Compositional Abstraction in Real-Time Model Checking[⋆]

Jasper Berendsen and Frits Vaandrager

Institute for Computing and Information Sciences
Radboud University Nijmegen
P.O. Box 9010, 6500 GL Nijmegen, the Netherlands
{J.Berendsen, F.Vaandrager}@cs.ru.nl

Abstract. The idea to use simulations (or refinements) as a compositional abstraction device is well-known, both in untimed and timed settings, and has already been studied theoretically and practically in many papers during the last three decades. Nevertheless, existing approaches do not handle two fundamental modeling concepts which, for instance, are frequently used in the popular Uppaal model checker: (1) a parallel composition operator that supports communication via shared variables as well as synchronization of actions, and (2) committed locations. In this paper, we describe a framework for compositional abstraction based on simulation relations that does support both concepts, and that is suitable for Uppaal. Our approach is very general and the only essential restriction is that the guards of input transitions do not depend on external variables. We have applied our compositional framework to verify the Zeroconf protocol for an arbitrary number of hosts.

1 Introduction

In this paper, we describe a framework for compositional abstraction based on simulation relations that is suitable for the Uppaal model checker [4]. The idea to use simulations (or refinements) as a compositional abstraction device is well-known, both in untimed and timed settings, and has already been studied theoretically and practically in many papers during the last three decades, see for instance [28,23,19,26,22,20,1,27,16,17,14,12,21]. Nevertheless, when we attempted to apply these existing approaches to fight state space explosions in a model of an industrial protocol [13], we ran into the problem that these approaches do not handle two fundamental modeling concepts that are frequently used in Uppaal.

The first concept is a parallel composition operator that supports communication via both shared variables and synchronization of actions. Models for reactive systems typically either support communication via shared variables (TLA [24], Reactive Modules [3], etc), or communication via synchronization of actions (CCS [29], I/O automata [26], etc). We are only aware of two studies of compositionality in which the two types of communication are combined [25,17].

[⋆] Research supported by NWO/EW project 612.000.103 Fault-tolerant Real-time Algorithms Analyzed Incrementally (FRAAI) and DFG/NWO bilateral cooperation project Validation of Stochastic Systems (VOSS2).

It is well known that both types of communication can be defined in terms of each other. A shared variable can be modeled as a separate process/automaton that communicates with its environment via read/write synchronization actions. However, in this approach the evaluation of, for instance, an integer expression may involve a sequence of interactions with shared variable automata. This blows up the state space and makes it more difficult to understand the model. Conversely, synchronization of actions can be modeled in a shared variable setting using some auxiliary flag variables and handshake transitions of the synchronizing automata. But again this blows up the state space and makes it harder to understand the model. The Uppaal model checker supports both shared variables and synchronization of actions, and this feature is extremely helpful for building tractable models of complex systems.

When combining shared variables and synchronization of actions, one has to deal with the scenario, illustrated in Figure 1, in which the transitions involved in a synchronization assign different values to a shared variable. One simple (but restrictive) approach, pursued in [25], is to impose syntactic conditions which ensure that the scenario does not occur: for each shared variable only one automaton gets write permission, and the other automata may read the variable but not assign a new value to it. A slightly more general approach is taken in [17], where the variables of each automaton are classified as *readable* and/or *writable*. Two automata may share writable variables, but in this case a synchronizing transition may only occur if both automata assign the same values to these variables. In practice, this means that multi-writer variables can only be updated via internal (non-synchronizing) transitions. In our opinion, the approach of [17] is basically flawed since parallel composition is not associative [6]; as a result a connection with the standard Uppaal semantics cannot be established.

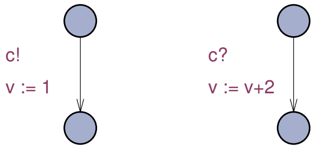

Fig. 1. Combining shared variables and synchronization of actions

In this paper, we present a very general approach that is also consistent with actual the treatment of synchronization in Uppaal. Uppaal deals with the situation of Figure 1 by first performing the assignment on the output transition $c!$, followed by the assignment on the input transition $c?$. This means that after occurrence of the synchronization transition v will have the value 3. Following Uppaal, we describe synchronization of automata by a rule of the form

$$\frac{r \xrightarrow{c!} r' \qquad s[r'] \xrightarrow{c?} s'}{r \| s \xrightarrow{\tau} r'[s'] \| s'} \tag{1}$$

Here $s[r']$ denotes state s but with the shared variables updated according to r', and $r'[s']$ denotes state r' but with the shared variables updated according to s'. In Uppaal, a synchronization may only occur if the guards of both transitions hold, and only if this is the case the assignments are carried out. This means that if we add a guard $v \neq 1$ to the rightmost transition in Figure 1, synchronization will be possible starting from any state satisfying this predicate. In a semantics with rule (1), however, synchronization will no longer be possible since after the assignment on the output transitions has been performed, the guard of the input transition no longer holds[1]. In order to rule out this scenario (which we have never observed in practical applications), our approach imposes the restriction that guards of input transitions do not refer to external (shared) variables.

The second modeling concept, which is not handled by any existing framework but needed for industrial applications, is the notion of a *committed location*. In Uppaal, locations of a timed automata can be designated as *committed*. If one automaton in a network of automata is in a committed location, time may not progress and the next transition (if any) has to start from a committed location. Committedness is useful to specify that certain transitions need to be executed *atomically* without intervening transitions from other components. Also, excluding certain behavior with committed locations may lead to serious reductions in the state space of a model [7]. In this paper, we present a compositional semantics for committedness. This is achieved by distinguishing, at the semantic level, between committed and uncommitted transitions. Our rules for describing committed locations involve negative antecedents and are similar to the rules that have been proposed in the process algebra literature to describe priorities [9,11,30,2]: a component may only perform a uncommitted τ-transition if other components may *not* perform a committed transition. Although there are some subtle differences at the technical level, basically our results show that one may view committedness as a form of priority.

We define the semantics of timed automata in terms of *timed transition systems (TTSs)*. These are labeled transition systems (LTSs) equipped with a bit of additional structure to capture relevant information about state variables, committedness of transitions, and real-time behavior. On TTSs we define the operations of parallel composition and a CCS style restriction operator. An important sanity check for our definitions is that, for any network of timed automata, the (noncompositional) Uppaal semantics (as defined in the Uppaal 4.0 help menu) is isomorphic to the (compositional) semantics obtained by associating TTSs to the timed automata in the network, composing these TTSs in parallel, applying a restriction operator, and then considering the underlying LTS. That is, if $\mathcal{N} = \langle \mathcal{A}_1, \ldots, \mathcal{A}_n \rangle$ is a network of timed automata then

$$\mathsf{LTS}(\mathcal{N}) \;\cong\; \mathsf{LTS}((\mathsf{TTS}(\mathcal{A}_1) \| \cdots \| \mathsf{TTS}(\mathcal{A}_n)) \backslash \mathcal{C}).$$

A key lemma needed to prove this result is associativity of parallel composition.

[1] Adding an antecedent $s \xrightarrow{c?}$ does not solve this problem in the general case with nondeterminism.

We define an abstraction relation on TTSs in terms of *timed step simulations*. These induce a behavioral preorder $\preceq$ that is somewhere in between strong and weak simulation. If $\mathcal{T}_1 \preceq \mathcal{T}_2$, then $\mathcal{T}_2$ can either mimic the transitions of $\mathcal{T}_1$ or (in case of an internal transition) do nothing, but it may not add internal transitions that are not present in $\mathcal{T}_1$. We establish that $\mathcal{T}_1 \preceq \mathcal{T}_2$ implies $\mathcal{T}_1 \| \mathcal{T}_3 \preceq \mathcal{T}_2 \| \mathcal{T}_3$. We briefly summarize the use of our compositional framework in the verification of the Zeroconf protocol for an arbitrary number of hosts [5]. Without our techniques, Uppaal can only verify instances with three hosts.

Section 2 introduces timed transition systems. In Section 3, we define timed step simulations and establish that the induced preorder is compositional. Section 4 presents networks of timed automata and defines their semantics both non-compositionally (as in Uppaal) and compositionally in terms of TTSs. Also, the consistency of the two semantic definitions is established and we briefly discuss the application of our framework. Finally, Section 5 discusses some extensions and future research. An appendix summarizes some of the notational conventions used in the paper.

2 Timed Transition Systems

In this section, we introduce the semantic model of timed transitions systems (TTSs). Basically, a TTS is a labeled transition system equipped with a bit of additional structure to support shared variables and committed transitions: states are defined as valuations of variables, and transitions may be committed, which gives them a certain priority in a parallel composition. TTSs can be placed in parallel and may communicate by means of shared variables and synchronization of actions. Like in CCS [29], two transitions may synchronize when their actions are complementary, leading to an internal transition in the composition.

Just to fix notation, we first recall the definition of a labeled transition system in Section 2.1. Since we consider systems that communicate via shared variables, we find it convenient to model states as functions that map state variables to values in their domain. Section 2.2 introduces a basic vocabulary for overriding and updating of functions that we need to describe shared variable communication. After these preliminaries, Section 2.3 presents the definition of a timed transition system. Next, Section 2.4 defines the operations of parallel composition and restriction, and establishes some key properties of these operations, in particular associativity of parallel composition. Finally, Section 2.5 discusses in some more detail why in our approach committedness is an attribute of transitions rather than states.

Throughout this paper, we write $\mathbb{R}_{\geq 0}$ for the set of nonnegative real numbers, $\mathbb{N}$ for the set of natural numbers, and $\mathbb{B} = \{1, 0\}$ for the set of Booleans. We let d range over $\mathbb{R}_{\geq 0}$, i, j, k, n over $\mathbb{N}$, and $b, b', \ldots$ over $\mathbb{B}$.

2.1 Labeled Transition Systems

We consider labeled transition systems with several types of state transitions, corresponding to different sets of actions. We assume a set $\mathcal{C}$ of *channels* and let c range over $\mathcal{C}$. The set of *external actions* is defined as $\mathcal{E} \triangleq \{c!, c? \mid c \in \mathcal{C}\}$. Actions of the form $c!$ are called *output actions* and actions of the form $c?$ are called *input actions*. We let e range over $\mathcal{E}$. We assume the existence of a special *internal action* τ, and write $\mathcal{E}_\tau$ for $\mathcal{E} \cup \{\tau\}$, the set of *discrete actions*. We let α range over $\mathcal{E}_\tau$. Finally, we assume a set of *durations* or *time-passage actions*, which in this paper are just the nonnegative real numbers in $\mathbb{R}_{\geq 0}$. We write Act for $\mathcal{E}_\tau \cup \mathbb{R}_{\geq 0}$, the set of *actions*, and let $a, a', \ldots$ range over Act.

The following definition is standard, except maybe for our specific choice of a universe of transition labels.

Definition 1 (LTS). *A* labeled transition system (LTS) *is a tuple*

$$\mathcal{L} = \langle S, s^0, \longrightarrow \rangle,$$

where S is a set of states, $s^0 \in S$ is the initial state, and $\longrightarrow \subseteq S \times Act \times S$ is the transition relation. We let $q, r, s, t \ldots$ range over states, and write $s \xrightarrow{a} t$ if $(s, a, t) \in \longrightarrow$. We refer to s as the source of the transition, and to t as the target. We say that an a-transition is enabled in s, notation $s \xrightarrow{a}$, whenever $s \xrightarrow{a} t$, for some t. A state s is reachable iff there exist states $s_1, \ldots s_n$ such that $s_1 = s^0$, $s_n = s$ and, for all $i < n$ there exists an a s.t. $s_i \xrightarrow{a} s_{i+1}$.

2.2 Notations for Functions

We write $dom(f)$ to denote the domain of a function f. If also X is a set, then we write $f \lceil X$ for the restriction of f to X, that is, the function g with $dom(g) = dom(f) \cap X$ such that $g(z) = f(z)$ for each $z \in dom(g)$. For functions f and g, we let $f \rhd g$ denote the *left-merge*, the combined function where f overrides g for all elements in the intersection of their domains.[2] Formally, we define $f \rhd g$ to be the function with $dom(f \rhd g) = dom(f) \cup dom(g)$ satisfying, for all $z \in dom(f \rhd g)$,

$$(f \rhd g)(z) \triangleq \begin{cases} f(z) & \text{if } z \in dom(f) \\ g(z) & \text{if } z \in dom(g) - dom(f) \end{cases}$$

We define the dual *right-merge* operator by $f \lhd g \triangleq g \rhd f$. Two functions f and g are *compatible*, notation $f \heartsuit g$, if they agree on the intersection of their domains, that is, $f(z) = g(z)$ for all $z \in dom(f) \cap dom(g)$. For compatible functions f and g, we define their *merge* by $f \| g \triangleq f \rhd g$. Whenever we write $f \| g$, we implicitly assume $f \heartsuit g$. We write $f[g]$ for the *update* of function f according to g, that is, $f[g] \triangleq (f \lhd g) \lceil dom(f)$.

The following elementary properties of $\rhd$, $\|$, $.[.]$ and $\heartsuit$ are used frequently in proofs.

[2] Essentially, this is the overriding operator "$\oplus$" from Z [32]. On finite domains, the operator is also defined in VDM[18], where it is written †. We prefer not to use a symmetric symbol for an asymmetric (non commutative) operator.

