambda) \in \Delta^M$ there is a transition $(m, a, m') \in \Delta$ with $(m', \lambda) \in \rho$, and* (2) *for every $(m, a, m') \in \Delta$ there is a transition $(p, a, \lambda) \in \Delta^m$ with $(m', \lambda) \in \rho$.*

We denote by $\mathsf{Mod}(\mathcal{R})$, the set of models of an MS $\mathcal{R} = (P^{\perp}, \lambda^0, \Delta^m, \Delta^M)$.

Remark in Definition 2.3 that inconsistent states of the specification cannot appear in the relation $\rho$. Consequently, a transition of the form $(p, a, \lambda) \in \Delta^m$ where $\lambda \in \mathcal{I}$ is inconsistent interprets as: in any model, no $a$-transition from a state in relation with $p$ is allowed. Moreover, for $\lambda^0 \in \mathcal{I}$ no $\rho$ can exist and actually we have:

**Lemma 2.4.** *Let $\mathcal{R}$ be a* MS. $\mathsf{Mod}(\mathcal{R}) \neq \emptyset$ *if, and only if, $\mathcal{R}$ is consistent.*

*Example 2.5.* The automaton $\mathcal{M}$ in Fig. 1(b) is a model of the MS $Cl$ in Fig. 1(a) as the binary relation $\rho = \{(a, 0), (b, 1), (c, 1)\}$ witnesses.

The semantic preorder between MS relies on an extension of Definition 2.3.

[4] also called *deterministic labeled transition systems.*

**Definition 2.6 (Modal Refinement Preorder).** *Given two* MS, $\mathcal{R}_1 = (P_1^{\perp\!\!\perp}, \lambda_1^0, \Delta_1^m, \Delta_1^M)$ *and* $\mathcal{R}_2 = (P_2^{\perp\!\!\perp}, \lambda_2^0, \Delta_2^m, \Delta_2^M)$*,* $\mathcal{R}_1$ *is* a refinement *of* $\mathcal{R}_2$*, written* $\mathcal{R}_1 \preceq \mathcal{R}_2$*, whenever there exists a binary relation* $\rho \subseteq (\mathcal{I}_1 \times \mathcal{I}_2) \cup (P_1^{\perp\!\!\perp} \times (P_2 \setminus \mathcal{I}_2))$ *such that* $(\lambda_1^0, \lambda_2^0) \in \rho$*, and for all* $(\lambda_1, \lambda_2) \in \rho \cap ((P_1 \setminus \mathcal{I}_1) \times (P_2 \setminus \mathcal{I}_2))$*:*

*(1) for every* $(\lambda_2, a, \lambda_2') \in \Delta_2^M$ *there exists* $(\lambda_1, a, \lambda_1') \in \Delta_1^M$ *with* $(\lambda_1', \lambda_2') \in \rho$
*(2) for every* $(\lambda_1, a, \lambda_1') \in \Delta_1^m$ *there exists* $(\lambda_2, a, \lambda_2') \in \Delta_2^m$ *with* $(\lambda_1', \lambda_2') \in \rho$*.*

Definition 2.6 requires some explanations. First, by definition of the domain of $\rho$, an inconsistent state of $\mathcal{R}_2$ can only be refined as an inconsistent state in $\mathcal{R}_1$ whereas a consistent state in $\mathcal{R}_2$ can either be linked to a consistent or inconsistent state in $\mathcal{R}_1$. Moreover, for pairs of consistent states, Condition (1) ensures that all required transition in $\mathcal{R}_2$ are also required in $\mathcal{R}_1$, and Condition (2) guarantees that each possible transition in $\mathcal{R}_1$ is also allowed in $\mathcal{R}_2$.

Under our assumption that MS are deterministic, we can show that the preorder $\preceq$ between MS matches the model inclusion preorder. We establish an intermediate result that exploits the embedding of automata into modal specifications.

**Definition 2.7 (Embedding in** MS**).** *An automaton* $\mathcal{M} = (M, m^0, \Delta)$ *can be interpreted as a modal specification* $\mathcal{M}^* = (M \cup \{\perp_*\}, m^0, \Delta_*^m, \Delta_*^M)$ *where* $\Delta = \Delta_*^M \subseteq \Delta_*^m$*, and* $(m, a, \perp_*) \in \Delta_*^m \setminus \Delta_*^M$ *when* $\Delta(m, a)$ *is undefined in* $\mathcal{M}$*.*

**Lemma 2.8.** *Given an automaton* $\mathcal{M}$ *and a* MS $\mathcal{R}$*,* $\mathcal{M} \models \mathcal{R}$ *iff* $\mathcal{M}^* \preceq \mathcal{R}$*.*

**Proposition 2.9.** *Let* $\mathcal{R}_1$ *and* $\mathcal{R}_2$ *be two* MS*, then:*

$$\mathcal{R}_1 \preceq \mathcal{R}_2 \text{ if, and only if, } \mathsf{Mod}(\mathcal{R}_1) \subseteq \mathsf{Mod}(\mathcal{R}_2).$$

Note that the determinism of modal specifications is crucial for the Proposition 2.9. In the nondeterministic case, modal refinement is not complete [20], that is $\mathsf{Mod}(\mathcal{R}_1) \subseteq \mathsf{Mod}(\mathcal{R}_2)$ does not imply $\mathcal{R}_1 \preceq \mathcal{R}_2$ in general.

As a consequence of Definition 2.6, inconsistent MS refine any MS, and consistent MS can only refine consistent MS. In the following, we write $\mathcal{R}_1 \equiv \mathcal{R}_2$, and say that $\mathcal{R}_1$ and $\mathcal{R}_2$ are *equivalent*, whenever $\mathcal{R}_1 \preceq \mathcal{R}_2$ and $\mathcal{R}_2 \preceq \mathcal{R}_1$. Remark that by merging all states of $\mathcal{I}$, every MS is equivalent to a MS where the set of inconsistent states is a singleton.

## 2.2 Modal event-clock specifications

Let $\mathcal{X}$ be a finite set of *clocks* and let $\mathbb{R}_{\geq 0}$ denote the set of non-negative reals. A *clock valuation* over $\mathcal{X}$ is a mapping $\nu : \mathcal{X} \to \mathbb{R}_{\geq 0}$. The set of clock valuations over $\mathcal{X}$ is denoted $\mathcal{V}$; in particular, $\overline{0} \in \mathcal{V}$ is the clock valuation such that $\overline{0}(x) = 0$

for all $x \in \mathcal{X}$. Given $\nu \in \mathcal{V}$ and $t \in \mathbb{R}_{\geq 0}$, we let $(\nu + t) \in \mathcal{V}$ be the clock-valuation obtained by letting $t$ time units elapse after $\nu$, formally, $(\nu + t)(x) = \nu(x) + t$ for every $x \in \mathcal{X}$.

