Verification of Timed Automata with Deadlines in Uppaal

Rodolfo Gómez

Technical Report No. 02-08
June 2008

Verification of Timed Automata with Deadlines in Uppaal[*]

Rodolfo Gómez

Computing Laboratory, University of Kent, United Kingdom
R.S.Gomez@kent.ac.uk

July 8, 2009[†]

Abstract

Timed Automata with Deadlines (TAD) are a form of timed automata that admit a more natural representation of urgent actions, with the additional advantage of avoiding the most common form of timelocks. We offer a compositional translation of a practically useful subset of TAD to timed safety automata (the well-known variant of timed automata where time progress conditions are expressed by invariants). More precisely, we translate networks of TAD to the modeling language of Uppaal, a state-of-the-art verification tool for timed automata. We also describe an implementation of this translation, which allows Uppaal to aid the design and analysis of TAD models.

1 Introduction

Timed automata (TA) [2] (in particular, *timed safety automata* [16]) are widely used as a formal language to model real-time systems. They strike a good balance between expressiveness and tractability, and are supported by many verification tools (e.g., Kronos [23] and Uppaal [4]).

In this paper, we focus on the representation of *urgent actions*, i.e., those whose execution cannot be delayed beyond a certain time bound. In TA models of real-time systems, urgent actions are represented indirectly by annotating automata locations with *invariants*. Invariants are clock constraints that typically impose upper bounds on the delays allowed in a particular location. Hence, when no further delay is allowed, enabled actions (i.e., those which *may* be executed) become urgent and *must* be executed.

One disadvantage of modeling urgency with invariants is that, the passage of time may be prevented even when no action is enabled at that point, giving rise to *timelocks* [8, 11]. Timelocks are anomalous states where no further execution may pass time beyond a certain bound. In TA networks (i.e., collections of concurrent, asynchronous TA), components synchronize implicitly on the passage of time, i.e., all components must agree in the allowed delays. Hence, timelocks have a global halting effect on executions and may prevent the exploration of interesting (e.g., erroneous) behaviors. Thus, in general, timelocks make the verification of correctness properties unreliable. Another known limitation of invariants is the difficulty to express certain forms of urgent behavior, such as *asap*-synchronization and other forms of synchronization [20, 7], and some forms of timeouts [9].

These limitations motivated the development of Timed Automata with Deadlines (TAD) [20, 7, 8], where *deadlines* replace invariants as time progress conditions. Deadlines are clock constraints associated with transitions in the automaton, which determine when the transition must be executed.

[*]This research has been supported by the UK Engineering and Physical Sciences Research Council under grant EP/D067197/1.

[†]This is a revised version of the original paper (June, 2008).

Importantly, neither internal actions nor synchronization on observable actions are made urgent unless they can be executed (TAD are *time-reactive* [6]). Hence, TAD avoid the most common form of timelocks occurring in formal models of real-time systems, where neither actions nor delays may occur (*time-actionlocks* [8]). TAD also allow a natural and concise representation of different urgency conditions, including those for which invariants are not well suited. (Although, there are urgency conditions that can be expressed with invariants but not with deadlines, e.g., actions that must occur in strictly less than n time units.)

Unfortunately, there is little tool support for the design and verification of TAD models (with the IF toolset being the notable exception [12]). This paper adds Uppaal [4] to the available tool support.

Our contribution. We present (to our knowledge) the first compositional translation from TAD to TA. The translation is applicable to a practically useful subset of TAD, and generates behaviorally equivalent Uppaal TA networks with at most linear (and reasonably small) increase in size. We describe an implementation of this translation, which allows Uppaal to aid the design and automatic verification of TAD models. Thus, TAD modelers will benefit from Uppaal's user-friendly GUI, rich modeling language, and efficient verification algorithms.

Related Work. The IF toolset and Uppaal each offer different modeling and verification environments. For instance, the IF toolset [12] verifies requirements on TAD models that are expressed in the alternation free μ-calculus [17], or are expressed as observers (safety properties). Instead, our translation allows requirements to be expressed in the fragment of TCTL [16] that is supported in Uppaal, and safety properties may also be specified by test automata [1].