Lemma 1. *For all functions f, g and h,*

$$(f \rhd g) \rhd h = f \rhd (g \rhd h) \tag{2}$$

$$f \heartsuit g \Leftrightarrow g \heartsuit f \tag{3}$$

$$f \| g = g \| f \tag{4}$$

$$f \heartsuit g \wedge (f \| g) \heartsuit h \Leftrightarrow f \heartsuit g \;\wedge\; f \heartsuit h \;\wedge\; g \heartsuit h \tag{5}$$

$$f \| (g \| h) = (f \| g) \| h \tag{6}$$

$$f \heartsuit g \Leftrightarrow f = f[g] \tag{7}$$

$$f \;\heartsuit\; g[f] \tag{8}$$

$$f \rhd g = f \| g[f] \tag{9}$$

$$f \heartsuit g \Rightarrow f[h] \heartsuit g[h] \tag{10}$$

$$(f \rhd g)[h] = f[h] \rhd g[h] \tag{11}$$

$$f[g][h] = f[h \rhd g] \tag{12}$$

Proof. The proofs are straightforward from the definitions. Here we only prove (2), the associativity of $\rhd$. The other proofs are easier and left to the reader. Write $X = \mathit{dom}(f)$, $Y = \mathit{dom}(g)$ and $Z = \mathit{dom}(h)$. For arbitrary $z \in X \cup Y \cup Z$:

$$\begin{aligned} ((f \rhd g) \rhd h)(z) &= \begin{cases} (f \rhd g)(z) & \text{if } z \in X \cup Y \\ h(x) & \text{if } z \in Z - (X \cup Y) \end{cases} \\ &= \begin{cases} f(z) & \text{if } z \in X \\ g(z) & \text{if } z \in Y - X \\ h(z) & \text{if } z \in Z - (X \cup Y) \end{cases} \end{aligned} \tag{13}$$

$$\begin{aligned} (f \rhd (g \rhd h))(z) &= \begin{cases} f(z) & \text{if } z \in X \\ (g \rhd h)(z) & \text{if } z \in (Y \cup Z) - X \end{cases} \\ &= \begin{cases} f(z) & \text{if } z \in X \\ g(z) & \text{if } z \in ((Y \cup Z) - X) \cap Y \\ h(z) & \text{if } z \in ((Y \cup Z) - X) \cap (Z - Y) \end{cases} \end{aligned} \tag{14}$$

Now observe that (13) and (14) are equivalent since, by elementary set theory,

$$\begin{aligned} ((Y \cup Z) - X) \cap Y &= (Y \cup Z) \cap \overline{X} \cap Y \\ &= (Y \cup Z) \cap Y \cap \overline{X} = Y \cap \overline{X} = Y - X, \\ ((Y \cup Z) - X) \cap (Z - Y) &= (Y \cup Z) \cap \overline{X} \cap Z \cap \overline{Y} \\ &= (Y \cup Z) \cap Z \cap \overline{X} \cap \overline{Y} \\ &= Z \cap \overline{(X \cup Y)} = Z - (X \cup Y). \qquad \square \end{aligned}$$

2.3 Timed Transition Systems

We assume a universal set $\mathcal{V}$ of typed *variables*, with a subset $\mathcal{X} \subseteq \mathcal{V}$ of *clock variables* or *clocks*. Clocks have domain $\mathbb{R}_{\geq 0}$. We let y range over $\mathcal{V}$ and x over

$\mathcal{X}$. A *valuation* for a set $V \subseteq \mathcal{V}$ is a function that maps each variable in V to an element in its domain. We let $u, v, w, \ldots$ range over valuations, and write $\mathit{Val}(V)$ for the set of valuations for V. For valuation $v \in \mathit{Val}(V)$ and duration $d \in \mathbb{R}_{\geq 0}$, we define $v \oplus d$ to be the valuation for V that increments clocks by d, and leaves the other variables untouched, that is, for all $y \in V$,

$$(v \oplus d)(y) \triangleq \begin{cases} v(y) + d & \text{if } y \in \mathcal{X} \\ v(y) & \text{otherwise} \end{cases}$$

A subset $P \subseteq \mathit{Val}(V)$ of valuations is called a *property over* V. Let $W \supseteq V$ and $v \in \mathit{Val}(W)$. We say that P *holds in* v, notation $v \models P$, if $v\lceil V \in P$. A property P over V is *left-closed* w.r.t. time-passage if, for all $v \in \mathit{Val}(V)$ and $d \in \mathbb{R}_{\geq 0}$, $v \oplus d \models P \Rightarrow v \models P$. We say that property P over V *does not depend on* a set of variables $W \subseteq V$ if, for all $v \in \mathit{Val}(V)$ and $u \in \mathit{Val}(W)$, $v \models P \Leftrightarrow v[u] \models P$. We write $\{y_1 \mapsto z_1, \ldots, y_n \mapsto z_n\}$ for the valuation that assigns value z_i to variable y_i, for $i = 1, \ldots, n$.

The state variables of a TTS are partitioned into external and internal variables. Internal variables may only be updated by the TTS itself and not by its environment. This in contrast to external variables, which may be updated by both the TTS and its environment. A new element in our definition of a TTS is that transitions are classified as either *committed* or *uncommitted*. Committed transitions have priority over time-passage transitions and over internal transitions that are not committed. Interestingly, whereas in Uppaal committedness is an attribute of locations, it must be treated as an attribute of transitions in order to obtain a compositional semantics. This issue will be further discussed in Section 2.5.

We are now ready to formally define our notion of a timed transition system.

Definition 2 (TTS). *A* timed transition system (TTS) *is a tuple*

$$\mathcal{T} = \langle E, H, S, s^0, \longrightarrow^1, \longrightarrow^0 \rangle,$$

where $E, H \subseteq \mathcal{V}$ are disjoint sets of external and internal variables, respectively, $V = E \cup H$, $S \subseteq \mathit{Val}(V)$, and $\langle S, s^0, \mathit{Act}, \longrightarrow^1 \cup \longrightarrow^0 \rangle$ is an LTS.

We write $r \xrightarrow{a,b} s$ if $(r, a, s) \in \longrightarrow^b$. The value b determines whether or not a transition is committed. We often omit b when it equals 0. We write $\mathsf{LTS}(\mathcal{T})$ to denote the underlying LTS of $\mathcal{T}$.

We require the following axioms to hold, for all $s, t \in S$, $a, a' \in \mathit{Act}$, $b \in \mathbb{B}$, $d \in \mathbb{R}_{\geq 0}$ and $u \in \mathit{Val}(E)$,

$$s \xrightarrow{a,1} \wedge s \xrightarrow{a',b} \;\Rightarrow\; a' \in \mathcal{E} \vee (a' = \tau \wedge b) \qquad \text{(Axiom I)}$$

$$s[u] \in S \qquad \text{(Axiom II)}$$

$$s \xrightarrow{c?,b} \;\Rightarrow\; s[u] \xrightarrow{c?,b} \qquad \text{(Axiom III)}$$

$$s \xrightarrow{d} t \;\Rightarrow\; t = s \oplus d \qquad \text{(Axiom IV)}$$

A state s of a TTS is called committed, notation $\mathit{Comm}(s)$, iff it enables an outgoing committed transition, that is, $s \xrightarrow{a,1}$ for some a. Axiom I states that in a committed state neither time-passage steps nor uncommitted τ's may occur. The axiom implies that committed transitions always have a label in $\mathcal{E}_\tau$. Note that a committed state may have outgoing uncommitted transitions with a label in $\mathcal{E}$. The reason is that, for instance, an uncommitted c!-transition may synchronize with a committed c?-transition from some other component, and thereby turn into a committed τ-transition.

In general, the states of a TTS constitute a proper subset of the set of all valuations of the state variables. This feature is used to model the concept of location invariants in timed automata: if a timed automaton has, for instance, a location invariant $x \leq 1$ then this automaton may never reach a state in which $x > 1$; semantically speaking states in which $x \leq 1$ does not hold simply do not exist. In a setting with shared variable communication, complications may arise if one component wants to update a shared variable in a way that violates the location invariant of another component. In Uppaal, a state transition is only possible if in all location invariants hold in the target state. Our position is that models in which state transitions may violate location invariants are bad models. Therefore, and also because it simplifies the technicalities of our paper, we postulate in Axiom II that if the external variables of a state are changed, the result is again a state.

Axiom III states that enabledness of input transitions is not affected by changing the external variables. This is a key property that we need in order to obtain compositionality, we will discuss this axiom in more detail in the next subsection. Axiom IV, finally, asserts that if time advances with an amount d, all clocks also advance with an amount d, and the other variables remain unchanged.

2.4 Composition and Restriction

This subsection introduces the operations of parallel composition and restriction on TTSs. In our setting parallel composition is a partial operation that is only defined when TTSs are *compatible*: the initial states must be compatible functions and the internal variables of one TTS may not intersect with the variables of the other. We can avoid the restriction on the internal variables via a straightforward renaming procedure, but this would complicate the definitions.

From now on, if we have multiple indexed systems (TTSs, or later timed automata), then we use the indices also to denote the components of individual systems. For example, we let E_i denote the set of external variables of $\mathcal{T}_i$.

Definition 3 (Parallel composition). *Two TTSs $\mathcal{T}_1$ and $\mathcal{T}_2$ are* compatible *if $H_1 \cap V_2 = H_2 \cap V_1 = \emptyset$ and $s_1^0 \heartsuit s_2^0$. In this case, their* parallel composition *$\mathcal{T}_1 \| \mathcal{T}_2$ is the tuple $\mathcal{T} = \langle E, H, S, s^0, \longrightarrow^1, \longrightarrow^0 \rangle$, where $E = E_1 \cup E_2$, $H = H_1 \cup H_2$, $S = \{r \| s \mid r \in S_1 \wedge s \in S_2 \wedge r \heartsuit s\}$, $s^0 = s_1^0 \| s_2^0$, and $\longrightarrow^1$ and $\longrightarrow^0$ are the least relations that satisfy the rules in Fig. 2. Here i, j range over $\{1, 2\}$, r, r' range over S_i, s, s' range over S_j, b, b' range over $\mathbb{B}$, e ranges over $\mathcal{E}$ and c over $\mathcal{C}$.*

The external and internal variables of the composition are simply obtained by taking the union of the external and internal variables of the components,

$\dfrac{r \xrightarrow{e,b}_i r'}{r \| s \xrightarrow{e,b} r' \rhd s}$ **EXT**	$\dfrac{r \xrightarrow{c!,b}_i r' \qquad s[r'] \xrightarrow{c?,b'}_j s' \qquad i \neq j \quad Comm(r) \vee Comm(s) \Rightarrow b \vee b'}{r \| s \xrightarrow{\tau, b \vee b'} r' \lhd s'}$ **SYNC**
$\dfrac{r \xrightarrow{\tau,b}_i r' \qquad Comm(s) \Rightarrow b}{r \| s \xrightarrow{\tau,b} r' \rhd s}$ **TAU**	$\dfrac{r \xrightarrow{d}_i r' \qquad s \xrightarrow{d}_j s' \qquad i \neq j}{r \| s \xrightarrow{d} r' \| s'}$ **TIME**

Fig. 2. Rules for parallel composition of TTSs

respectively. The states (and start state) of a composed TTS are obtained by merging the states (resp. start state) of the components (in the sense of Section 2.2). The interesting part of the definition consists of the transition rules. Rule **EXT** states that an external transition of a component induces a corresponding transition of the composition. The component that takes the transition may override some of the shared variables. Observe that, since $r' \rhd s = r' \| s[r']$ by Lemma 1(9), and since $s[r']$ is a state of $\mathcal{T}_{3-i}$ by Axiom II, it follows that $r' \rhd s$ is a state of $\mathcal{T}$. Similarly, rule **TAU** states that an internal transition of a component induces a corresponding transition of the composition, except that an uncommitted transition may only occur if the other component is in an uncommitted state. Rule **SYNC** describes the synchronization of components. If $\mathcal{T}_i$ has an output transition from r to r', and if $\mathcal{T}_j$ has a corresponding input transition from s, updated by r', to s', the composition has a τ transition to $r' \lhd s'$. The synchronization is committed iff one of the participating transitions is committed. However, an uncommitted synchronization may only occur if both components are in an uncommitted state. By Axiom II for $\mathcal{T}_j$ it follows that in rule **SYNC** $s[r']$ is a state of $\mathcal{T}_j$, and by $r' \lhd s' = r'[s'] \| s'$ and Axiom II for $\mathcal{T}_i$ it follows that in rule **SYNC** $r' \lhd s'$ is a state of $\mathcal{T}$. Rule **TIME**, finally, states that a time step d of the composition may occur when both components perform a time step d. Observe that $r \heartsuit s$ and Axiom IV for both $\mathcal{T}_1$ and $\mathcal{T}_2$ imply $r' \heartsuit s'$.

In order to prove that the composition of two TTS is again a TTS, we need the following technical lemma, which states that a state of the composition is committed iff one of the component states is committed.

Lemma 2. *Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be compatible TTSs. Let $r \in S_1$ and $s \in S_2$ such that $r \heartsuit s$. Then $Comm(r \| s) \Leftrightarrow Comm(r) \vee Comm(s)$.*

Proof. $\Rightarrow$ Suppose that $Comm(r \| s)$. Then there exists a transition of the form $r \| s \xrightarrow{a,1} r' \| s'$. Since the transition is committed, it can only be established using rules **EXT**, **TAU** or **SYNC**. Clearly, if it is established using rule **EXT** or **TAU** with $i = 1$ then $Comm(r)$. Similarly, if it is established using rule **EXT** or **TAU** with $i = 2$ then $Comm(s)$. In both cases $Comm(r) \vee Comm(s)$, as required. Now suppose that the transition $r \| s \xrightarrow{a,1} r' \| s'$ is established using rule **SYNC**. Assume w.l.o.g. that $i = 1$. Then there are transitions $r \xrightarrow{c!,b}_1 r''$ and $s[r''] \xrightarrow{c?,b'}_2 s''$ such that $b \vee b'$. By Axiom III, $s \xrightarrow{c?,b'}_2$. Hence either $Comm(r)$ or $Comm(s)$, as required.

$\Leftarrow$ Now suppose $Comm(r) \vee Comm(s)$. W.l.o.g. assume that $Comm(r)$. Then, using Axiom I, we know that $r \xrightarrow{a,1}$, for some $a \in \mathcal{E}_\tau$. By application of either rule **EXT** or rule **TAU**, this implies $r \| s \xrightarrow{a,1}$. Hence $Comm(r \| s)$. □

One can check that composition is a well-defined operation on TTSs.