A *guard* over $\mathcal{X}$ is a finite conjunction of expressions of the form $x \sim c$ where $x \in \mathcal{X}$, $c \in \mathbb{N}$ is a *constant*, and $\sim \in \{<, \leq, =, \geq, >\}$. We then denote by $\xi[\mathcal{X}]$ the set of all guards over $\mathcal{X}$. For some fixed $N \in \mathbb{N}$, $\xi_N[\mathcal{X}]$ represents the set of guards involving only constants equal to or smaller than $N$. The *satisfaction relation* $\models \subseteq (\mathcal{V} \times \xi[\mathcal{X}])$ between clock valuations and guards is defined in a natural way and we write $\nu \models g$ whenever $\nu$ satisfies $g$. In the following, we will often abuse notation and write $g$ to denote the guard $g$ as well as the set of valuations which satisfy $g$.

*Event-clock* automata [2], form a subclass of timed automata where clock resets are not arbitrary: each action $a$ comes with a clock $x_a$ which is reset exactly when action $a$ occurs. We consider event-clock automata with possibly infinitely many locations.

**Definition 2.10 (Event-clock automata).** *An* event-clock automaton *(*ECA*) over $\Sigma$ is a tuple $\mathcal{C} = (C, c^0, \delta)$ where $C$ is a set of states, $c^0 \in C$ is the initial state, and $\delta \subseteq C \times \xi_N[\mathcal{X}_\Sigma] \times \Sigma \times C$ is the transition relation (for some $N \in \mathbb{N}$). The pair $(\Sigma, N)$ is the* signature *of $\mathcal{C}$.*

The semantics of an ECA is similar to the one of a timed automaton [1], except that the set of clocks that are reset by a transition is determined by the action of that transition: while firing a transition labeled by $a$, precisely clock $x_a$ is reset. Event-clock automata do form a strict subclass of timed automata, but they enjoy nice properties: they are closed under union and intersection, and more interestingly they can be determinized (as opposed to the class of arbitrary timed automata). The determinizability of event-clock automata comes from the way clocks are reset and this property significantly eases the definition of binary operators (such as lower bound, product and quotient) on modal variants of event-clock automata.

For a fixed signature $(\Sigma, N)$, a *region* is an equivalence class $\theta$ of clock-valuations that satisfy the same guards in $\xi_N[\mathcal{X}_\Sigma]$. We denote by $\Theta_N$, or simply $\Theta$, the set of all regions. Given a region $\theta \in \Theta$, we write $\mathsf{Succ}(\theta)$ for the union of all regions that can be obtained from $\theta$ by letting time elapse: $\mathsf{Succ}(\theta) = \{\theta'' \mid \exists \nu'' \in \theta''\ \exists \nu \in \theta\ \exists t \in \mathbb{R}_{\geq 0} \text{ s.t. } \nu'' = \nu + t\}$.

**Definition 2.11 (Region automaton [1]).** *The* region automaton *associated to an* ECA *$\mathcal{C} = (C, c^0, \delta)$ is the automaton $R(\mathcal{C}) = (C \times \Theta, (c^0, \overline{0}), \Delta)$ over the alphabet $\Theta \times \Sigma$, where the set $\Delta$ of transitions is defined as follows: for each $c, c' \in C$, $\theta, \theta', \theta'' \in \Theta$, and $a \in \Sigma$, $((c, \theta), \theta'', a, (c', \theta')) \in \Delta$ whenever there exists $(c, g, a, c') \in \delta$ with $\theta'' \subseteq \mathsf{Succ}(\theta) \cap g$, and $\theta' = \theta''[x_a = 0]$ is the region obtained from $\theta''$ by resetting clock $x_a$.*

Note that the region automata we consider extend the ones introduced in [1] since their transition labels keep track of the intermediate region where the action is fired. As a consequence, any automaton over the alphabet $\Theta \times \Sigma$ uniquely defines an ECA whose signature is of the form $(\Sigma, N_\Theta)$, with $N_\Theta$ determined by the set of regions $\Theta$. We denote by $T$ the natural injection of region automata into ECA; this mapping enables us to distinguish between the two interpretations of the same syntactic object: $R(\mathcal{C})$ is an automaton whereas $T(R(\mathcal{C}))$ is an ECA.

**Definition 2.12 (Modal event-clock specification).** *A* modal event-clock specification (MECS) *over the finite alphabet $\Sigma$ is a tuple $\mathcal{S} = (Q^{\perp\!\!\!\perp}, \lambda^0, \delta^m, \delta^M)$ where*

- *$Q^{\perp\!\!\!\perp} := Q \cup \perp\!\!\!\perp$ is a finite set of* locations*, with $\perp\!\!\!\perp \cap Q = \emptyset$, and the initial state is $\lambda^0 \in Q^{\perp\!\!\!\perp}$.*
- *$\delta^M \subseteq \delta^m \subseteq Q \times \xi[\mathcal{X}_\Sigma] \times \Sigma \times Q^{\perp\!\!\!\perp}$ are finite sets of respectively* must- *and* may-*transitions. Given a may-transition $(q, g, a, \lambda) \in \delta^m$, $q$ is the source state, $\lambda$ is the destination state, $g \in \xi[\mathcal{X}_\Sigma]$ is the guard that specifies the valuations for which the transition can be taken, $a \in \Sigma$ is the action labeling the transition – recall that the only clock that is then reset is $x_a$.*

*Moreover we require that $\delta^m$ is* deterministic *(hence, so is $\delta^M$) and* complete*: for any state $q \in Q$, any action $a \in \Sigma$, and any clock valuation $\nu \in \mathcal{V}$, there is exactly one transition $(q, g, a, \lambda) \in \delta^m$ such that $\nu \models g$.*

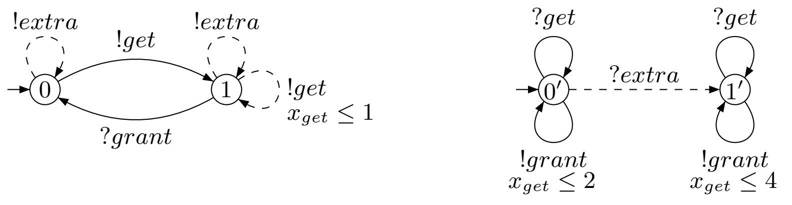
*Example 2.13.* As an example of a MECS, we consider in Fig. 2(a) a timed variant of the client *Cl* introduced earlier. The clock corresponding to the action *get* is $x_{get}$.

In this example again, for simplification purposes, transitions of the form $q \overset{g,a}{\dashrightarrow} \perp$ are not depicted. As MECS are complete, these transitions can easily be recovered by taking $g = \neg(\bigvee_i g_i)$ where the $g_i$'s are the guards appearing in the transitions of the form $q \overset{g_i,a}{\dashrightarrow} \lambda$ or $q \xrightarrow{g_i,a} \lambda$. When the guard of a transition is not indicated, it is implicitly true.

The MECS *Cl* for the client in Fig. 2(a) specifies that a *get* request may be sent again at most one time unit after the last request.

In the sequel, we generalize the graphical convention already used for untimed objects by writing $q \overset{g,a}{\dashrightarrow} \lambda'$ whenever $(q, g, a, \lambda') \in (\delta^m \setminus \delta^M)$ and $q \xrightarrow{g,a} \lambda'$ whenever $(q, g, a, \lambda') \in \delta^M$.