MoDeST [13] specifications also admit deadlines. However, as far as we know, there is no tool support for the verification of such specifications (the associated toolset, MOTOR [5], cannot perform exhaustive verification, and assumes maximal progress of actions). Interestingly, this paper may suggest a way to translate (a subset of) MoDeST to Uppaal TA networks.

Bornot et al. [7] suggested a way to translate TAD to TA, but the translation is not compositional (it requires a product automaton construction) and assumes a non-standard semantics of invariants.

Barbuti and Tesei [3] proposed an extension of TA with urgent transitions, where a parameter ℓ is used to define an interval of tolerance: urgent transitions cannot be delayed for more than ℓ time units after they become enabled. This provides an interpretation of urgency in left-open intervals, but the right value for ℓ must be determined by the user, depending on the case. Also, as $\ell > 0$ is required, the semantics of deadlines can only be approximated. A non-compositional translation to TA is given, based on the region automaton [2].

Uppaal provides urgent channels to model asap-synchronization without timing constraints. Unfortunately, this restriction limits the applicability of urgent channels, and their use may give rise to timelocks due to mismatched synchronization. Nonetheless, we shall see that urgent channels allows us to obtain a compositional translation.

Paper Outline. Timed automata with deadlines are introduced in § 2. Timed automata, as supported in Uppaal, are described in § 3. The translation is formalized in § 4 (the associated tool is also described in this section). Section 5 generalizes the translation to more complex TAD models. Section 6 explains how to translate TAD models with shared data variables and diagonal constraints. Conclusions and further work are discussed in Section 7. Appendices are included to offer further details on proofs and other elements of the translation.

2 Timed Automata with Deadlines

This section introduces a common form of Timed Automata with Deadlines [20], where transitions are classified either as *lazy actions* (non-urgent), *eager actions* (urgent as soon as they are enabled), or *delayable actions* (urgent on their upper bounds) [7]. Formally, we will define the model using eager and lazy actions as the only primitives; delayable actions will be derived from these (fig. 1 (right)).

Preliminaries. Let $CA = \{a, b, \ldots\}$ and $HA = \{a?, a! \,|\, a \in CA\}$ (we define complementary labels, s.t. $\overline{a!} = a?$ and $\overline{a?} = a!$). Let $D = \{\mathit{lazy}, \mathit{eager}\}$. Let $\mathbb{C}$ be the set of clocks (a clock is variable in the non-negative reals, $\mathbb{R}^{+0}$). Let Φ be the set of clock constraints over $\mathbb{C}$, s.t.

$$\phi \in \Phi ::= \mathit{true} \,|\, x \sim c \,|\, \phi \wedge \phi$$

where $x \in \mathbb{C}$, $\sim \in \{<, >, =, \leq, \geq\}$ and $c \in \mathbb{N}$. A *valuation* is a mapping from $\mathbb{C}$ to $\mathbb{R}^{+0}$. Let $\mathbb{V}$ be the set of valuations. Let $\models$ denote the satisfiability of clock constraints over valuations. Let $v \in \mathbb{V}$, $\delta \in \mathbb{R}$ and $r \subseteq \mathbb{C}$. The valuation $v + \delta \in \mathbb{V}$ is defined s.t. $(v + \delta)(x) = v(x) + \delta$ if $v(x) + \delta \geq 0$, and $(v + \delta)(x) = 0$ otherwise, for all $x \in \mathbb{C}$. The valuation $r(v) \in \mathbb{V}$ is defined s.t. $r(v)(x) = 0$ for all $x \in r$, and $r(v)(x) = v(x)$ for all $x \notin r$.

A timed transition system [19] is a tuple $(S, s_0, \mathit{Lab} \cup \mathbb{R}^+, T)$, where S is a set of states, $s_0 \in S$ is the initial state, Lab is a set of action labels and $T \subseteq S \times \mathit{Lab} \cup \mathbb{R}^+ \times S$ is a set of transitions. *Action transitions* are of the form $(s, a, s') \in T$, $a \in \mathit{Lab}$. *Delay transitions* are of the form $(s, \delta, s') \in T$, $\delta \in \mathbb{R}^+$.