Lemma 3 (Composition well-defined). *Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be compatible TTSs. Then $\mathcal{T}_1 \| \mathcal{T}_2$ is a TTS.*

Proof. Since $E_1 \cap H_1 = \emptyset$ and $E_2 \cap H_2 = \emptyset$ ($\mathcal{T}_1$ and $\mathcal{T}_2$ are TTSs), and $E_1 \cap H_2 = \emptyset$ and $E_2 \cap H_1 = \emptyset$ ($\mathcal{T}_1$ and $\mathcal{T}_2$ are compatible), $E \cap H = \emptyset$. By definition, $S \subseteq Val(V)$ and $s^0 \in S$. We check that $\mathcal{T}_1 \| \mathcal{T}_2$ satisfies the four axioms for a TTS. Suppose $r \in S_1$, $s \in S_2$ and $r \heartsuit s$.

1. Assume that $r \| s \xrightarrow{a,1} \wedge r \| s \xrightarrow{a',b}$. In order to prove Axiom I, we must establish that $a' \in \mathcal{E} \vee (a' = \tau \wedge b)$. By Lemma 2, either $Comm(r)$ or $Comm(s)$. By Axiom I for $\mathcal{T}_1$ and $\mathcal{T}_2$, this implies that either r or s has no outgoing time-passage transitions. Hence, by rule **TIME**, also $r \| s$ has no outgoing time-passage transitions, that is $a' \in \mathcal{E}_\tau$. Assume $a' = \tau$. It suffices to prove that $b = 1$. Since $a' = \tau$ either rule **TAU** or rule **SYNC** has been used to prove $r \| s \xrightarrow{a',b}$. Assume w.l.o.g. that $i = 1$. If rule **TAU** is used and $Comm(r)$ then $b = 1$ by Axiom I for $\mathcal{T}_1$. If rule **TAU** is used and $Comm(s)$ then $b = 1$ by definition. If rule **SYNC** is used then $Comm(r) \vee Comm(s)$ implies $b = 1$.
2. Let $r \| s \in S$ and let $u \in Val(E)$. In order to prove Axiom II, we must show that $(r \| s)[u] \in S$. By Lemma 1(10), $r[u] \heartsuit s[u]$ and by Lemma 1(11), $(r \| s)[u] = r[u] \| s[u]$. Since $\mathcal{T}_1$ and $\mathcal{T}_2$ are compatible, $r[u] = r[u \lceil E_1]$ and $s[u] = s[u \lceil E_2]$. By Axiom II for $\mathcal{T}_1$ and $\mathcal{T}_2$, $r[u \lceil E_1] \in S_1$ and $s[u \lceil E_2] \in S_2$. Hence $(r \| s)[u] \in S$.
3. In order to prove Axiom III, suppose $r \| s \xrightarrow{c?,b}$ and $u \in Val(E)$. We must establish that $(r \| s)[u] \xrightarrow{c?,b}$. By rule **EXT**, either $r \xrightarrow{c?,b}$ or $s \xrightarrow{c?,b}$. Assume w.l.o.g. that $r \xrightarrow{c?,b}$. As in the previous case, we may infer $r[u] \heartsuit s[u]$, $(r \| s)[u] = r[u] \| s[u]$, and $r[u] = r[u \lceil E_1]$. Hence, by Axiom III for $\mathcal{T}_1$, $r[u] \xrightarrow{c?,b}$. Using rule **EXT**, we obtain $(r \| s)[u] \xrightarrow{c?,b}$.
4. Axiom IV for $\mathcal{T}_1 \| \mathcal{T}_2$ follows trivially from Axiom IV for $\mathcal{T}_1$ and $\mathcal{T}_2$ and rule **TIME**. □

Commutativity and associativity are highly desirable properties for parallel composition operators. However, associativity becomes nontrivial in a setting with both shared variables and synchronization of actions. In [6], we have shown that the composition operator defined in [17] is not associative. The parallel composition operator defined in this paper is both commutative and associative. Commutativity is immediate from the symmetry in the definitions.

Theorem 1 (Commutativity). *Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be compatible TTSs. Then $\mathcal{T}_1 \| \mathcal{T}_2 = \mathcal{T}_2 \| \mathcal{T}_1$.*

Proof. Straightforward, using the symmetry in the definitions. □

The proof of associativity is involved since there are many cases to consider. The proof heavily relies on Lemma 1.

Theorem 2 (Associativity). *Let $\mathcal{T}_1$, $\mathcal{T}_2$ and $\mathcal{T}_3$ be pairwise compatible TTSs. Then $(\mathcal{T}_1 \| \mathcal{T}_2) \| \mathcal{T}_3 = \mathcal{T}_1 \| (\mathcal{T}_2 \| \mathcal{T}_3)$.*

Proof. Observe that, since $\mathcal{T}_1$, $\mathcal{T}_2$ and $\mathcal{T}_3$ are pairwise compatible, $\mathcal{T}_1 \| \mathcal{T}_2$ is compatible with $\mathcal{T}_3$, and $\mathcal{T}_1$ is compatible with $\mathcal{T}_2 \| \mathcal{T}_3$ (use Lemma 1(5)). Let $\mathcal{T}_L = (\mathcal{T}_1 \| \mathcal{T}_2) \| \mathcal{T}_3$ and $\mathcal{T}_R = \mathcal{T}_1 \| (\mathcal{T}_2 \| \mathcal{T}_3)$. It is easy to see that $\mathcal{T}_L$ and $\mathcal{T}_R$ agree on all components, except for the transition relations. In order to prove that the set of transitions of $\mathcal{T}_R$ is contained in the set of transitions of $\mathcal{T}_L$, we distinguish 13 cases. The converse inclusion follows by a symmetric argument. The 13 cases correspond to the different ways in which an outgoing transition of a state $r \| (s \| t)$ of $\mathcal{T}_R$ may be proved using the rules of Figure 2: a transition of the composed system may either be labeled by an external action originating from one of the components (3 cases), or by a τ originating from one of the components (3 cases), or by a τ that is the result of a synchronization between 2 components (6 cases, depending on who does the output and who does the input), or by a time-passage action (1 case). The various cases are denoted below in self-explanatory notation.

- Case $(e \;\; \bullet \;\; \bullet)$. In this case $r \xrightarrow{e,b}_1 r'$ and, by application of rule **EXT**, $r \| (s \| t) \xrightarrow{e,b}_R r' \rhd (s \| t)$. Applying rule **EXT** twice gives a corresponding L-transition $(r \| s) \| t \xrightarrow{e,b}_L (r' \rhd s) \rhd t$. In fact, the R and L-transitions coincide since, by definition of $\|$ and associativity of $\rhd$, $r' \rhd (s \| t) = (r' \rhd s) \rhd t$.
- Case $(\bullet \; e \; \bullet)$. In this case $s \xrightarrow{e,b}_2 s'$ and, by double application of rule **EXT**, $r \| (s \| t) \xrightarrow{e,b}_R (s' \rhd t) \rhd r$. Via another double application of rule **EXT** we derive the corresponding L-transition $(r \| s) \| t \xrightarrow{e,b}_L (s' \rhd r) \rhd t$. The two transitions coincide since, by associativity of $\rhd$, definition of $\|$, and commutativity of $\|$,

$$(s' \rhd t) \rhd r \;=\; s' \rhd (t \rhd r) \;=\; s' \rhd (t \| r) \;=\; s' \rhd (r \| t) \;=\; s' \rhd (r \rhd t) \;=\; (s' \rhd r) \rhd t.$$

- Case $(\bullet \;\; \bullet \;\; e)$. In this case $t \xrightarrow{e,b}_3 t'$ and, by double application of rule **EXT**, $r \| (s \| t) \xrightarrow{e,b}_R (t' \rhd s) \rhd r$. Via application of rule **EXT** we derive the corresponding L-transition $(r \| s) \| t \xrightarrow{e,b}_L t' \rhd (r \| s)$. The two transitions coincide since, by associativity of $\rhd$, definition of $\|$, and commutativity of $\|$,

$$(t' \rhd s) \rhd r \;=\; t' \rhd (s \rhd r) \;=\; t' \rhd (s \| r) \;=\; t' \rhd (r \| s).$$

- Case ($\tau\ \bullet\ \bullet$). In this case $r \xrightarrow{\tau,b}_1 r'$, $Comm(s\|t) \Rightarrow b$ and, by application of rule **TAU**, $r\|(s\|t) \xrightarrow{\tau,b}_R r' \rhd (s\|t)$. By Lemma 2 and propositional logic,

  $$Comm(s\|t) \Rightarrow b \quad \Leftrightarrow \quad (Comm(s) \Rightarrow b) \wedge (Comm(t) \Rightarrow b).$$

  Thus, by applying rule **TAU** twice, we may derive the corresponding L-transition $(r\|s)\|t \xrightarrow{\tau,b}_L (r' \rhd s) \rhd t$. The two transitions coincide since, as in case ($e\ \bullet\ \bullet$), $r' \rhd (s\|t) = (r' \rhd s) \rhd t$.
- Case ($\bullet\ \tau\ \bullet$). Similar to the previous case.
- Case ($\bullet\ \bullet\ \tau$). Similar to the pre-previous case.
- Case ($!c\ ?c\ \bullet$). In this case we have $r \xrightarrow{c!,b}_1 r'$, $s[r'] \xrightarrow{c?,b'}_2 s'$, and $Comm(r) \vee Comm(s\|t) \Rightarrow b \vee b'$. By rule **EXT**, $s[r']\|t[r'] \xrightarrow{c?,b'} s' \rhd t[r']$ and, since $s[r']\|t[r'] = (s\|t)[r']$ by Lemma 1(11), application of rule **SYNC** gives $r\|(s\|t) \xrightarrow{\tau,b\vee b'}_R r' \lhd (s' \rhd t[r'])$. By Lemma 2 and propositional logic,

  $$Comm(r) \vee Comm(s\|t) \Rightarrow b \vee b' \quad \Leftrightarrow$$
  $$(Comm(r) \vee Comm(s) \Rightarrow b \vee b') \wedge (Comm(t) \Rightarrow b \vee b').$$

  Thus, by applying rule **SYNC**, we may derive a transition $r\|s \xrightarrow{\tau,b\vee b'} r' \lhd s'$, and, by subsequent application of rule **TAU**, $(r\|s)\|t \xrightarrow{\tau,b\vee b'}_L (r' \lhd s') \rhd t$. The R and L-transition coincide since, by definition of $\lhd$ and Lemma 1,

  $$\begin{aligned} r' \lhd (s' \rhd t[r']) \;&=\; (s' \rhd t[r']) \rhd r' \;=\; s' \rhd (t[r'] \rhd r') \;= \\ &=\; s' \rhd (r' \rhd t) \;=\; (s' \rhd r') \rhd t \;=\; (r' \lhd s') \rhd t. \end{aligned}$$

- Case ($?c\ !c\ \bullet$). In this case $s \xrightarrow{c!,b}_2 s'$ and by rule **EXT**, $s\|t \xrightarrow{c!,b} s' \rhd t$. Moreover, $r[s' \rhd t] \xrightarrow{c?,b'}_1 r'$, and $Comm(r) \vee Comm(s\|t) \Rightarrow b \vee b'$. Since $\|$ is commutative, rule **SYNC** gives $r\|(s\|t) \xrightarrow{\tau,b\vee b'}_R (s' \rhd t) \lhd r'$. As in the previous case,

  $$Comm(r) \vee Comm(s\|t) \Rightarrow b \vee b' \quad \Leftrightarrow$$
  $$(Comm(r) \vee Comm(s) \Rightarrow b \vee b') \wedge (Comm(t) \Rightarrow b \vee b').$$

  Also observe that, by Lemmas 1(12) and 1(7),

  $$r[s' \rhd t] \;=\; r[t][s'] \;=\; r[s'].$$

  Thus, by applying rule **SYNC** and using commutativity of $\|$, we may derive a transition $r\|s \xrightarrow{\tau,b\vee b'} s' \lhd r'$, and, by subsequent application of rule **TAU**, $(r\|s)\|t \xrightarrow{\tau,b\vee b'}_L (s' \lhd r') \rhd t$. The R and L-transition coincide since,

  $$(s' \rhd t) \lhd r' \;=\; r' \rhd (s' \rhd t) \;=\; (r' \rhd s') \rhd t \;=\; (s' \lhd r') \rhd t.$$

- Case ($!c\ \bullet\ ?c$). Similar to cases ($?c\ !c\ \bullet$) and ($!c\ ?c\ \bullet$).

- Case ($?c \bullet !c$). Similar to cases ($?c\ !c \bullet$) and ($!c\ ?c \bullet$).
- Case ($\bullet\ !c\ ?c$). Similar to case ($?c\ !c \bullet$).
- Case ($\bullet\ ?c\ !c$). Similar case ($!c\ ?c \bullet$).
- Case ($d\ d\ d$). In this case, $r \xrightarrow{d}_1 r'$, $s \xrightarrow{d}_2 s'$, and $t \xrightarrow{d}_3 t'$, for some $d \in \mathbb{R}_{\geq 0}$, and, by double application of rule **TIME**, $r \| (s \| t) \xrightarrow{d}_R r' \| (s' \| t')$. By another double application of rule **TIME** we may derive the equivalent L-transition $(r \| s) \| t \xrightarrow{d}_L (r' \| s') \| t'$. □

The next definition introduces a standard restriction operator, very similar to the one in CCS [29]. The restriction operator *internalizes* a set of channels so that no further TTSs may synchronize on it.

Definition 4 (Restriction). *Given a TTS $\mathcal{T}$ and a set $C \subseteq \mathcal{C}$ of channels, we define $\mathcal{T} \backslash C$ to be the TTS that is identical to $\mathcal{T}$, except that all transitions with a label in $\{c!, c? \mid c \in C\}$ have been removed from the transition relations.*

We write $\Sigma(\mathcal{T})$ for the set of channels that occur in transitions of $\mathcal{T}$. Using this notation, we can formulate restriction laws, as in CCS [29][p80]:

Lemma 4 (Restriction laws).