Remark that a natural untimed object associated to a MECS $\mathcal{S}$ is its region modal automaton, obtained by generalizing Definition 2.11 from event-clock automata to their modal extension. More precisely, $R(\mathcal{S})$ reflects the modalities of $\mathcal{S} = (Q^{\perp\!\!\!\perp}, \lambda^0, \delta^m, \delta^M)$ as done in [9], the initial state is $(\lambda^0, \overline{0})$ and the set of locally inconsistent states in $R(\mathcal{S})$ is $\perp\!\!\!\perp_{\mathcal{S}} \times \Theta$. A MECS $\mathcal{S}$ is said to be *inconsistent*



(a) Client *Cl* (b) Access controller *Acc*

**Fig. 2.** Client *Cl* and access controller *Acc*

if $R(\mathcal{S})$ is inconsistent; otherwise, it is *consistent*. Given a modal event-clock specification $\mathcal{S}$ over signature $(\Sigma, N)$, $R(\mathcal{S})$ is a modal specification over the extended alphabet $\Sigma \times \Theta_N$; similarly, given an event-clock automaton $\mathcal{C}$, $R(\mathcal{C})$ is an automaton over alphabet $\Sigma \times \Theta_N$. Having this in mind, the model relation in the timed case is inherited from the one in the untimed case via the region construction:

**Definition 2.14 (Model relation).** *Let $\mathcal{S}$ be a* MECS. *An event-clock automaton $\mathcal{C}$ is a* model *of $\mathcal{S}$, written $\mathcal{C} \models \mathcal{S}$, if $R(\mathcal{C}) \models R(\mathcal{S})$.*

The set of models of a MECS $\mathcal{S}$, is defined by $\mathsf{Mod}(\mathcal{S}) := \{\mathcal{C} \mid \mathcal{C} \models \mathcal{S}\}$.

Observing that given a MECS $\mathcal{S}$, $R(T(R(\mathcal{S})))$ and $R(\mathcal{S})$ are isomorphic, we obtain the following:

**Lemma 2.15.** *Let $\mathcal{S}$ be a* MECS. *Then,* $\mathsf{Mod}(T(R(\mathcal{S}))) = \mathsf{Mod}(\mathcal{S})$.

In the spirit of Def.2.14 for the model relation, the *modal refinement preorder* between MECS also relies on a region-based construction:

**Definition 2.16 (Modal refinement preorder).** *Given two* MECS *$\mathcal{S}_1$ and $\mathcal{S}_2$, $\mathcal{S}_1$* refines *$\mathcal{S}_2$, written $\mathcal{S}_1 \preceq \mathcal{S}_2$, whenever $R(\mathcal{S}_1) \preceq R(\mathcal{S}_2)$.*

As a corollary of the analogous results in the untimed setting on MS, it is decidable whether a MECS refines another one. Moreover, refinement and inclusion of models match:

**Corollary 2.17.** *Let $\mathcal{S}$, $\mathcal{S}_1$ and $\mathcal{S}_2$ be* MECS. *Then,*

– $\mathsf{Mod}(\mathcal{S}) \neq \emptyset$ *if, and only if $\mathcal{S}$ is consistent;*
– $\mathcal{S}_1 \preceq \mathcal{S}_2$ *if, and only if* $\mathsf{Mod}(\mathcal{S}_1) \subseteq \mathsf{Mod}(\mathcal{S}_2)$.

*About consistency* We recall that a specification is *consistent* if, and only if, it admits a model. According to Lemma 2.4, checking whether an untimed specification is consistent amounts to checking that the set of states $\perp\!\!\!\perp$ cannot be reached from its initial state by a sequence of must-transitions. The consistency problem is thus NLOGSPACE-complete for modal specifications and PSPACE-complete in the timed case.

## 3 Operations on specifications

In this section, we introduce operations on modal event-clock specifications, which enable compositional reasoning. More precisely, we define the greatest lower bound, the product, and the quotient over MECS. For each of these operations, we establish important theoretical properties.

### 3.1 Greatest lower bound of MECS

We study the concept of *greatest lower bound*, which corresponds to the conjunction of two modal specifications and equivalently to their best shared refinement. We first recall the definition of the greatest lower bound in the untimed case. Let $\mathcal{R}_1 = (P_1^{\perp\!\!\!\perp}, \lambda_1^0, \Delta_1^m, \Delta_1^M)$ and $\mathcal{R}_2 = (P_2^{\perp\!\!\!\perp}, \lambda_2^0, \Delta_2^m, \Delta_2^M)$ be two MS. The *greatest lower bound* of $\mathcal{R}_1$ and $\mathcal{R}_2$ is $\mathcal{R}_1 \wedge \mathcal{R}_2 = (P^{\perp\!\!\!\perp}, (\lambda_1^0, \lambda_2^0), \Delta_\wedge^m, \Delta_\wedge^M)$ with $P := P_1 \times P_2$, $\perp\!\!\!\perp := (\perp\!\!\!\perp_1 \times P_2^{\perp\!\!\!\perp}) \cup (P_1^{\perp\!\!\!\perp} \times \perp\!\!\!\perp_2)$, and whose transition relations are derived from the following rules:

$$\frac{\lambda_1 \overset{a}{\dashrightarrow} \lambda_1' \text{ and } \lambda_2 \overset{a}{\dashrightarrow} \lambda_2'}{(\lambda_1, \lambda_2) \overset{a}{\dashrightarrow} (\lambda_1', \lambda_2')}\ (Glb1) \qquad \frac{\lambda_1 \xrightarrow{a} \lambda_1' \text{ and } \lambda_2 \overset{a}{\dashrightarrow} \lambda_2'}{(\lambda_1, \lambda_2) \xrightarrow{a} (\lambda_1', \lambda_2')}\ (Glb2)$$

$$\frac{\lambda_1 \overset{a}{\dashrightarrow} \lambda_1' \text{ and } \lambda_2 \xrightarrow{a} \lambda_2'}{(\lambda_1, \lambda_2) \xrightarrow{a} (\lambda_1', \lambda_2')}\ (Glb3) \qquad \frac{\lambda_1 \xrightarrow{a} \lambda_1' \text{ and } \lambda_2 \xrightarrow{a} \lambda_2'}{(\lambda_1, \lambda_2) \xrightarrow{a} (\lambda_1', \lambda_2')}\ (Glb4)$$

Remark in particular, that if in a state $\lambda = (\lambda_1, \lambda_2)$, we have the contradictory requirements that $a$ is required ($\lambda_1 \xrightarrow{a} \lambda_1' \in P_1$) and $a$ should not happen ($\lambda_2 \overset{a}{\dashrightarrow} \lambda_2' \in \perp\!\!\!\perp_2$), then $\lambda$ is inconsistent. This is indeed guaranteed by the definition of $\mathcal{R}_1 \wedge \mathcal{R}_2$ which imposes $P_1 \times \perp\!\!\!\perp_2 \subseteq \perp\!\!\!\perp$.

*Greatest lower bound of* MECS. The notion of greatest lower bound easily extends to MECS. Let $\mathcal{S}_1, \mathcal{S}_2$ be two MECS. The modalities for the transitions in $\mathcal{S}_1 \wedge \mathcal{S}_2$ are derived from those induced in the untimed case (Rules (*Glb*1) to (*Glb*4)), and the labels of the transitions are obtained by intersecting the guards for common actions. As an example, Rule (*Glb*1) becomes (*tGlb*1) as follows.

$$\frac{\lambda_1 \overset{g_1,a}{\dashrightarrow} \lambda_1' \text{ and } \lambda_2 \overset{g_2,a}{\dashrightarrow} \lambda_2'}{(\lambda_1, \lambda_2) \overset{g_1 \wedge g_2,a}{\dashrightarrow} (\lambda_1', \lambda_2')}\ (tGlb1)$$

Thanks to Lemma 2.15, the set of models of a MECS $\mathcal{S}$ matches the set of models of its region version $T(R(\mathcal{S}))$. The following proposition characterizes the greatest lower bound of two MECS via the region graphs.

**Proposition 3.1.** *For any two* MECS $\mathcal{S}_1$ *and* $\mathcal{S}_2$, $R(\mathcal{S}_1 \wedge \mathcal{S}_2) \equiv R(\mathcal{S}_1) \wedge R(\mathcal{S}_2)$.

In Proposition 3.1, operator $\wedge$ is overloaded: on the right hand side, it corresponds to the greatest lower bound of MS whereas on the left hand side, it corresponds to the greatest lower bound of MECS.

Computing the conjunction of two MS via rules (*Gbl*1) to (*Gbl*4) is polynomial in the size of the arguments. Due to the construction of the region graphs, starting from two MECS $\mathcal{S}_1$ and $\mathcal{S}_2$ computing $R(\mathcal{S}_1) \wedge R(\mathcal{S}_2)$ is exponential. The direct construction of the greatest lower bound by using the timed variants of (*Gbl*1) to (*Gbl*4) is polynomial and therefore worth adopting for effective methods.

Finally, according to the above, one can establish that the greatest lower bound yields the intersection of the models.

**Theorem 3.2.** *For any two* MECS $\mathcal{S}_1$ *and* $\mathcal{S}_2$, $\mathsf{Mod}(\mathcal{S}_1 \wedge \mathcal{S}_2) = \mathsf{Mod}(\mathcal{S}_1) \cap \mathsf{Mod}(\mathcal{S}_2)$.

Application of the greatest lower bound is the following: in the design of a component one gives several specifications, each of them describing a particular requirement. The greatest lower bound of these specifications enables to check the compatibility of these requirements, by deciding consistency.

## 3.2 Product of MECS

The product of MECS relates to the synchronous parallel composition of models. For MS, it generalizes the synchronized product of automata $\mathcal{M}_1 \otimes \mathcal{M}_2$ that denotes the intersection of their behaviors (languages).

We first recall the product of MS: Let $\mathcal{R}_1 = (P_1^{\perp\!\!\!\perp}, \lambda_1^0, \Delta_1^m, \Delta_1^M)$ and $\mathcal{R}_2 = (P_2^{\perp\!\!\!\perp}, \lambda_2^0, \Delta_2^m, \Delta_2^M)$ be two MS over the same alphabet $\Sigma$. The product of $\mathcal{R}_1$ and $\mathcal{R}_2$, denoted by $\mathcal{R}_1 \otimes \mathcal{R}_2$, is the MS $(P^{\perp\!\!\!\perp}, (\lambda_1^0, \lambda_2^0), \Delta_\otimes^m, \Delta_\otimes^M)$, with $P := P_1 \times P_2$, $\perp\!\!\!\perp := (\perp\!\!\!\perp_1 \times P_2^{\perp\!\!\!\perp}) \cup (P_1^{\perp\!\!\!\perp} \times \perp\!\!\!\perp_2)$, and whose transitions are derived from the following rules:

$$\frac{\lambda_1 \overset{a}{\dashrightarrow} \lambda_1' \text{ and } \lambda_2 \overset{a}{\dashrightarrow} \lambda_2'}{(\lambda_1, \lambda_2) \overset{a}{\dashrightarrow} (\lambda_1', \lambda_2')} \ (Prod1) \qquad \frac{\lambda_1 \xrightarrow{a} \lambda_1' \text{ and } \lambda_2 \overset{a}{\dashrightarrow} \lambda_2'}{(\lambda_1, \lambda_2) \overset{a}{\dashrightarrow} (\lambda_1', \lambda_2')} \ (Prod2)$$

$$\frac{\lambda_1 \overset{a}{\dashrightarrow} \lambda_1' \text{ and } \lambda_2 \xrightarrow{a} \lambda_2'}{(\lambda_1, \lambda_2) \overset{a}{\dashrightarrow} (\lambda_1', \lambda_2')} \ (Prod3) \qquad \frac{\lambda_1 \xrightarrow{a} \lambda_1' \text{ and } \lambda_2 \xrightarrow{a} \lambda_2'}{(\lambda_1, \lambda_2) \xrightarrow{a} (\lambda_1', \lambda_2')} \ (Prod4)$$

Notice that Rules (*Prod*1) to (*Prod*4) uniformly consider consistent and inconsistent states.

*Product of* MECS. The product of MECS extends the synchronized product of ECA which consists in synchronizing transitions on action names and in taking the conjunction of the guards of the combined transitions.

Let $\mathcal{S}_1, \mathcal{S}_2$ be two MECS. The modalities for the transitions in $\mathcal{S}_1 \otimes \mathcal{S}_2$ are derived from those proposed in the untimed case, and the labels of the transitions are composed of the intersection of the guards together with the common action. For example, the timed version of $(Prod1)$ becomes $(tProd1)$ as follows.

$$\frac{\lambda_1 \overset{g_1,a}{\dashrightarrow} \lambda_1' \text{ and } \lambda_2 \overset{g_2,a}{\dashrightarrow} \lambda_2'}{(\lambda_1, \lambda_2) \overset{g_1 \wedge g_2,a}{\dashrightarrow} (\lambda_1', \lambda_2')} \ (tProd1)$$

Similarly to Proposition 3.1 for the greatest lower bound, the product of MECS can be alternatively computed by building the product of the region graphs. This construction however causes an exponential blow-up whereas the direct construction is polynomial. Notice that operator $\otimes$ is overloaded to ease the presentation.

**Proposition 3.3.** *$R(\mathcal{S}_1 \otimes \mathcal{S}_2) \equiv R(\mathcal{S}_1) \otimes R(\mathcal{S}_2)$.*

In the untimed setting, it is known [23] that the product is monotonic with respect to the refinement, and that a product of models is a model of the product. Those properties extend to the timed case as stated in the following theorem.