1. $\mathcal{T} \backslash C = \mathcal{T}$ *if* $\Sigma(\mathcal{T}) \cap C = \emptyset$
2. $\mathcal{T} \backslash C \backslash C' = \mathcal{T} \backslash (C \cup C')$
3. $(\mathcal{T} \| \mathcal{T}') \backslash C = \mathcal{T} \| (\mathcal{T}' \backslash C)$ *if* $\Sigma(\mathcal{T}) \cap C = \emptyset$

2.5 Committed States versus Committed Transitions

In TTSs committedness is an attribute of transitions. This contrasts with the Uppaal syntax, where committedness is defined as an attribute of locations, which are part of the state. So why don't we follow Uppaal? This would have the additional advantage that the rules of Figure 2 can be simplified to those of Figure 3. The problem has to do with the definition of committedness for composed states. There appears to be a choice between defining $r \| s$ to be committed if r *and* s are committed, or defining $r \| s$ to be committed if r *or* s is committed. In order to see that both choices are wrong, consider the four TTSs in Fig. 4, each consisting of only two states, where a $\subset$ inside a state indicates that is is committed. In the conjunctive scenario, there is no transition $((q \| r) \| s) \| t \xrightarrow{\tau} ((q \| r') \| s) \| t$, even though this is allowed according to Uppaal. In the disjunctive scenario, there is a transition $((q \| r) \| s) \| t \xrightarrow{\tau} ((q' \| r) \| s') \| t$, which is not allowed according to Uppaal.

3 Compositional Abstraction

In our approach, timed step simulations capture what it means that one TTS is an abstraction of another. In this section, we formally define timed step simulations, establish compositionality of the induced preorder, and briefly discuss possible applications of this result.

$$\frac{r \xrightarrow{e}_i r'}{r\|s \xrightarrow{e} r' \rhd s} \quad \textbf{EXT}$$

$$\frac{r \xrightarrow{\tau}_i r' \qquad \mathit{Comm}(s) \Rightarrow \mathit{Comm}(r)}{r\|s \xrightarrow{\tau} r' \rhd s} \quad \textbf{TAU}$$

$$\frac{r \xrightarrow{c!}_i r' \qquad s[r'] \xrightarrow{c?}_j s' \qquad i \neq j}{r\|s \xrightarrow{\tau} r' \lhd s'} \quad \textbf{SYNC}$$

$$\frac{r \xrightarrow{d}_i r' \qquad s \xrightarrow{d}_j s' \qquad i \neq j}{r\|s \xrightarrow{d} r'\|s'} \quad \textbf{TIME}$$

Fig. 3. Oversimplified rules for parallel composition of TTSs

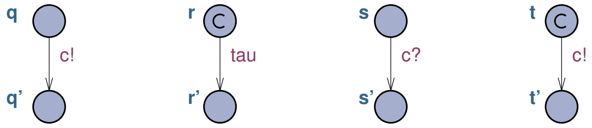

Fig. 4. A problem with committed states

A timed step simulation relates the states of two TTSs that have the same external interface, that is, the same sets of external variables. Initial states must always be related. Also, related states must agree on their external variables, and the relation must be preserved by consistently changing the external variables. If the low level system does a step, then either this can be simulated by an identical step in the high level system that preserves the relation, or the low level step is an internal computation step that preserves the simulation relation. Finally, high level committed states may only be related to low level committed states.

Definition 5 (Timed step simulation). *Two TTSs $\mathcal{T}_1$ and $\mathcal{T}_2$ are* comparable *if they have the same external variables, that is $E_1 = E_2$. Given comparable TTSs $\mathcal{T}_1$ and $\mathcal{T}_2$, we say that a relation $\mathrm{R} \subseteq S_1 \times S_2$ is a* timed step simulation *from $\mathcal{T}_1$ to $\mathcal{T}_2$, provided that $s_1^0 \, \mathrm{R} \, s_2^0$ and if $s \, \mathrm{R} \, r$ then*

1. *$s \lceil E_1 = r \lceil E_2$,*
2. *$\forall u \in \mathit{Val}(E_1) : s[u] \;\; \mathrm{R} \;\; r[u]$,*
3. *if $\mathit{Comm}(r)$ then $\mathit{Comm}(s)$,*
4. *if $s \xrightarrow{a,b} s'$ then either there exists an r' such that $r \xrightarrow{a,b} r'$ and $s' \, \mathrm{R} \, r'$, or $a = \tau$ and $s' \, \mathrm{R} \, r$.*

We write $\mathcal{T}_1 \preceq \mathcal{T}_2$ when there exists a timed step simulation from $\mathcal{T}_1$ to $\mathcal{T}_2$.

It is straightforward to prove that $\preceq$ is a preorder on the class of TTS, that is, $\preceq$ is reflexive and transitive. Our first main theorem states that $\preceq$ is a precongruence for parallel composition. This means that timed step simulations can be used as a compositional abstraction device.

Theorem 3. *Let $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3$ be TTSs with $\mathcal{T}_1$ and $\mathcal{T}_2$ comparable, $\mathcal{T}_1 \preceq \mathcal{T}_2$, and both $\mathcal{T}_1$ and $\mathcal{T}_2$ compatible with $\mathcal{T}_3$. Then $\mathcal{T}_1 \| \mathcal{T}_3 \preceq \mathcal{T}_2 \| \mathcal{T}_3$.*

Proof. Observe that, since $\mathcal{T}_1$ and $\mathcal{T}_2$ are comparable, $\mathcal{T}_1 \| \mathcal{T}_3$ and $\mathcal{T}_2 \| \mathcal{T}_3$ are comparable as well. Let $\mathcal{T}_{13} = \mathcal{T}_1 \| \mathcal{T}_3$ and $\mathcal{T}_{23} = \mathcal{T}_2 \| \mathcal{T}_3$. Let Q be a timed step simulation from $\mathcal{T}_1$ to $\mathcal{T}_2$. Define relation $\mathrm{R} \subseteq S_{13} \times S_{23}$ by

$$q \| s \;\; \mathrm{R} \;\; r \| s' \Leftrightarrow (q \, \mathrm{Q} \, r \wedge s = s').$$

We show that R is a timed step simulation from $\mathcal{T}_{13}$ to $\mathcal{T}_{23}$. First observe that $(s_1^0 \| s_3^0) \, \mathrm{R} (s_2^0 \| s_3^0)$ because $s_1^0 \, \mathrm{Q} \, s_2^0$. For arbitrary $(q \| s, r \| s) \in \mathrm{R}$, we prove that the four conditions in the definition of a timed step simulation are satisfied.

1. From $q \lceil E_1 = r \lceil E_2$ follows that $(q \| s) \lceil E_{13} = (r \| s) \lceil E_{23}$.
2. Pick $u \in \mathit{Val}(E_{13})$ and let $u' = u \lceil E_1$. Since Q is a timed step simulation, $q[u'] \, \mathrm{Q} \, r[u']$. Since $\mathcal{T}_3$ is compatible with $\mathcal{T}_1$ and $\mathcal{T}_2$, $q[u'] = q[u]$ and $r[u'] = r[u]$. Thus $q[u] \, \mathrm{Q} \, r[u]$ and therefore, by definition of R, $q[u] \| s[u] \, \mathrm{R} \, r[u] \| s[u]$. Hence $(q \| s)[u] \, \mathrm{R} (r \| s)[u]$, using Lemma 1(11).
3. We derive

$$\begin{aligned} \mathit{Comm}(r \| s) &\Rightarrow \mathit{Comm}(r) \vee \mathit{Comm}(s) && \text{(by Lemma 2)} \\ &\Rightarrow \mathit{Comm}(q) \vee \mathit{Comm}(s) && \text{(Q a timed step simulation)} \\ &\Rightarrow \mathit{Comm}(q \| s) && \text{(by Lemma 2)} \end{aligned}$$

4. Assume that $q \| s \xrightarrow{a,b} q' \| s'$. Via a case distinction on the rule instance from Figure 2 used to construct this transition, we establish that either there exists a transition $r \| s \xrightarrow{a,b} r' \| s''$ such that $q' \| s' \, \mathrm{R} \, r' \| s''$, or $a = \tau$ and $q' \| s' \, \mathrm{R} \, r \| s$.
   - Rule **EXT** with $i = 1$. Then $a \in \mathcal{E}$, $q \xrightarrow{a,b}_1 q'$ and $s' = s[q']$. Since Q is a simulation, there exists a transition $r \xrightarrow{a,b}_2 r'$ such that $q' \, \mathrm{Q} \, r'$. Let $s'' = s[r']$. Then $r \| s \xrightarrow{a,b} r' \| s''$. Since $q' \, \mathrm{Q} \, r'$, we know that $q' \lceil E_1 = r' \lceil E_2$. Hence $s' = s[q'] = s[r'] = s''$ and we may infer $q' \| s' \, \mathrm{R} \, r' \| s''$, as requested.
   - Rule **EXT** with $i = 3$. Then $a \in \mathcal{E}$, $s \xrightarrow{a,b}_3 s'$ and $q' = q[s']$. Let $s'' = s'$ and $r' = r[s']$. Then $r \| s \xrightarrow{a,b} r' \| s''$. Let $u = s' \lceil E_1$. Since $\mathcal{T}_3$ is compatible with $\mathcal{T}_1$ and $\mathcal{T}_2$, $q[s'] = q[u]$ and $r[s'] = r[u]$. Because Q is a simulation, $q[u] \, \mathrm{Q} \, r[u]$. Hence, $q[u] \| s' \, \mathrm{R} \, r[u] \| s'$. This implies $q' \| s' \, \mathrm{R} \, r' \| s''$, as requested.

– Rule **TAU** with $i = 1$. Then $a = \tau$, $q \xrightarrow{a,b}_1 q'$, $\mathit{Comm}(s) \Rightarrow b$ and $s' = s[q']$. Since Q is a simulation, either there exists a transition $r \xrightarrow{a,b}_2 r'$ such that $q' \mathrel{\mathrm{Q}} r'$, or $q' \mathrel{\mathrm{Q}} r$.

- In the first case, let $s'' = s[r']$. Then $r \| s \xrightarrow{a,b} r' \| s''$ by rule **TAU**. Since $q' \mathrel{\mathrm{Q}} r'$, we know that $q' \lceil E_1 = r' \lceil E_2$. Hence $s' = s[q'] = s[r'] = s''$ and we may infer $q' \| s' \mathrel{\mathrm{R}} r' \| s''$, as requested.
- In the second case, where $q' \mathrel{\mathrm{Q}} r$, observe that $q \lceil E_1 = r \lceil E_2 = q' \lceil E_1$. $s' = s[q'] = s[q] = s$. Hence $q' \| s' \mathrel{\mathrm{R}} r \| s$, as requested.

– Rule **TAU** with $i = 3$. Then $a = \tau$, $s \xrightarrow{a,b}_3 s'$, $\mathit{Comm}(q) \Rightarrow b$, and $q' = q[s']$. Since Q is a simulation, $\mathit{Comm}(r) \Rightarrow b$. Let $s'' = s'$ and $r' = r[s']$. Then $r \| s \xrightarrow{a,b} r' \| s''$. Let $u = s' \lceil E_1$. Since $\mathcal{T}_3$ is compatible with $\mathcal{T}_1$ and $\mathcal{T}_2$, $q[s'] = q[u]$ and $r[s'] = r[u]$. Because Q is a simulation, $q[u] \mathrel{\mathrm{Q}} r[u]$. Hence, $q[u] \| s' \mathrel{\mathrm{R}} r[u] \| s'$. This implies $q' \| s' \mathrel{\mathrm{R}} r' \| s''$, as requested.

– Rule **SYNC** with $i = 1$. Then $a = \tau$ and, for some $c \in \mathcal{C}$, $b_1, b_2 \in \mathbb{B}$, and $\hat{q} \in S_1$, $q \xrightarrow{c!,b_1}_1 \hat{q}$, $s[\hat{q}] \xrightarrow{c?,b_2}_3 s'$, $b = b_1 \vee b_2$, $\mathit{Comm}(q) \vee \mathit{Comm}(s) \Rightarrow b$, and $q' = \hat{q}[s']$. Since Q is a simulation, there exists a state $r_2 \in S_2$ such that $r \xrightarrow{c!,b_1}_2 r_2$ and $\hat{q} \mathrel{\mathrm{Q}} r_2$. Moreover, $\mathit{Comm}(r) \vee \mathit{Comm}(s) \Rightarrow b$. Since $\hat{q} \mathrel{\mathrm{Q}} r_2$, it follows that $\hat{q} \lceil E_1 = r_2 \lceil E_2$. Since $\mathcal{T}_3$ is compatible with $\mathcal{T}_1$ and $\mathcal{T}_2$, this implies that $s[\hat{q}] = s[r_2]$. Let $s'' = s'$ and $r' = r_2[s']$. We can apply rule **SYNC** to infer that $r \| s \xrightarrow{a,b} r' \| s''$. Since $\hat{q} \mathrel{\mathrm{Q}} r_2$ and $\mathcal{T}_3$ is compatible with $\mathcal{T}_1$ and $\mathcal{T}_2$, it follows that $\hat{q}[s'] \mathrel{\mathrm{Q}} r_2[s']$. This implies $q' \mathrel{\mathrm{Q}} r'$, which in turn implies $q' \| s' \mathrel{\mathrm{R}} r' \| s'$, which in turn implies $q' \| s' \mathrel{\mathrm{R}} r' \| s''$, as requested.