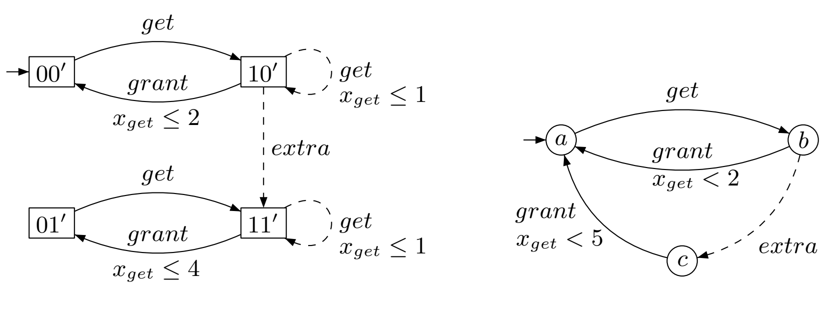
**Theorem 3.4 (Properties of the product).** *For any* MECS $\mathcal{S}_1, \mathcal{S}_1', \mathcal{S}_2, \mathcal{S}_2'$, *and any* ECA $\mathcal{C}_1, \mathcal{C}_2$,

$$\begin{array}{rcl}(\mathcal{S}_1 \preceq \mathcal{S}_2 \text{ and } \mathcal{S}_1' \preceq \mathcal{S}_2') & \Longrightarrow & \mathcal{S}_1 \otimes \mathcal{S}_1' \preceq \mathcal{S}_2 \otimes \mathcal{S}_2'; \text{ and} \\ (\mathcal{C}_1 \models \mathcal{S}_1 \text{ and } \mathcal{C}_2 \models \mathcal{S}_2) & \Longrightarrow & \mathcal{C}_1 \otimes \mathcal{C}_2 \models \mathcal{S}_1 \otimes \mathcal{S}_2.\end{array}$$

As a consequence, the product operation satisfies the property of *independent implementability*, in the sense of [12]: an implementation of a specification of the form $\mathcal{S}_1 \otimes \mathcal{S}_2$ can be obtained by composing any two independent implementations of $\mathcal{S}_1$ and $\mathcal{S}_2$ respectively.

*Example 3.5.* The MECS *Acc* in Fig. 2(b) page 9 specifies the behavior of an access controller; the access to the resource will be granted for 2 time units after the reception of a *get* request. In case of a privileged access with an extra time, this duration will be extended to 4 time units.

The product $Cl \otimes Acc$ is depicted in Fig. 3(a). In the resulting specification, *extra* can now only occur *after* a *get* request. Timing constraints on the *grant* action issued from the access controller are also propagated.



(a) The product $Cl \otimes Acc$ (b) A desired behavior $G$

**Fig. 3.** The global model $Cl \otimes Acc$ and its specified behavior $G$

## 3.3 Quotient of MECS

In this section, we define the *quotient operation*. Intuitively, the quotient describes a part of a global specification assuming another part will be realized by some component. We thus consider quotient of *specifications* which is different from the constructions studied in [17] where at least one of the operand is a system. We start by recalling the quotient operation on untimed modal specifications, then extend it to MECS.

In the untimed setting, we aim at defining an operation dual to the product of Section 3.2 in the following sense. Given two MS $\mathcal{R} = (P^{\perp\!\!\!\perp}, \lambda^0, \Delta^m, \Delta^M)$ and $\mathcal{R}_1 = (P_1^{\perp\!\!\!\perp}, \lambda_1^0, \Delta_1^m, \Delta_1^M)$, we want the *quotient* of $\mathcal{R}$ by $\mathcal{R}_1$ to be the MS written $\mathcal{R} \oslash \mathcal{R}_1$ which satisfies the following properties.

**Proposition 3.6.** *For every automaton $\mathcal{M}_2$,*

$$\mathcal{M}_2 \models \mathcal{R} \oslash \mathcal{R}_1 \iff \forall \mathcal{M}_1.\ [\mathcal{M}_1 \models \mathcal{R}_1 \Rightarrow \mathcal{M}_1 \otimes \mathcal{M}_2 \models \mathcal{R}] \tag{1}$$

*and $\mathcal{R} \oslash \mathcal{R}_1$ is the greatest such one, namely*

$$\mathcal{R}_1 \otimes \mathcal{R}_2 \preceq \mathcal{R} \iff \mathcal{R}_2 \preceq \mathcal{R} \oslash \mathcal{R}_1 \tag{2}$$

The definition of the quotient follows [23], but it is here revisited with a uniform way to handle both consistent and inconsistent states, as opposed to the original definition where so-called *pseudo-specifications* needed being considered.

Formally, the quotient of $\mathcal{R} = (P^{\perp\!\!\!\perp}, \lambda^0, \Delta^m, \Delta^M)$ by $\mathcal{R}_1 = (P_1^{\perp\!\!\!\perp}, \lambda_1^0, \Delta_1^m, \Delta_1^M)$ is the MS $\mathcal{R} \oslash \mathcal{R}_1 = (P'^{\perp\!\!\!\perp}, (\lambda^0, \lambda_1^0), \Delta_\oslash^m, \Delta_\oslash^M)$, with $P' \subseteq (P \times P_1) \cup \{\top\}$, where $\top$ is fresh element, and the set $\perp\!\!\!\perp'$ of locally inconsistent states of $\mathcal{R} \oslash \mathcal{R}_1$ contains at least an element $\perp'$ but also other elements: the rules below describe these elements as well as the set of transitions. Notation $\lambda \overset{a}{\dashrightarrow} \mathcal{I}$ means that the $a$-may-transition from $\lambda$ leads to an inconsistent state in $\mathcal{I}$. We also use notations

$\lambda \overset{a}{\dashrightarrow} P \setminus \mathcal{I}$, $\lambda \overset{a}{\longrightarrow} \mathcal{I}$, and $\lambda \overset{a}{\longrightarrow} P \setminus \mathcal{I}$ with the expected meaning, and $\lambda \overset{a}{\dashrightarrow}$ whenever there is no $a$-must-transition from state $\lambda$.

$$\frac{\lambda \in \mathcal{I} \text{ and } \lambda_1 \notin \mathcal{I}_1}{(\lambda, \lambda_1) \in \perp\!\!\!\perp'}\ {\scriptstyle (I \wedge \neg I1)} \qquad \frac{\lambda \in \mathcal{I} \text{ and } \lambda_1 \in \mathcal{I}_1}{(\lambda, \lambda_1) \overset{a}{\dashrightarrow} \top}\ {\scriptstyle (I \wedge I1)}$$

$$\frac{\mathcal{I} \not\ni \lambda \overset{a}{\dashrightarrow} \text{ and } \lambda_1 \in \mathcal{I}_1}{(\lambda, \lambda_1) \in \perp\!\!\!\perp'}\ {\scriptstyle (\neg Imust \wedge I1)} \qquad \frac{\mathcal{I} \not\ni \lambda \overset{a}{\dashrightarrow} \text{ and } \lambda_1 \in \mathcal{I}_1}{(\lambda, \lambda_1) \overset{a}{\dashrightarrow} \top}\ {\scriptstyle (\neg Imay \wedge I1)}$$

$$\frac{}{\top \overset{a}{\dashrightarrow} \top}\ {\scriptstyle (top)}$$

Assume now that both $\lambda$ and $\lambda_1$ are consistent, *i.e.*, $\lambda \notin \mathcal{I}$ and $\lambda_1 \notin \mathcal{I}_1$:

$$\frac{\lambda \overset{a}{\dashrightarrow} \mathcal{I} \text{ and } (\lambda_1 \overset{a}{\dashrightarrow} P_1 \setminus \mathcal{I}_1 \text{ or } \lambda_1 \overset{a}{\longrightarrow} P_1 \setminus \mathcal{I}_1)}{(\lambda, \lambda_1) \overset{a}{\dashrightarrow} \perp'}\ {\scriptstyle (maynot)}$$

$$\frac{\lambda \overset{a}{\dashrightarrow} \text{ and } \lambda_1 \overset{a}{\dashrightarrow} \mathcal{I}_1}{(\lambda, \lambda_1) \overset{a}{\dashrightarrow} \top}\ {\scriptstyle (may1)}$$

$$\frac{\lambda \overset{a}{\dashrightarrow} \lambda' \notin \mathcal{I} \text{ and } (\lambda_1 \overset{a}{\dashrightarrow} \lambda_1' \notin \mathcal{I}_1 \text{ or } \lambda_1 \overset{a}{\longrightarrow} \lambda_1' \notin \mathcal{I}_1)}{(\lambda, \lambda_1) \overset{a}{\dashrightarrow} (\lambda', \lambda_1')}\ {\scriptstyle (may2)}$$

$$\frac{\lambda \overset{a}{\longrightarrow} \lambda' \text{ and } \lambda_1 \overset{a}{\dashrightarrow} \lambda_1'}{(\lambda, \lambda_1) \in \perp\!\!\!\perp'}\ {\scriptstyle (inconsistency)}$$

$$\frac{\lambda \overset{a}{\longrightarrow} \lambda' \text{ and } \lambda_1 \overset{a}{\longrightarrow} \lambda_1'}{(\lambda, \lambda_1) \overset{a}{\longrightarrow} (\lambda', \lambda_1')}\ {\scriptstyle (must)}$$

We now give intuitive explanations for the rules above in particular with respect to the first requirement of Proposition 3.6. To do so, let $\mathcal{R}^\lambda$ be the MS informally defined as the sub-specification of $\mathcal{R}$ with initial state $\lambda$. When explaining a rule involving transitions outgoing $\lambda$ in $\mathcal{R}$ and $\lambda_1$ in $\mathcal{R}_1$ we will thus speak about models in $\mathcal{R}^\lambda$, $\mathcal{R}_1^{\lambda_1}$ and $\mathcal{R}^\lambda \oslash \mathcal{R}_1^{\lambda_1}$. $\mathcal{R}^\lambda$ and $\mathcal{R}_1^{\lambda_1}$ are just introduced in order to be able to regard local models of $\mathcal{R}$ and $\mathcal{R}_1$ from states $\lambda$ and $\lambda_1$. When, say $\lambda \in \mathcal{I}$, we have $\mathsf{Mod}(\mathcal{R}^\lambda) = \emptyset$.

Rule $(I \wedge \neg I_1)$ ensures that since there are no models for $\mathcal{R}^\lambda$ and there are models for $\mathcal{R}_1^{\lambda_1}$, there should not be models of $\mathcal{R}^\lambda \oslash \mathcal{R}_1^{\lambda_1}$, otherwise we would not have the right to left implication of Equation (1) in Proposition 3.6.

For Rules $(\neg Imust \wedge I1)$ and $(\neg Imay \wedge I1)$ (together with Rule $(top)$), since $\mathsf{Mod}(\mathcal{R}_1^{\lambda_1}) = \emptyset$, the right hand side of Equation (1) is trivially satisfied. Therefore in $(\neg Imust \wedge I1)$, the $a$-transition required from $\lambda$ cannot be guaranteed; hence the quotient is not consistent. On the other hand for Rule $(\neg Imay \wedge I1)$, since nothing particular is required from $\lambda$ for the $a$-transition, nothing either needs being required for models of the quotient; to guarantee Equation (2) of Proposition 3.6 (which states the maximality of the quotient) we set the quotient to be universal, *i.e.* it accepts every model.

Rule $(I \wedge I_1)$ together with Rule (*top*), is the case where both $\mathsf{Mod}(\mathcal{R}^\lambda) = \emptyset$ and $\mathsf{Mod}(\mathcal{R}_1^{\lambda_1}) = \emptyset$. In this case, the universal MS that accepts every model can be in the quotient, and this is what is chosen in order to get the greatest such MS, as required by Equation (2).

We now come to the set of rules where both $\lambda$ and $\lambda_1$ are consistent ($\lambda \notin \mathcal{I}$ and $\lambda_1 \notin \mathcal{I}_1$), which by Lemma 2.4 amounts to say that $\mathsf{Mod}(\mathcal{R}^\lambda) \neq \emptyset$ and $\mathsf{Mod}(\mathcal{R}_1^{\lambda_1}) \neq \emptyset$.

In Rule (*may*1), $a$ is not possible from $\lambda_1$, and $a$ is not mandatory from $\lambda$, it can therefore safely be authorized in the quotient. Rule (*maynot*) deals with the case where $a$ is forbidden in $\mathcal{R}^\lambda$, but is authorized or even mandatory in $\mathcal{R}_1^{\lambda_1}$: it should be forbidden in the quotient.

Rule (*may*2) is very straightforward, as models of the quotient may have an $a$-transition irrespectively of what is required in $\mathcal{R}_1^{\lambda_1}$.

Finally, Rules (*inconsistency*) and (*must*) consider the case where we have must transitions in $\mathcal{R}^\lambda$. Rule (*inconsistency*) corresponds to the inability of guaranteeing the $a$-transition required in $\mathcal{R}^\lambda$ since it may not exist in some models of $\mathcal{R}^{\lambda_1}$. Hence only an inconsistent MS can be considered so that Equation (1) holds. Rule (*must*) is the simple case of must requirements; notice that we implicitly have $\lambda_1' \notin \mathcal{I}_1$, since by assumption $\lambda_1 \notin \mathcal{I}_1$.

One can easily verify that the conditions of the premises of Rules from $(I \wedge \neg I_1)$ to (*must*) are exclusive, hence the quotient construction yields a deterministic object. Also, the quotient MS is complete.

*Quotient of* MECS. The quotient of a MECS $\mathcal{S} = (Q^{\perp\!\!\!\perp}, \lambda^0, \delta^m, \delta^M)$ by a MECS $\mathcal{S}_1 = (Q_1^{\perp\!\!\!\perp}, \lambda_1^0, \delta_1^m, \delta_1^M)$ is the MECS $\mathcal{S} \oslash \mathcal{S}_1 = (Q'^{\perp\!\!\!\perp}, (\lambda^0, \lambda_1^0), \delta_\oslash^m, \delta_\oslash^M)$, where $Q' \subseteq (Q \times Q_1) \cup \{\top\}$ and where the set of locally inconsistent states and the transition modalities follow the rules $(I \wedge \neg I_1)$ to (*must*) of the untimed case; the guard of a transition is the conjunction of the local guards of $\mathcal{S}$ and $\mathcal{S}_1$. For example, the untimed rule (*must*) becomes (*tmust*) as follows.

$$\frac{\lambda \xrightarrow{g,a} \lambda' \text{ and } \lambda_1 \xrightarrow{g_1,a} \lambda_1'}{(\lambda, \lambda_1) \xrightarrow{g \wedge g_1, a} (\lambda', \lambda_1')} \ (tmust)$$

Besides, the rule (*ttop*) is the following:

$$\frac{}{\top \overset{\mathbf{true}, a}{\dashrightarrow} \top} \ (ttop)$$

This quotient operation for MECS can be used on ECA as the class of deterministic ECA can be embedded into the one of MECS; it suffices to type with must every existing transitions in the ECA, and to complete it by adding transitions typed by may to a state in $\perp\!\!\!\perp$. Assuming determinacy of event-clock automata is not restrictive, since they are known to be determinizable [2]. Observe that then the quotient of two event-clock automata is not an event-clock automaton since *e.g.* Rule $(\neg Imay \land I_1)$ introduces a may transition to the top state.