– Rule **SYNC** with $i = 3$. Then $a = \tau$ and, for some $c \in \mathcal{C}$, $b_1, b_2 \in \mathbb{B}$, and $\hat{s} \in S_3$, $s \xrightarrow{c!,b_1}_3 \hat{s}$, $q[\hat{s}] \xrightarrow{c?,b_2}_1 q'$, $b = b_1 \vee b_2$, $\mathit{Comm}(q) \vee \mathit{Comm}(s) \Rightarrow b$, and $s' = \hat{s}[q']$. Since $q \mathrel{\mathrm{Q}} r$, Q is a simulation, and $\mathcal{T}_3$ is compatible with $\mathcal{T}_1$ and $\mathcal{T}_2$, $q[\hat{s}] \mathrel{\mathrm{Q}} r[\hat{s}]$ and there exists a state $r' \in S_2$ such that $r[\hat{s}] \xrightarrow{c?,b_2}_2 r'$ and $q' \mathrel{\mathrm{Q}} r'$. Moreover, $\mathit{Comm}(r) \vee \mathit{Comm}(s) \Rightarrow b$. Let $s'' = \hat{s}[r']$. We can apply rule **SYNC** to infer that $r \| s \xrightarrow{a,b} r' \| s''$. Since $q' \mathrel{\mathrm{Q}} r'$, $q' \lceil E_1 = r' \lceil E_2$. Since $\mathcal{T}_3$ is compatible with $\mathcal{T}_1$ and $\mathcal{T}_2$, it follows that $\hat{s}[q'] = \hat{s}[r']$. This implies $s = s''$, which in turn implies $q' \| s' \mathrel{\mathrm{R}} r' \| s''$, as requested.

– Rule **TIME**. Then $a \in \mathbb{R}_{\geq 0}$, $b = 0$, $q \xrightarrow{a,b}_1 q'$ and $s \xrightarrow{a,b}_2 s'$. Since Q is a simulation, there exists a transition $r \xrightarrow{a,b}_2 r'$ such that $q' \mathrel{\mathrm{Q}} r'$. Let $s'' = s'$. Then $r \| s \xrightarrow{a,b} r' \| s''$ and $q' \| s' \mathrel{\mathrm{R}} r' \| s''$, as requested. □

The timed step simulation preorder $\preceq$ is in general not a precongruence for restriction. The problem is that the restriction operator removes transitions: this may affect enabledness of committed transitions and invalidate the property that high-level committed states may only be related to low-level committed states. In the theorem below, we explicitly add the condition needed for compositionality: if a state is committed in $\mathcal{T}_1$ it should still be committed in $\mathcal{T}_1 \backslash C$.

Theorem 4. *Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be comparable TTSs such that $\mathcal{T}_1 \preceq \mathcal{T}_2$. Let $C \subseteq \mathcal{C}$. If, for all states s of $\mathcal{T}_1$, $\mathit{Comm}(s) \quad \Rightarrow \quad \exists a \in \mathcal{E}_\tau - \{c!, c? \mid c \in C\} : s \xrightarrow{a,1}_1$ then $\mathcal{T}_1 \backslash C \preceq \mathcal{T}_2 \backslash C$.*

In practice, the side condition of Theorem 4 is unproblematic, for instance because in committed locations of components in a network only output transitions are enabled, and the corresponding input transitions are always enabled in other components. In such a network, a committed state always enables a committed τ-transition, which implies the side condition.

4 Networks of Timed Automata

In this section, we introduce networks of timed automata (NTA), a mathematical model for describing real-time systems inspired by the Uppaal modeling language. We present two different definitions of the semantics of NTAs and establish their equivalence. The first definition is not compositional and closely follows the Uppaal semantics (as defined in the Uppaal 4.0 help menu). The second definition constructs an LTS compositionally by first associating a TTS to each TA in the network, composing these, applying a restriction operator, and then considering the underlying LTS.

An NTA consists of a number of timed automata that may communicate via synchronization of transition labels and via a global set of multi-reader/multi-writer variables. Our model supports committed locations and a restricted form of urgency by allowing internal transitions to be urgent.[3] Our definition of timed automata abstracts from syntactic details and the various restrictions from Uppaal that are needed to make model checking decidable. These aspects that are not relevant for our compositionality result. However, in order to obtain compositionality, we need to impose some axioms on timed automata that are not required by Uppaal. Also, several Uppaal features have not been incorporated within our NTA model, in particular broadcast channels, urgent synchronization channels, and priorities. We expect that these features can be incorporated in our approach (at the price of complicating the definitions) but it remains future work to work out the details.

Definition 6 (TA). *A* timed automaton (TA) *is defined to be a tuple $\mathcal{A} = \langle L, K, l^0, E, H, v^0, I, \longrightarrow, \longrightarrow^u \rangle$, where L is a set of locations, $K \subseteq L$ is a set of committed locations, $l^0 \in L$ is the initial location, $E, H \subseteq \mathcal{V}$ are disjoint sets of external and hidden variables, respectively, $V = E \cup H$, $v^0 \in \mathit{Val}(V)$ is the initial valuation, $I : L \to 2^{\mathit{Val}(V)}$ assigns a left-closed invariant property to each location such that $v^0 \models I(l^0)$,*

$$\longrightarrow \subseteq L \times 2^{\mathit{Val}(V)} \times \mathcal{E}_\tau \times (\mathit{Val}(V) \to \mathit{Val}(V)) \times L$$

[3] Urgent internal transitions can be encoded in Uppaal by declaring a special urgent broadcast channel `urg`, labeling urgent internal transitions by `urg`!, and ensuring that no transitions carry the label `urg`?. Urgent internal transitions are very convenient for modeling systems since they allow one to specify that a component reacts instantaneously to some change of the external variables.

is the transition relation, and $\longrightarrow^u \subseteq \longrightarrow$ *is the urgent transition relation. We let* $l, \ldots$ *range over locations, write* $l \xrightarrow{g,\alpha,\rho} l'$ *if* $(l, g, \alpha, \rho, l') \in \longrightarrow$*, refer to* l *as the source of the transition, to* l' *as the target, to* g *as the guard, and to* ρ *as the update (or reset) function. We require:*

$$I(l) \textit{ does not depend on } E \qquad \text{(Axiom V)}$$

$$l \xrightarrow{g,c?,\rho} l' \quad \Rightarrow \quad g \textit{ does not depend on } E \qquad \text{(Axiom VI)}$$

$$\forall l \in K \; \forall v \in I(l) \; \exists (l \xrightarrow{g,\alpha,\rho} l') : v \models g \wedge \rho(v) \models I(l') \qquad \text{(Axiom VII)}$$

$$l \xrightarrow{g,\alpha,\rho}{}^{u} \, l' \quad \Rightarrow \quad \alpha = \tau \; \wedge \; g \textit{ does not depend on } \mathcal{X} \qquad \text{(Axiom VIII)}$$

Recall that a property P is left-closed if, for all $v \in \mathit{Val}(V)$ and $d \in \mathbb{R}_{\geq 0}$, $v \oplus d \models P \Rightarrow v \models P$. In Uppaal, left-closedness of location invariants is ensured syntactically by disallowing lower bounds on clocks in invariants. Axiom V asserts that location invariants do not depend on external variables. This restriction is not imposed by Uppaal, but run-time errors may occur in Uppaal when one automaton modifies external variables in a way that violates the location invariant of another automaton. Although it may be possible to come up with a compositional semantics for a setting without Axiom V, it is clear that the axiom eliminates a number of technical complications. We are not aware of Uppaal applications in which the axiom is violated. Axiom VI asserts that the guards of input transitions do not depend on external variables. This is a key axiom that we need for our approach to work: it ensures that the update function of an output transition does not affect the enablesness of matching input transitions. Axiom VII states that in a committed location always at least one transition is possible. We need this axiom to ensure that a state in the TTS semantics of a timed automaton is committed iff the corresponding location is committed. The axiom is a prerequisite for what is called time reactivity in [31] and timelock freedom in [8], that is, whenever time progress stops there exists at least one enabled transition. Uppaal does not impose this axiom, but we would like to argue that any model that does not satisfy it is a bad model. Axiom VIII, finally, states that only internal transitions can be urgent and that the guards of urgent transitions may not depend on clocks. The constraint that urgent transitions may not depend on clocks is syntactically enforced in Uppaal by requiring that clock variables may not occur in the guards of urgent transitions.

A network of timed automata can now be defined as a finite collection of compatible timed automata:

Definition 7 (NTA). *Two timed automata* A_1 *and* A_2 *are* compatible *if* $H_1 \cap V_2 = H_2 \cap V_1 = \emptyset$ *and* $v_1^0 \heartsuit v_2^0$. *A* network of timed automata (NTA) *consists of a finite collection* $\mathcal{N} = \langle \mathcal{A}_1, \ldots, \mathcal{A}_n \rangle$ *of pairwise compatible timed automata.*

The operational semantics of NTAs can be defined in terms of labeled transition systems.

Definition 8 (LTS semantics of NTA). *Let $\mathcal{N} = \langle \mathcal{A}_1, \ldots, \mathcal{A}_n \rangle$ be an NTA. Let $V = \bigcup_i (V_i \cup \{\mathsf{loc}_i\})$, with for each i, loc_i a fresh variable with type L_i. The* semantics *of $\mathcal{N}$, notation $\mathsf{LTS}(\mathcal{N})$, is the LTS $\langle S, s^0, \longrightarrow \rangle$, where*

$$\begin{aligned} S &= \{v \in \mathit{Val}(V) \mid \forall i : v \models I_i(v(\mathsf{loc}_i))\}, \\ s^0 &= v_1^0 \| \cdots \| v_n^0 \| \{\mathsf{loc}_1 \mapsto l_1^0, \ldots, \mathsf{loc}_n \mapsto l_n^0\}, \end{aligned}$$

and $\longrightarrow$ is defined by the rules in Fig. 5. We use the convention that if an update function $\rho : \mathit{Val}(W) \to \mathit{Val}(W)$ is applied to a valuation $v \in \mathit{Val}(W')$ with $W \subset W'$, it only affects the variables in W, that is $\rho(v) \triangleq v[\rho(v \lceil W)]$.

$$\frac{\begin{array}{c} l \xrightarrow{g,\tau,\rho}_i l' \\ s(\mathsf{loc}_i) = l \\ s \models g \end{array} \qquad \begin{array}{c} s' = \rho(s)[\{\mathsf{loc}_i \mapsto l'\}] \\ (\forall k : s(\mathsf{loc}_k) \notin K_k) \vee l \in K_i \end{array}}{s \xrightarrow{\tau} s'} \ \textbf{TAU}$$

$$\frac{\begin{array}{c} l_i \xrightarrow{g_i,c!,\rho_i} l_i' \\ s(\mathsf{loc}_i) = l_i \\ s \models g_i \end{array} \quad \begin{array}{c} l_j \xrightarrow{g_j,c?,\rho_j} l_j' \\ s(\mathsf{loc}_j) = l_j \\ s \models g_j \end{array} \quad \begin{array}{c} s' = \rho_j(\rho_i(s))[\{\mathsf{loc}_i \mapsto l_i', \mathsf{loc}_j \mapsto l_j'\}] \\ (\forall k : s(\mathsf{loc}_k) \notin K_k) \vee l_i \in K_i \vee l_j \in K_j \\ i \neq j \end{array}}{s \xrightarrow{\tau} s'} \ \textbf{SYNC}$$

$$\frac{s' = s \oplus d \quad \forall k : s(\mathsf{loc}_k) \notin K_k \quad \nexists (l \xrightarrow{g,\tau,\rho}{}_i^{u} l') : s(\mathsf{loc}_i) = l \wedge s \models g}{s \xrightarrow{d} s'} \ \textbf{TIME}$$

Fig. 5. Uppaal style LTS semantics of an NTA

Definition 8 describes the semantics of an NTA in terms of an LTS. The states of this LTS are valuations of a set V of variables. This set V contains the variables of all TAs and also, for each TA $\mathcal{A}_i$, a special variable loc_i to store the current location of $\mathcal{A}_i$. The set of states S only contains valuations in which the location invariants for all TAs hold. The initial state s^0 is the state where all automata are in their initial location and all variables have their initial value.

The transition relation $\longrightarrow$ contains two kinds of transitions: delay transitions and action transitions. We have a delay transition $s \xrightarrow{d} s'$ iff s contains no committed locations, no urgent transition is enabled in s, and s' is obtained from s by incrementing all clocks with d and leaving the other variables unchanged, that is $s' = s \oplus d$. Note that, since s' is a state, s' satisfies the location invariants. In fact, since we require that location invariants are left-closed, we have that, for all $d' \in [0, d]$, $s \oplus d'$ satisfies the location invariants. Also, since the guards of urgent transitions may not depend on clocks, we have that, for all $d' \in [0, d]$, $s \oplus d'$ does not enable any urgent transition.

For action transitions there are two cases: internal transitions and binary synchronizations. We have an internal transition $s \xrightarrow{\tau} s'$ if there is an automaton $\mathcal{A}_i$ that enables an internal transition $l \xrightarrow{g,\tau,\rho} l'$: $s(\mathsf{loc}_i) = l$ and $s \models g$. We require that either l is committed or no location in s is committed. Furthermore, s' is obtained from s by assigning to loc_i the value l', and applying the update function ρ. We have a synchronization transition $s \xrightarrow{\tau} s'$ if there are distinct components $\mathcal{A}_i$ and $\mathcal{A}_j$ that enable an output transition $l_i \xrightarrow{g_i,c!,\rho_i} l'_i$ and input transition $l_j \xrightarrow{g_j,c?,\rho_j} l'_j$, respectively. We require that either l_i or l_j is committed, or no location in s is committed. State s' is obtained from s by first applying update ρ_i and then update ρ_j. In addition the location variables are updated.

The key step towards a *compositional* semantics of NTAs is the definition below, which associates a TTS to an individual TA. Essentially this is a simplified version of Definition 8 in which a transition is made committed iff it originates from a committed location.

Definition 9 (TTS semantics of TA). *Let* $\mathcal{A} = \langle L, K, l^0, E, H, v^0, I, \longrightarrow\rangle$ *be a TA. The TTS* associated to $\mathcal{A}$, *notation* $\mathsf{TTS}(\mathcal{A})$, *is the tuple*

$$\langle E, H \cup \{\mathsf{loc}\}, S, s^0, \longrightarrow^1, \longrightarrow^0\rangle,$$

where loc *is a fresh variable with type* L, $W = E \cup H \cup \{\mathsf{loc}\}$, $S = \{v \in \mathit{Val}(W) \mid v \models I(v(\mathsf{loc}))\}$, $s^0 = v^0 \| \{\mathsf{loc} \mapsto l^0\}$, *and the transitions are defined by the rules in Fig. 6.*

$$\frac{l \xrightarrow{g,\alpha,\rho} l' \quad s(\mathsf{loc}) = l \quad s \models g \quad s' = \rho(s)[\{\mathsf{loc} \mapsto l'\}] \quad b \Leftrightarrow (l \in K)}{s \xrightarrow{\alpha,b} s'} \ \textbf{ACT}$$

$$\frac{s' = s \oplus d \qquad s(\mathsf{loc}) \notin K \qquad \nexists (l \xrightarrow{g,\tau,\rho}{}^{u} l') : s(\mathsf{loc}) = l \wedge s \models g}{s \xrightarrow{d,0} s'} \ \textbf{TIME}$$

Fig. 6. TTS semantics of a TA

We can check that the structure that we have just defined is indeed a TTS.

Lemma 5. $\mathsf{TTS}(\mathcal{A})$ *is a TTS.*

Proof. Since $\mathcal{A}$ is a TA, E and H are disjoint. Hence, since loc is fresh, also E and $H \cup \{\mathsf{loc}\}$ are disjoint. By the definition of a timed automaton, $v^0 \models I(l^0)$. This implies $s^0 \in S$, as required. We check that $\mathsf{TTS}(\mathcal{A})$ satisfies the four axioms for a TTS:

- Suppose that, for some state $s \in S$, $s \xrightarrow{a,1}$ and $s \xrightarrow{a',b}$. Committed transitions can only be inferred using rule **ACT**, and it follows that $s(\mathsf{loc}) \in K$. This implies that rule **TIME** can not be used to establish outgoing transitions from s, and thus $a' \in \mathcal{E}_\tau$. In fact, since outgoing transitions of s can only be established using rule **ACT**, it follows that $b = 1$. This suffices to prove Axiom I.

- Axiom II follows directly from the fact that, by Axiom V, location invariants do not depend on external variables.
- Axiom III follows directly from the fact that, by Axiom VI, input guards do not depend on external variables.
- Axiom IV is immediate from rule **TIME**. □

The next technical lemma asserts that a state in the TTS semantics is committed iff the corresponding location is committed.

Lemma 6. *Let $\mathcal{A}$ be a TA and let s be a state of $\mathsf{TTS}(\mathcal{A})$. Then $s(\mathsf{loc}) \in K \Leftrightarrow$ $Comm(s)$.*

Proof. Suppose that $s(\mathsf{loc}) \in K$. Let $l = s(\mathsf{loc})$. By Axiom VII, $\mathcal{A}$ has a transition $l \xrightarrow{g,\alpha,\rho} l'$ such that $s \models g$ and $\rho(s) \models I(l')$. This means that $\mathsf{TTS}(\mathcal{A})$ has a transition $s \xrightarrow{\alpha,1} s'$, where $s' = \rho(s)[\{\mathsf{loc}{\mapsto}l'\}]$. Hence $Comm(s)$.

Now suppose that $Comm(s)$. This means that s has an outgoing committed transition in $\mathsf{TTS}(\mathcal{A})$. But such a transition can only be derived using rule **ACT** provided $s(\mathsf{loc}) \in K$. □

We now come to our second main theorem, which states that a compositional semantics of NTAs defined in terms of TTSs coincides (modulo isomorphism) with the noncompositional Uppaal style semantics of Definition 8.

Theorem 5. *Let $\mathcal{N} = \langle \mathcal{A}_1, \ldots, \mathcal{A}_n \rangle$ be an NTA. Then*

$$\mathsf{LTS}(\mathcal{N}) \;\cong\; \mathsf{LTS}((\mathsf{TTS}(\mathcal{A}_1) \| \cdots \| \mathsf{TTS}(\mathcal{A}_n)) \backslash \mathcal{C}).$$

Proof. W.l.o.g. we assume that the fresh location variable of $\mathsf{TTS}(\mathcal{A}_m)$ is loc_m.[4] It follows directly from the definitions that both sides of the equation have the same states and the same initial state. What remains is to prove that both sides also have the same set of transitions. Since the $\backslash\mathcal{C}$ operation prunes away all the external transitions, we need to prove $\subseteq$ as well as $\supseteq$ for two types of transitions, namely τ-transitions and time-passage transitions.

$\subseteq$ *τ-transitions.* Assume $\mathcal{N}$ has a transition $s \xrightarrow{\tau}_{\mathcal{N}} s'$. Write $s_m = s\lceil W_m$, for $1 \leq m \leq n$. According to Definition 8, transition $s \xrightarrow{\tau}_{\mathcal{N}} s'$ is constructed either by rule **TAU** or rule **SYNC**.

Rule TAU For some $\mathcal{A}_i$ all of the following hold.

$$\begin{array}{cc} l \xrightarrow{g,\tau,\rho}_i l' & s' = \rho(s)[\{\mathsf{loc}_i{\mapsto}l'\}] \\ s(\mathsf{loc}_i) = l & (\forall k : s(\mathsf{loc}_k) \notin K_k) \vee l \in K_i \\ s \models g & \end{array}$$

[4] Without this assumption we need to drag around an isomorphism that takes care of appropriate renamings of location variables.

Let $s'_i = \rho(s_i)[\{\mathsf{loc}_i \mapsto l'\}]$ and $b \Leftrightarrow (l \in K_i)$. Then, by rule **ACT** of the TA semantics, $s_i \xrightarrow{\tau,b}_i s'_i$.
By associativity of parallel composition we may write $\mathsf{TTS}(\mathcal{A}_i) \| R$, where R is the parallel composition of TTSs of all TAs except $\mathcal{A}_i$. We define $\overline{s} = s\lceil(\bigcup_{m \neq i} W_m)$, that is, state s restricted to the variables of R. Observe that

$$\begin{aligned}
Comm(\overline{s}) &\Rightarrow \exists m \neq i. Comm(s_m) \quad \text{by Lemma 2} \\
&\Rightarrow \exists m \neq i. s_m(\mathsf{loc}_m) \in K_m \quad \text{by Lemma 6} \\
&\Rightarrow \exists m. s_m(\mathsf{loc}_m) \in K_m \\
&\Rightarrow l \in K_i \quad \text{by assumption above} \\
&\Rightarrow b
\end{aligned}$$

Hence we can apply rule **TAU** for parallel composition of TTSs

$$\frac{s_i \xrightarrow{\tau,b} s'_i \qquad Comm(\overline{s}) \Rightarrow b}{s_i \| \overline{s} \xrightarrow{\tau,b} s'_i \rhd \overline{s}} \ \textbf{TAU}$$

and after applying $\backslash\mathcal{C}$ and LTS we obtain $s \xrightarrow{\tau} s'$, as required.
Rule SYNC For some $\mathcal{A}_i$ and $\mathcal{A}_j$ all of the following hold.

$$\begin{array}{ccc}
l_i \xrightarrow{g_i, c!, \rho_i} l'_i & l_j \xrightarrow{g_j, c?, \rho_j} l'_j & s' = \rho_j(\rho_i(s))[\{\mathsf{loc}_i \mapsto l'_i, \mathsf{loc}_j \mapsto l'_j\}] \\
s(\mathsf{loc}_i) = l_i & s(\mathsf{loc}_j) = l_j & (\forall k : s(\mathsf{loc}_k) \notin K_k) \vee l_i \in K_i \vee l_j \in K_j \\
s \models g_i & s \models g_j & i \neq j
\end{array}$$

Let $s_i = s\lceil W_i$ and $s_j = s\lceil W_j$.

- Clearly $s_i \heartsuit s_j$.
- Similarly to the previous case we get $s_i \xrightarrow{c!, (l_i \in K_i)} s'_i$, where $s'_i = \rho_i(s_i)[\{\mathsf{loc}_i \mapsto l'_i\}]$.
- By Axiom VI, g_j does not depend on E_j, therefore $s_j[s'_i] \models g_j$. Furthermore, clearly $s_j[s'_i](\mathsf{loc}_j) = l_j$, and by TA semantics:

$$\frac{l_j \xrightarrow{g_j, c?, \rho_j} l'_j \quad s_j[s'_i](\mathsf{loc}_j) = l_j \quad s_j[s'_i] \models g_j \quad s'_j = \rho_j(s_j[s'_i])[\{\mathsf{loc}_j \mapsto l'_j\}] \quad b_j \Leftrightarrow (l_j \in K_j)}{s_j[s'_i] \xrightarrow{c?, b_j} s'_j} \ \textbf{ACT}$$

- $Comm(s_i) = (\exists \alpha \in \mathcal{E}_\tau : s_i \xrightarrow{\alpha,1})$. By rule **ACT** of TA semantics: $Comm(s_i) \Leftrightarrow s_i(\mathsf{loc}_i) \in K_i$. Similarly $Comm(s_j) \Leftrightarrow s_j(\mathsf{loc}_j) \in K_j$, and therefore $Comm(s_i) \vee Comm(s_j) \Rightarrow (l_i \in K_i) \vee (l_j \in K_j)$.

Now by parallel composition:

$$\frac{\begin{array}{c} s_i \xrightarrow{c!, (l_i \in K_i)} s'_i \qquad s_j[s'_i] \xrightarrow{c?, (l_j \in K_j)} s'_j \qquad i \neq j \\ Comm(s_i) \vee Comm(s_j) \Rightarrow (l_i \in K_i) \vee (l_j \in K_j) \end{array}}{s_i \| s_j \xrightarrow{\tau, (l_i \in K_i) \vee (l_j \in K_j)} s'_i \lhd s'_j} \ \textbf{SYNC}$$

By associativity of parallel composition we may write $\mathsf{TTS}(\mathcal{A}_i) \| \mathsf{TTS}(\mathcal{A}_j) \| R$, where R is the parallel composition of TTSs of all TAs except $\mathcal{A}_i$ and $\mathcal{A}_j$.

We define $\overline{s} = r\lceil(\bigcup_{m \notin \{i,j\}} W_m)$, the state s without the variables solely used by $\mathcal{A}_i$ or $\mathcal{A}_j$. By parallel composition:

Now we will proof that $s'_i \lhd s_j = s'\lceil(W_i \cup W_j)$. First we need the following identity, which is easy to derive:

$$f[g] \lceil V = (f \lceil V)[g] \tag{15}$$

Now for the proof:

$$\underbrace{s'}_{\text{expand}} =$$

$$\overbrace{\underbrace{\rho_j(\rho_i(s))}_{\text{definition of } \rho_j} [\{\mathsf{loc}_i \mapsto l'_i, \mathsf{loc}_j \mapsto l'_j\}]} =$$

$$\overbrace{\underbrace{\rho_i(s)}_{\text{definition of } \rho_i} [\rho_j(\rho_i(s) \lceil V_j)]}[\{\mathsf{loc}_i \mapsto l'_i, \mathsf{loc}_j \mapsto l'_j\}] =$$

$$\underbrace{\overbrace{s[\rho_i(s \lceil V_i)]}[\rho_j(\rho_i(s) \lceil V_j)][\{\mathsf{loc}_i \mapsto l'_i, \mathsf{loc}_j \mapsto l'_j\}]}_{\text{basic axiom}} =$$

$$s[\underbrace{\overbrace{\rho_i(s \lceil V_i) \lhd \rho_j(\rho_i(s) \lceil V_j) \lhd \{\mathsf{loc}_i \mapsto l'_i, \mathsf{loc}_j \mapsto l'_j\}}}_{\text{disjoint domains and reordering}}] =$$

$$s[\overbrace{\underbrace{\rho_i(s \lceil V_i) \lhd \{\mathsf{loc}_i \mapsto l'_i\}}_{\text{definitions } \rho_i \text{ and } s_i} \lhd \rho_j(\rho_i(s) \lceil V_j) \lhd \{\mathsf{loc}_i \mapsto l'_i, \mathsf{loc}_j \mapsto l'_j\}}] =$$

$$s[\underbrace{\overbrace{\rho_i(s_i)[\{\mathsf{loc}_i \mapsto l'_i\}]}}_{\text{equivalent}} \lhd \rho_j(\rho_i(s) \lceil V_j) \lhd \{\mathsf{loc}_i \mapsto l'_i, \mathsf{loc}_j \mapsto l'_j\}] =$$

$$s[\overbrace{s'_i} \lhd \rho_j(\underbrace{\rho_i(s)}_{\text{definition of } \rho_i} \lceil V_j) \lhd \{\mathsf{loc}_i \mapsto l'_i, \mathsf{loc}_j \mapsto l'_j\}] =$$

$$s[s'_i \lhd \underbrace{\rho_j(\overbrace{s[\rho_i(s \lceil V_i)]} \lceil V_j) \lhd \{\mathsf{loc}_i \mapsto l'_i, \mathsf{loc}_j \mapsto l'_j\}}_{\text{definitions } \rho_i \text{ and } s_i}] =$$

$$s[s'_i \lhd \overbrace{\rho_j(\underbrace{s[\rho_i(s \lceil V_i)] \lceil W_j}_{15})[\{\mathsf{loc}_i \mapsto l'_i, \mathsf{loc}_j \mapsto l'_j\}]}] =$$

$$s[s'_i \lhd \underbrace{\rho_j(\overbrace{s_j[\rho_i(s \lceil V_i)]})[\{\mathsf{loc}_i \mapsto l'_i, \mathsf{loc}_j \mapsto l'_j\}]}_{\text{by } s_i = s \lceil (V_i \cup \{\mathsf{loc}_i\})}] =$$

$$s[s'_i \lhd \underbrace{\overbrace{\rho_j(s_j[\rho_i(s_i)])[\{\mathsf{loc}_i \mapsto l'_i, \mathsf{loc}_j \mapsto l'_j\}]}}_{\text{definition of } \rho_i \text{ and } \rho_j}] =$$

$$s[s'_i \lhd \overbrace{\rho_j(s_j[\underbrace{\rho_i(s_i)[\{\mathsf{loc}_i \mapsto l'_i\}]}_{\text{equivalent}}])[\{\mathsf{loc}_j \mapsto l'_j\}]}] =$$

$$s[s'_i \lhd \underbrace{\rho_j(s_j[\overbrace{s'_i}])[\{\mathsf{loc}_j \mapsto l'_j\}]}_{\text{equivalent}}] =$$

$$s[s'_i \lhd \overbrace{s'_j}]$$

$$\frac{s_i \| s_j \xrightarrow{\tau,(l_i \in K_i)\vee(l_j \in K_j)} s'_i \lhd s'_j}{(s_i \| s_j)\|\overline{s} \xrightarrow{\tau,(l_i \in K_i)\vee(l_j \in K_j)} (s'_i \lhd s'_j) \rhd \overline{s}} \ \textbf{EXT}$$

Finally from $\mathsf{LTS}(\cdot)$ we get $s \xrightarrow{\tau} s'$.