Finally, the quotienting operation yields a deterministic and complete specification. Hence:

**Lemma 3.7.** *Modal event-clock specifications are closed under quotient.*

As for the product operation, the quotient operation in the timed and untimed settings relate via the region construction (for the extended alphabet) as follows.

**Proposition 3.8.** $R(\mathcal{S} \oslash \mathcal{S}_1) \equiv R(\mathcal{S}) \oslash R(\mathcal{S}_1)$.

The correctness of the quotient construction is stated by the following.

**Theorem 3.9 (Properties of the quotient).** *For any* MECS $\mathcal{S}, \mathcal{S}_1, \mathcal{S}_2$, *and any* ECA $\mathcal{C}_2$,

$$\mathcal{C}_2 \models \mathcal{S} \oslash \mathcal{S}_1 \iff \forall \mathcal{C}_1.\ [\mathcal{C}_1 \models \mathcal{S}_1 \Rightarrow \mathcal{C}_1 \otimes \mathcal{C}_2 \models \mathcal{S}];\ \textit{and} \tag{3}$$

$$\mathcal{S}_1 \otimes \mathcal{S}_2 \preceq \mathcal{S} \iff \mathcal{S}_2 \preceq \mathcal{S} \oslash \mathcal{S}_1. \tag{4}$$

From a practical point of view, the quotient operation enables incremental design: consider a desired global specification $\mathcal{S}$, and the specification $\mathcal{S}_1$ of a preexisting component. By computing $\mathcal{S} \oslash \mathcal{S}_1$ and by checking its consistency, one can test whether a component implementing $\mathcal{S}_1$ can be reused in order to realize $\mathcal{S}$, or not. Note that by (4) the specification $\mathcal{S} \oslash \mathcal{S}_1$ is maximally permissive in the sense that it characterizes all components $\mathcal{C}_2$ such that for any $\mathcal{C}_1$ implementing $\mathcal{S}_1$, the composed system $\mathcal{C}_1 \otimes \mathcal{C}_2$ implements $\mathcal{S}$.

*Example 3.10.* A desired global behavior $G$ is depicted in Fig. 3(b), page 13. It specifies that any *get* request must be fulfilled; the access to the resource is granted for 2 time units and 5 time units in the privileged mode.

A model of $G/(Cl \otimes Acc)$ will act as a protocol converter between $Cl$ and the access controller $Acc$ ; the overall obtained system will satisfy $G$. The MECS $G/(Cl \otimes Acc)$ is represented in Fig. 4. Not surprisingly, the state $c/11'$ is inconsistent. This is because, in the state $11'$ in Fig. 3(a), the resource is granted for 4 units of time whereas in the state $c$ of the desired behavior $G$ in Fig. 3(b), it must be granted for 5 units of time. To avoid this inconsistency, the transition *extra* from state $b/10'$ to $c/11'$ will not be implemented in any model of $G/(Cl \otimes Acc)$. Thus, the protocol converter will disallow the privileged mode.

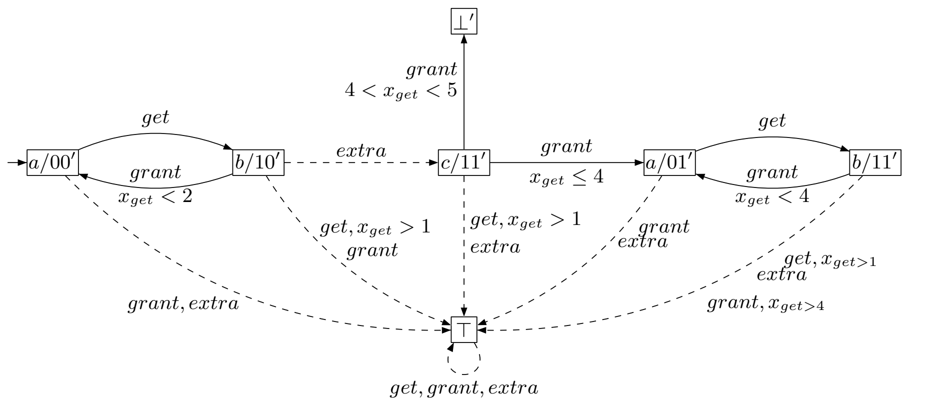


**Fig. 4.** The quotient $G/(Cl \otimes Acc)$

The quotient operation we gave has nice properties: its construction is in essence a cartesian product, thus yielding a polynomial time complexity, as opposed to the exponential blow-up caused by the region graph construction of Proposition 3.8. Also the quotient, defined at the level of specifications and abstracting from a particular choice of implementations, amounts to quotienting logical statements denoted by specifications. In the untimed setting, the quotient operation is a particular case of the exponential construction introduced by [7] for arbitrary mu-calculus statements. However, we take here advantage of the restricted logical fragment covered by the modal specifications, namely the conjunction nu-calculus [15], to get an ad-hoc polynomial-time complexity of this quotient construction. The present contribution suggests a similar situation for a timed extension of the mu-calculus.

## 4 Related work

Regarding a theory of interfaces, we compare our approach with the following settings: Interface automata of [12], timed interfaces of [13], and a timed extensions of modal specifications of [10].

*Interface automata.* In interface automata [13], an interface is represented by an input/output automaton [21], *i.e.*, an automaton whose transitions are typed with *input* and *output* rather than must and may modalities. The semantics of such an automaton is given by a two-player game: the *input* player represents the environment, and the *output* player represents the component itself. As explained [24], interfaces and modalities are in essence orthogonal to each other. Moreover, interface automata do not encompass any notion of model, and thus neither the model relation nor the consistency, because one cannot distinguish between interfaces and components. Alternatively, properties of interfaces are described in game-based logics, *e.g.*, ATL [3], with a high-cost complexity. Refinement between interface automata corresponds to the alternating refinement relation between games [4], *i.e.*, an interface refines another one if its environment is more permissive whereas its component is more restrictive. There is no notion of component reuse and shared refinement is defined in an ad-hoc manner [14]. Composition of interface automata differs from the one over modal specifications. Indeed, in interface automata, the game-based approach offers an optimistic treatment of composition: two interfaces can be composed if there exists at least one environment in which they can interact together in a safe way. In [19], Larsen et al. proposed *modal interfaces* that are modal specifications composed in a game-based manner. This work suggests that modal specifications subsume interface automata.