$\supseteq$ *τ-transitions.* Assume $s \xrightarrow{\tau} s'$ in $\mathsf{LTS}((\mathsf{TTS}(\mathcal{A}_1)\| \cdots \|\mathsf{TTS}(\mathcal{A}_n))\backslash\mathcal{C})$. By definition of $\mathsf{LTS}(\cdot)$ and $\backslash\mathcal{C}$ there must be a transition $s \xrightarrow{\tau,b} s'$ in $\mathsf{TTS}(\mathcal{A}_1)\| \cdots \|\mathsf{TTS}(\mathcal{A}_n)$. By parallel composition and its associativity we see this transition is constructed either by rule **TAU** or rule **SYNC**.

Rule TAU Some $\mathsf{TTS}(\mathcal{A}_i)$ has transition $s_i \xrightarrow{\tau,b} s'_i$, and $s' = s[s'_i]$, where $s_i = s \lceil W_i$. By rule **ACT** of TA semantics all of the following hold:

$$l \xrightarrow{g,\tau,\rho}_i l' \quad s_i(\mathsf{loc}_i) = l \quad s_i \models g \quad s'_i = \rho(s_i)[\{\mathsf{loc}_i \mapsto l'\}] \quad b \Leftrightarrow (l \in K_i)$$

From this we have the following:

- $s \lceil W_k$ is the part of the state s that is determined by $\mathsf{TTS}(\mathcal{A}_k)$. Now we have:
$$\begin{aligned} l \notin K_i &\Rightarrow \neg b \\ &\Rightarrow \neg Comm(s) \\ &\Rightarrow \nexists k : Comm(s\lceil W_k) \text{ by Lemma 2} \\ &\Rightarrow \forall k : s(\mathsf{loc}_k) \notin K_k \quad \text{by TA semantics} \end{aligned}$$
- $(s_i \models g) \Rightarrow (s \models g)$
- $s' = s[s'_i] = s[\rho(s_i)[\{\mathsf{loc}_i \mapsto l'\}]] = s[\rho(s_i)][\{\mathsf{loc}_i \mapsto l'\}] = \rho(s)[\{\mathsf{loc}_i \mapsto l'\}]$

Finally by NTA semantics we are done:

$$\frac{\begin{array}{cc} l \xrightarrow{g,\tau,\rho}_i l' & \\ s(\mathsf{loc}_i) = l & s' = \rho(s)[\{\mathsf{loc}_i \mapsto l'\}] \\ s \models g & (\forall k : s(\mathsf{loc}_k) \notin K_k) \vee l \in K_i \end{array}}{s \xrightarrow{\tau}_{\mathcal{N}} s'} \ \textbf{TAU}$$

Rule SYNC Some $\mathsf{TTS}(\mathcal{A}_i)$ and $\mathsf{TTS}(\mathcal{A}_j)$, with $i \neq j$ synchronize on transitions: $s_i \xrightarrow{c!,b_i} s'_i$, $s_j[s'_i] \xrightarrow{c?,b_j} s'_j$, where $s_i = s \lceil W_i$, $s_j = s \lceil W_j$, and $s' = s[s'_i \lhd s'_j]$.
and $b = b_i \vee b_j$. By rule **ACT** of TA semantics all of the following hold:

$$l_i \xrightarrow{g_i,c!,\rho_i} l'_i \quad s_i(\mathsf{loc}_i) = l_i \quad s_i \models g_i \quad s'_i = \rho(s_i)[\{\mathsf{loc}_i \mapsto l'_i\}] \quad b_i \Leftrightarrow (l_i \in K_i)$$

$$l_j \xrightarrow{g_j,c?,\rho_j} l'_j \quad s_j[s'_i](\mathsf{loc}_j) = l_j \quad s_j[s'_i] \models g_j \quad s'_j = \rho(s_j[s'_i])[\{\mathsf{loc}_j \mapsto l'_j\}] \quad b_j \Leftrightarrow (l_j \in K_j)$$

From this we have the following:

- $s \lceil W_k$ is the part of the state s that is determined by $\mathsf{TTS}(\mathcal{A}_k)$. Now we have:
$$\begin{aligned} \neg(l_i \in K_i \vee l_j \in K_j) &\Leftrightarrow \neg b \\ &\Rightarrow \neg Comm(s) \\ &\Rightarrow \nexists k : Comm(s\lceil W_k) \text{ by Lemma 2} \\ &\Rightarrow \forall k : s(\mathsf{loc}_k) \notin K_k \quad \text{by TA semantics} \end{aligned}$$

- $s_i(\mathsf{loc}_i) = l_i \Rightarrow s(\mathsf{loc}_i) = l_i$
- $s_i \models g_i \Rightarrow s \models g_i$
- $s_j[s'_i](\mathsf{loc}_j) = l_j \Rightarrow s(\mathsf{loc}_j) = l_j$
- $(s_j[s'_i] \models g_j) \Rightarrow (s \models g_j)$

By NTA semantics:

$$\dfrac{\begin{array}{ccc} l_i \xrightarrow{g_i,c!,\rho_i} l'_i & l_j \xrightarrow{g_j,c?,\rho_j} l'_j & r' = \rho_j(\rho_i(s))[\{\mathsf{loc}_i{\mapsto}l'_i, \mathsf{loc}_j{\mapsto}l'_j\}] \\ s(\mathsf{loc}_i) = l_i & s(\mathsf{loc}_j) = l_j & (\forall k : s(\mathsf{loc}_k) \notin K_k) \vee l_i \in K_i \vee l_j \in K_j \\ s \models g_i & s \models g_j & i \neq j \end{array}}{s \xrightarrow{\tau}_{\mathcal{N}} r'}\ \textbf{SYNC}$$

Finally $\rho_j(\rho_i(s))[\{\mathsf{loc}_i{\mapsto}l'_i, \mathsf{loc}_j{\mapsto}l'_j\}] = s[s'_i \lhd s'_j]$, using the proof on page 24.

$\subseteq$ *time-passage transitions.* Assume a transition $s \xrightarrow{d} s'$. By rule **TIME** of NTA semantics all of the following hold:

$$s' = s \oplus d \qquad \forall k : s(\mathsf{loc}_k) \notin K_k \qquad \nexists(l \xrightarrow{g,\tau,\rho}{}^{u}_{i}\ l') : s(\mathsf{loc}_i) = l \wedge s \models g \tag{16}$$

We proceed our proof by induction on the number of timed automata that are put in parallel. In case $n = 1$, $(\forall k : s(\mathsf{loc}_k) \notin K_k) \Leftrightarrow s(\mathsf{loc}_1) \notin K_1$, and by TA semantics:

$$\dfrac{s' = s \oplus d \qquad s(\mathsf{loc}_1) \notin K_1 \qquad \nexists(l \xrightarrow{g,\tau,\rho}{}^{u}\ l') : s(\mathsf{loc}_1) = l \wedge s \models g}{s \xrightarrow{d,0} s'}\ \textbf{TIME}$$

Finally by definition of $\mathsf{LTS}(\cdot)$, we have $s \xrightarrow{d} s'$, the transition we needed.

Now assume the theorem holds for $m-1$, we will prove it holds for m. We define $\overline{s} = s\lceil(W_1 \cup \cdots \cup W_{m-1})$, the state s without the variables solely used by $\mathcal{A}_m$. We define $s_m = s\lceil W_m$.

Equation (16) implies the premises of rule **TIME** of TA semantics, so:

$$\dfrac{s'_m = s_m \oplus d \qquad s_m(\mathsf{loc}_m) \notin K_m \qquad \nexists(l \xrightarrow{g,\tau,\rho}{}^{u}_{m}\ l') : s_m(\mathsf{loc}_m) = l \wedge s_m \models g}{s_m \xrightarrow{d,0} s'_m}\ \textbf{TIME}$$

By the induction hypothesis there exists a transition $\overline{s} \xrightarrow{d} \overline{s}'$ in $\mathsf{LTS}((\mathsf{TTS}(\mathcal{A}_1)\| \cdots \|\mathsf{TTS}(\mathcal{A}_{m-1}))\backslash\mathcal{C})$. By definition of $\mathsf{LTS}(\cdot)$ and $\backslash\mathcal{C}$, there is a transition $\overline{s} \xrightarrow{d,0} \overline{s}'$ in $\mathsf{TTS}(\mathcal{A}_1)\| \cdots \|\mathsf{TTS}(\mathcal{A}_{m-1})$

By parallel composition we get the transition we need:

$$\dfrac{s_m \xrightarrow{d} s'_m \qquad \overline{s} \xrightarrow{d} \overline{s}'}{s_m\|\overline{s} \xrightarrow{d} s'_m\|\overline{s}'}\ \textbf{TIME}$$

Finally from $\mathsf{LTS}(\cdot)$ we get $s \xrightarrow{d} s'$.

$\supseteq$ *time-passage transitions.* Assume $s \xrightarrow{d} s'$ in $\mathsf{LTS}((\mathsf{TTS}(\mathcal{A}_1)\|\cdots\|\mathsf{TTS}(\mathcal{A}_n))\backslash\mathcal{C})$. By definition of $\mathsf{LTS}(\cdot)$ and $\backslash\mathcal{C}$ there must be a transition $s \xrightarrow{d,0} s'$ in $\mathsf{TTS}(\mathcal{A}_1)\|\cdots\|\mathsf{TTS}(\mathcal{A}_n)$.

We proceed our proof by induction on the number of timed automata that are put in parallel. In case $n = 1$, by rule **TIME** of TA semantics we the premises of rule **TIME** of NTA semantics, so:

$$\frac{s' = s \oplus d \qquad \forall k : s(\mathsf{loc}_k) \notin K_k \qquad \nexists(l \xrightarrow{g,\tau,\rho}{}_i^u\, l') : s(\mathsf{loc}_i) = l \wedge s \models g}{s \xrightarrow{d} s'}\ \textbf{TIME}$$

Now assume the theorem holds for $m - 1$, we will prove it holds for m. We define $\overline{s} = s\lceil(W_1 \cup \cdots \cup W_{m-1})$, the state s without the variables solely used by $\mathcal{A}_m$. We define $s_m = s\lceil W_m$.

By associativity of parallel composition we can write:

$$(\mathsf{TTS}(\mathcal{A}_1)\|\cdots\|\mathsf{TTS}(\mathcal{A}_{m-1}))\|\mathsf{TTS}(\mathcal{A}_m)$$

By rule **TIME** of parallel composition we have the transitions: $s_m \xrightarrow{d,0} s'_m$ in $\mathsf{TTS}(\mathcal{A}_m)$, and $\overline{s} \xrightarrow{d,0} \overline{s}'$ in $\mathsf{TTS}(\mathcal{A}_1)\|\cdots\|\mathsf{TTS}(\mathcal{A}_{m-1})$.

By TA semantics all of the following hold:

$$s_m(\mathsf{loc}_m) \notin K_m \qquad \nexists(l \xrightarrow{g,\tau,\rho}{}_m^u\, l') : s_m(\mathsf{loc}) = l \wedge s_m \models g \tag{17}$$

From the induction hypothesis we know there exists a transition $\overline{s} \xrightarrow{d}_{\mathcal{N}} \overline{s}'$ in the semantics of the NTA made up of automata $\mathcal{A}_1, \ldots, \mathcal{A}_{m-1}$. Now by rule **TIME** of NTA semantics all of the following hold:

$$\forall k : \overline{s}(\mathsf{loc}_k) \notin K_k \qquad \nexists(l \xrightarrow{g,\tau,\rho}{}_i^u\, l') : \overline{s}(\mathsf{loc}_i) = l \wedge \overline{s} \models g \tag{18}$$

Together equations (17) and (18) imply the premises of rule **TIME** of NTA semantics, so finally:

$$\frac{s' = s \oplus d \qquad \forall k : s(\mathsf{loc}_k) \notin K_k \qquad \nexists(l \xrightarrow{g,\tau,\rho}{}_i^u\, l') : s(\mathsf{loc}_i) = l \wedge s \models g}{s \xrightarrow{d}_{\mathcal{N}} s'}\ \textbf{TIME}$$

□

In the remainder of this section, we discuss how the previous results may help to alleviate the state space explosion problem. Simulation preorders preserve a rich class of properties (for instance, for Kripke structures all $\forall$CTL* properties, see [15]), but for simplicity we limit ourselves here to verification of invariants.

Definition 10 (Invariants). *Let $\mathcal{L} = \langle S, s^0, \longrightarrow\rangle$ be an LTS with $S \subseteq Val(V)$, for some set of variables V. Let P be a property over a subset of the variables of V. We say that P is an* invariant *of $\mathcal{L}$, notation $\mathcal{L} \models \forall\Box P$, iff, for all reachable states s of $\mathcal{L}$, $s \models P$.*

By extension, we say that P is an invariant *of an NTA $\mathcal{N}$, notation $\mathcal{N} \models \forall\Box P$, iff it is an invariant of $\mathsf{LTS}(\mathcal{N})$, and that P is an* invariant *of a TTS $\mathcal{T}$, notation $\mathcal{T} \models \forall\Box P$, iff it is an invariant of $\mathsf{LTS}(\mathcal{T})$.*

Timed step simulations preserve invariant properties in one direction: if an invariant property holds for the abstract system, we may conclude it also holds for the concrete system.