*Timed interfaces.* In [13], de Alfaro et al. proposed *timed interfaces* which extend timed automata just as interface automata extend automata. The composition of timed interfaces has only been partially studied, because of the underlying timed games semantics: in particular, an extra feature needs being incorporated to prevent players from winning by using Zeno strategies. Moreover, no refinement relation is defined. Recently, Chatain et al. [11] proposed a notion of alternating timed refinement between timed automata, implemented in the UPPAAL toolset [25]. In all cases, operations between specifications have not been investigated in a systematic way, and to our knowledge, no quotient construction has been addressed.

*A timed extension of modal specifications.* A timed extension of modal specifications appeared in [10] in a process algebra style. The formalism proposed is a variant of CCS whose semantics relies on the configuration graph rather than on the region graph, as done here. No logical characterization is developed, neither any notion of model relation (satisfaction) or consistency (satisfiability). Moreover, the quotient has not been considered at all.

## 5 Conclusion

Modal specifications offer a well-adapted algebraic framework for compositional reasoning on component-based systems, that enables incremental design as well as reuse of component. In this paper, we have presented a timed extension of modal specifications using event-clock timed automata. All essential features expected from a theory of interface (such as refinement, conjunction, satisfiability, product, and quotient) are fully addressed and efficiently treated in this framework.

Several research directions still need being investigated in the future. We aim at studying timed modal specifications in a broader framework than the one of MECS, since event-clock automata are strictly less expressive than timed automata. However, a generalization to arbitrary timed modal specifications raises complex issues on how different sets of clocks can be combined in the quotient. Another topic concerns a logical characterization of modal event-clock specifications (or even more general timed modal specifications), in the spirit of [15] who established the correspondence between simple modal specifications and conjunctive $\nu$-calculus. Such characterization brings insight on the expressiveness of the specification formalism.

*Acknowledgment* We are very thankful to the reviewers for relevant comments that helped us improving the paper.

## References

1. Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
2. Rajeev Alur, Limor Fix, and Thomas A. Henzinger. Event-clock automata: A determinizable class of timed automata. *Theoretical Computer Science*, 211:1–13, 1999.
3. Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49(5):672–713, 2002.
4. Rajeev Alur, Thomas A. Henzinger, Orna Kupferman, and Moshe Y. Vardi. Alternating refinement relations. In *Proceedings of the 9th International Conference on Concurrency Theory (CONCUR'98)*, volume 1466 of *Lecture Notes in Computer Science*, pages 163–178. Springer, 1998.
5. Adam Antonik, Michael Huth, Kim G. Larsen, Ulrik Nyman, and Andrzej Wasowski. 20 years of modal and mixed specifications. *Bulletin of European Association of Theoretical Computer Science*, 1(94), 2008.
6. André Arnold and Maurice Nivat. Metric interpretations of infinite trees and semantics of non deterministic recursive programs. *Theoretical Computer Science*, 11:181–205, 1980.
7. André Arnold, Aymeric Vincent, and Igor Walukiewicz. Games for synthesis of controllers with partial observation. *Theoretical Computer Science*, 303(1):7–34, 2003.
8. Nathalie Bertrand, Axel Legay, Sophie Pinchinat, and Jean-Baptiste Raclet. A compositional approach on modal specifications for timed systems. Technical report, INRIA 7039, September 2009.
9. Nathalie Bertrand, Sophie Pinchinat, and Jean-Baptiste Raclet. Refinement and consistency of timed modal specifications. In *Proceedings of the 3rd International Conference on Language and Automata Theory and Applications (LATA'09)*, volume 5457 of *Lecture Notes in Computer Science*, pages 152–163. Springer, 2009.
10. Kārlis Čerāns, Jens Chr. Godskesen, and Kim G. Larsen. Timed modal specification - theory and tools. In *Proceedings of the 5th International Conference on Computer Aided Verification (CAV'93)*, volume 697 of *Lecture Notes in Computer Science*, pages 253–267. Springer, 1993.
11. Thomas Chatain, Alexandre David, and Kim G. Larsen. Playing games with timed games. In *Proceedings of the 3rd IFAC Conference on Analysis and Design of Hybrid Systems (ADHS'09)*, 2009. To appear.

12. Luca de Alfaro and Thomas A. Henzinger. Interface automata. In *Proceedings of the 9th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE'01)*, pages 109–120, 2001.
13. Luca de Alfaro, Thomas A. Henzinger, and Mariëlle Stoelinga. Timed interfaces. In *Proceedings of the 2nd International Workshop on Embedded Software (EMSOFT'02)*, volume 2491 of *Lecture Notes in Computer Science*, pages 108–122. Springer, 2002.
14. Laurent Doyen, Thomas A. Henzinger, Barbara Jobstmann, and Tatjana Petrov. Interface theories with component reuse. In *Proceedings of the 8th International Conference on Embedded Software (EMSOFT'08)*, pages 79–88. ACM Press, 2008.
15. Guillaume Feuillade and Sophie Pinchinat. Modal specifications for the control theory of discrete-event systems. *Discrete Event Dynamic Systems*, 17(2):181–205, 2007.
16. Thomas A. Henzinger and Joseph Sifakis. The embedded systems design challenge. In *Proceedings of the 14th International Symposium on Formal Methods (FM'06)*, volume 4085 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2006.
17. Bengt Jonsson and Kim G. Larsen. On the complexity of equation solving in process algebra. In *Proceedings of the International Joint Conference on Theory and Practice of Software Development (TAPSOFT'91)*, pages 381–396. Springer, 1991.
18. Kim G. Larsen. Modal specifications. In *Proceedings of the International Workshop on Automatic Verification Methods for Finite State Systems*, volume 407 of *Lecture Notes in Computer Science*, pages 232–246. Springer, 1989.
19. Kim G. Larsen, Ulrik Nyman, and Andrzej Wasowski. Modal i/o automata for interface and product line theories. In *Proceedings of the 16th European Symposium on Programming (ESOP'07)*, volume 4421 of *Lecture Notes in Computer Science*, pages 64–79. Springer, 2007.
20. Kim G. Larsen, Ulrik Nyman, and Andrzej Wasowski. On modal refinement and consistency. In *Proceedings of the 18th International Conference on Concurrency Theory (CONCUR'07)*, volume 4703 of *Lecture Notes in Computer Science*, pages 105–119. Springer, 2007.
21. Nancy Lynch and Mark R. Tuttle. An introduction to Input/Output automata. *CWI-quarterly*, 2(3), 1989.
22. Jean-Baptiste Raclet. *Quotient de spécifications pour la réutilisation de composants*. PhD thesis, Université de Rennes I, december 2007. (In French).
23. Jean-Baptiste Raclet. Residual for component specifications. In *Proceedings of the 4th International Workshop on Formal Aspects of Component Software (FACS'07)*, 2007.
24. Jean-Baptiste Raclet, Eric Badouel, Albert Benveniste, Benoit Caillaud, and Roberto Passerone. Why are modalities good for interface theories? In *Proceedings of the 9th International Conference on Application of Concurrency to System Design (ACSD'09)*, pages 199–127. IEEE Computer Society Press, 2009.
25. The UPPAAL tool. Available at `http://www.uppaal.com/`.