Theorem 6. *Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be comparable TTSs such that $\mathcal{T}_1 \preceq \mathcal{T}_2$. Let P be a property over the external variables of $\mathcal{T}_1$ and $\mathcal{T}_2$. If $\mathcal{T}_2 \models \forall\Box P$, then $\mathcal{T}_1 \models \forall\Box P$.*

Proof. Let R be a timed step simulation from $\mathcal{T}_1$ to $\mathcal{T}_2$. By a simple inductive argument, one may prove that each reachable state of $\mathcal{T}_1$ is related by R to at least one reachable state of $\mathcal{T}_2$. Since P holds for any reachable state of $\mathcal{T}_2$, P only depends on the external variables of $\mathcal{T}_2$, $E_1 = E_2$, and related states agree on their external variables, it follows that P holds for any reachable state of $\mathcal{T}_1$. □

We can lift this results to NTAs as follows. With abuse of notation write $\mathcal{A}_1 \| \cdots \| \mathcal{A}_i \prec \mathcal{B}_1 \| \cdots \| \mathcal{B}_j$ if $\mathsf{LTS}(\mathcal{A}_1 \| \cdots \| \mathsf{LTS}(\mathcal{A}_i) \prec \mathsf{LTS}(\mathcal{B}_1) \| \cdots \| \mathsf{LTS}(\mathcal{B}_j)$. Assume that $\mathcal{A}_1 \| \cdots \| \mathcal{A}_i \prec \mathcal{B}_1 \| \cdots \| \mathcal{B}_j$, and the timed automata on the right-hand-side are simpler than those on the left-hand-side. Then, by the definitions and straightforward application of Theorems 3, 4 (assuming the side condition holds), 5 and 6,

$$\langle \mathcal{B}_1, \ldots, \mathcal{B}_j, \mathcal{A}_{i+1}, \ldots, \mathcal{A}_n \rangle \models \forall\Box P \quad \Rightarrow \quad \langle \mathcal{A}_1, \ldots, \mathcal{A}_n \rangle \models \forall\Box P$$

Thus, instead of model checking $\langle \mathcal{A}_1, \ldots, \mathcal{A}_n \rangle$ it suffices to model check the simpler system obtained by substituting $\mathcal{B}_1, \ldots, \mathcal{B}_j$ for $\mathcal{A}_1, \ldots, \mathcal{A}_i$. Variations of this result can be obtained by using the restriction laws of Lemma 4.

We have successfully used this approach in order to analyze Zeroconf [13], a protocol for dynamic configuration of IPv4 link-local addresses that has been defined by the IETF. Below we briefly summarize the different types of abstractions that we applied:

1. Weakening guards and location invariants of component timed automata. Use of this type of "overapproximations" can be formally justified using timed step simulations.
2. After weakening guards and location invariants, state variables that are not mentioned in the global invariant and that are no longer tested in guards, can be omitted. Again such transformations can be formally justified using timed step simulations.
3. In order to verify instances of the protocol with an arbitrary number of hosts, we applied the Spotlight Principle [33] and abstracted all hosts except two into a "chaos" automaton, a very coarse abstraction with a single state that enables every action at any time.
4. At some point, we abstracted one automaton by a composition of two automata. Unlike the other abstractions, proving correctness of this abstraction by hand turned out to be nontrivial. With help of Thomas Chatain, we succeeded to prove existence of a timed step simulation fully automatically using Uppaal-Tiga [10], a branch of Uppaal that is able to solve timed games on NTAs. It turns out that a timed step simulation corresponds to a winning strategy for a certain timed game.

Using our abstraction techniques, Uppaal was able to verify the Zeroconf for an arbitrary number of hosts. Without our techniques, Uppaal can only handle instances with three hosts. Except for the one case, which was handled using Uppaal-Tiga, soundness of all the abstractions was proved by hand. In all these cases the simulation proofs were trivial though, and we expect that with the right tactics a general purpose theorem prover or proof assistant will be able to discard these proof obligations automatically.

5 Future Work

Our framework deals with an important part of the Uppaal modeling language, and is for instance suitable for dealing with the Zeroconf protocol. Nevertheless, several features have not been dealt with, notably

1. Urgent channels. Our approach supports urgent internal transitions but not general urgent channels as in Uppaal. We expect that urgent channels can be easily incorporated in our approach, using the concept of timed ready simulations from [16].
2. Broadcast communication. General broadcast communication, as supported by Uppaal, does not have a neat semantics: the order in which automata are declared influences the operational semantics of a network. It should be possible though to identify a well-behaved subset (for instance, by requiring that the variables modified by different input actions be disjoint). Once this has been done, we expect that the results of this paper can easily be generalized.
3. Priorities. Uppaal supports channel priorities. As we have shown, committed locations induce a priority mechanism, and we expect that channel priorities can be described in an analogous manner.

Although conceptually there will be no major difficulties involved in generalizing our results to a setting which includes these features, the proofs will become tedious and very long. Since the syntax and algorithms for Uppaal are being extended all the time, we envisage that eventually proof assistants and theorem provers will become indispensable for establishing correctness of verification methods that combine all these features and algorithms.

Although from a theoretical viewpoint, implementing our framework is less interesting, from a practical viewpoint it is all the more. We envisage a version of Uppaal that maintains networks of timed automata at different levels of abstraction, and which can automatically prove correctness of abstractions using Uppaal-Tiga and by use of a theorem prover.

References

1. M. Abadi and L. Lamport. The existence of refinement mappings. *Theoretical Computer Science*, 82(2):253–284, 1991.

2. L. Aceto, B. Bloom, and F.W. Vaandrager. Turning SOS rules into equations. *LICS'92 Special Issue of Information and Computation*, 111(1):1–52, May 1994.
3. R. Alur and T.A. Henzinger. Reactive Modules. *Formal Methods in System Design*, 15(1):7–48, 1999.
4. G. Behrmann, A. David, and K.G. Larsen. A tutorial on Uppaal. In M. Bernardo and F. Corradini, editors, *Formal Methods for the Design of Real-Time Systems, International School on Formal Methods for the Design of Computer, Communication and Software Systems, SFM-RT 2004, Bertinoro, Italy, September 13-18, 2004, Revised Lectures*, volume 3185 of *Lecture Notes in Computer Science*, pages 200–236. Springer, 2004.
5. J. Berendsen, B. Gebremichael, F.W. Vaandrager, and M. Zhang. Formal specification and analysis of zeroconf using Uppaal. Report ICIS-R07XXX, Institute for Computing and Information Sciences, Radboud University Nijmegen, 2007.
6. J. Berendsen and F.W. Vaandrager. Parallel composition in a paper of Jensen, Larsen and Skou is not associative. Technical note available at `http://www.ita.cs.ru.nl/publications/papers/fvaan/BV07.html`, September 2007.
7. G. Bhat, R. Cleaveland, and G. Lüttgen. Dynamic priorities for modeling real-time. In A. Togashi, T. Mizuno, N. Shiratori, and T. Higashino, editors, *FORTE*, volume 107 of *IFIP Conference Proceedings*, pages 321–336. Chapman & Hall, 1997.
8. H. Bowman. Modelling timeouts without timelocks. In *ARTS'99, 5th International AMAST Workshop on Real-time and Probabilistic Systems*, LNCS, page 20. Springer-Verlag, May 1999.
9. J. Camilleri and G. Winskel. CCS with priority choice. *Inf. Comput.*, 116(1):26–37, 1995.
10. F. Cassez, A. David, E. Fleury, K.G. Larsen, and D. Lime. Efficient on-the-fly algorithms for the analysis of timed games. In M. Abadi and L. de Alfaro, editors, *CONCUR*, volume 3653 of *Lecture Notes in Computer Science*, pages 66–80. Springer, 2005.
11. Rance Cleaveland, Gerald Lüttgen, and V. Natarajan. A process algebra with distributed priorities. *Theor. Comput. Sci.*, 195(2):227–258, 1998.
12. G. Frehse. *Compositional Verification of Hybrid Systems using Simulation Relations*. PhD thesis, Radboud University Nijmegen, October 2005.
13. B. Gebremichael, F.W. Vaandrager, and M. Zhang. Analysis of the Zeroconf protocol using Uppaal. In *Proceedings 6th Annual ACM & IEEE Conference on Embedded Software (EMSOFT 2006),* Seoul, South Korea, October 22-25, 2006, pages 242–251. ACM Press, 2006.
14. W.O.D. Griffioen and F.W. Vaandrager. A theory of normed simulations. *ACM Transactions on Computational Logic*, 2004. To appear.
15. O. Grumberg and D.E. Long. Model checking and modular verification. *ACM Trans. Program. Lang. Syst.*, 16(3):843–871, 1994.
16. H.E. Jensen. *Abstraction-Based Verification of Distributed Systems*. PhD thesis, Department of Computer Science, Aalborg University, Denmark, June 1999.
17. H.E. Jensen, K.G. Larsen, and A. Skou. Scaling up Uppaal: Automatic verification of real-time systems using compositionality and abstraction. In M. Joseph, editor, *Formal Techniques in Real-Time and Fault-Tolerant Systems, 6th International Symposium, FTRTFT 2000,* Pune, India, September 20-22*, Proceedings*, volume 1926 of *Lecture Notes in Computer Science*, pages 19–30. Springer, 2000.
18. C.B. Jones. *Systematic Software Development using VDM*. Prentice-Hall International, Englewood Cliffs, 1986.

19. B. Jonsson. A model and proof system for asynchronous networks. In *Proceedings of the* 4^{th} *Annual ACM Symposium on Principles of Distributed Computing,* Minaki, Ontario, Canada, pages 49–58, 1985.
20. B. Jonsson. Simulations between specifications of distributed systems. In J.C.M. Baeten and J.F. Groote, editors, *Proceedings CONCUR 91,* Amsterdam, volume 527 of *Lecture Notes in Computer Science*, pages 346–360. Springer-Verlag, 1991.
21. D.K. Kaynar, N.A. Lynch, R. Segala, and F.W. Vaandrager. *The Theory of Timed I/O Automata*. Morgan & Claypool Publishers, 2006. Synthesis Lecture on Computer Science, 101pp, ISBN 159829010X.
22. N. Klarlund and F.B. Schneider. Proving nondeterministically specified safety properties using progress measures. *Information and Computation*, 107(1):151–170, November 1993.
23. L. Lamport. What good is temporal logic? In R.E. Mason, editor, *Information Processing 83*, pages 657–668. North-Holland, 1983.
24. L. Lamport. The temporal logic of actions. *ACM Transactions on Programming Languages and Systems*, 16(3):872–923, May 1994.
25. N.A. Lynch, R. Segala, F.W. Vaandrager, and H.B. Weinberg. Hybrid I/O automata. In R. Alur, T.A. Henzinger, and E.D. Sontag, editors, *Hybrid Systems III*, volume 1066 of *Lecture Notes in Computer Science*, pages 496–510. Springer-Verlag, 1996.
26. N.A. Lynch and M.R. Tuttle. Hierarchical correctness proofs for distributed algorithms. In *Proceedings of the* 6^{th} *Annual ACM Symposium on Principles of Distributed Computing*, pages 137–151, August 1987. A full version is available as MIT Technical Report MIT/LCS/TR-387.
27. N.A. Lynch and F.W. Vaandrager. Forward and backward simulations, I: Untimed systems. *Information and Computation*, 121(2):214–233, September 1995.
28. R. Milner. An algebraic definition of simulation between programs. In *Proceedings* 2^{nd} *Joint Conference on Artificial Intelligence*, pages 481–489. British Computer Society Press, London, 1971. Also available as Report No. CS-205, Computer Science Department, Stanford University, February 1971.
29. R. Milner. *Communication and Concurrency.* Prentice-Hall International, Englewood Cliffs, 1989.
30. I. Phillips. CCS with priority guards. In K.G. Larsen and M. Nielsen, editors, *CONCUR*, volume 2154 of *Lecture Notes in Computer Science*, pages 305–320. Springer, 2001.
31. J. Sifakis. The compositional specification of timed systems - a tutorial. In N. Halbwachs and D. Peled, editors, *Proceedings of the 11th International Conference on Computer Aided Verification,* Trento, Italy, volume 1633 of *Lecture Notes in Computer Science*, pages 2–7. Springer-Verlag, July 1999.
32. J.M. Spivey, editor. *The Z notation: a reference manual.* Prentice-Hall International, 1989.
33. B. Wachter and B. Westphal. The Spotlight Principle: On Process-Summarizing State Abstractions. In A. Podelski and B. Cook, editors, *Verification, Model Checking and Abstract Interpretation*, volume 4349 of *Lecture Notes in Computer Science.* Springer, 2007. Nice, France.

A Notational Conventions

a	actions
b	Booleans
c	channels
d	durations (nonnegative real numbers)
e	external actions
f, g, h	functions (g also for guards)
i, j, k, n	natural numbers
l	locations
q, r, s, t	states
u, v, w	valuations
x	clocks
y	variables
E	sets of external variables
H	sets of internal (hidden) variables
I	mappings from locations to invariants
K	sets of committed locations
L	sets of locations
P	properties
Q, R	(simulation) relations
S	sets of states
V, W	sets of variables
X, Y, Z	sets
$\mathcal{A}$, $\mathcal{B}$	timed automata
$\mathcal{C}$	universe of channels
$\mathcal{E}$	universe of external actions
$\mathcal{L}$	labelled transition systems
$\mathcal{N}$	networks of timed automata
$\mathcal{T}$	timed transition systems
$\mathcal{V}$	universe of variables
$\mathcal{X}$	universe of clock variables
$\mathbb{B}$	the Booleans
$\mathbb{N}$	the natural numbers
$\mathbb{R}$	the real numbers
α	discrete actions
ρ	update functions
τ	the internal action