Syntax and semantics. A *timed automaton with deadlines* (TAD) is a tuple $A = (L, l_0, \mathit{Lab}, T, C)$, where L is a set of locations; $l_0 \in L$ is the initial location; $\mathit{Lab} \subseteq CA \cup HA$ is a set of labels; $T \subseteq L \times \Phi \times \mathit{Lab} \times D \times 2^{\mathbb{C}} \times L$ is a set of transitions (edges) and $C \in \mathbb{C}$ is a set of clocks.

Given a transition $t = (l, a, g, d, r, l') \in T$, l is the source location, a is the label; g is the guard; d is the deadline; r is the reset set and l' is the target location (resp., $\mathit{src}(t)$, $\mathit{lab}(t)$, $g(t)$, $d(t)$, $r(t)$ and $\mathit{tgt}(t)$). Transitions labeled with $a \in CA$ (resp. $a \in HA$) will be referred to as *completed actions* (resp. *half actions*). Transitions with deadline *lazy* (resp. *eager*) will be referred to as *lazy actions* (resp. *eager actions*).

A *TAD network* is a tuple $|A = \langle A_1, \ldots, A_n \rangle$, where $A_i = (L_i, l_{i,0}, \mathit{Lab}_i, T_i, C_i)$ is a TAD $(i : 1..n)$. Let $\mathbb{C} = \bigcup_{i=1}^n C_i$ (we say that $x \in \mathbb{C}$ is a *shared clock* if $x \in C_i \cup C_j$ for some $1 \leq i \neq j \leq n$; otherwise x is a *local clock*). The behavior of $|A$ is given by the timed transition system $(S, s_0, \mathit{Lab} \cup \mathbb{R}^+, T)$, where $S \subseteq (\prod_{i=1}^n L_i) \times \mathbb{V}$ (states are denoted $s = \langle \bar{l}, v \rangle$, where $\bar{l} \in \prod_{i=1}^n L_i$ and $v \in \mathbb{V}$); $s_0 = \langle \bar{l}_0, v_0 \rangle$ (s.t. $\bar{l}_0 = \langle l_{1,0}, \ldots, l_{n,0} \rangle$, $\forall\, x \in \mathbb{C}.\, v_0(x) = 0$); $\mathit{Lab} \subseteq CA$ and T is the smallest set of transitions that satisfies the following conditions. (We refer to elements of $\prod_{i=1}^n L_i$ as *location vectors*. We use $\bar{l}[l'_i/l_i]$ to denote substitution of l'_i for l_i in the location vector $\bar{l} = \langle l_1, \ldots, l_n \rangle$.)

1. **(completed actions)** $(\langle \bar{l}, v \rangle, a, \langle \bar{l}\,[l'_i/l_i], r_i(v) \rangle) \in T$ if
   $(l_i, a, g_i, d_i, r_i, l'_i) \in T_i$, $a \in CA$ and $v \models g_i$

2. **(synchronization)** $\left(\langle \bar{l}, v \rangle, a, \left\langle \bar{l}\,[l'_i/l_i][l'_j/l_j], (r_i \cup r_j)(v) \right\rangle\right) \in T$ if
   $(l_i, a!, g_i, d_i, r_i, l'_i) \in T_i$, $(l_j, a?, g_j, d_j, r_j, l'_j) \in T_j$ and $v \models g_i \wedge g_j$ $(i \neq j)$

3. **(delays)** $(\langle \bar{l}, v \rangle, \delta, \langle \bar{l}, v + \delta \rangle) \in T$ if
   $\delta \in \mathbb{R}^+$ and for all $\delta' \in \mathbb{R}^{+0}$, $\delta' < \delta$: (1) $(v + \delta') \not\models g(t)$ for all $t \in T_i$ $(i : 1..n)$ s.t. $\mathit{lab}(t) \in CA$,

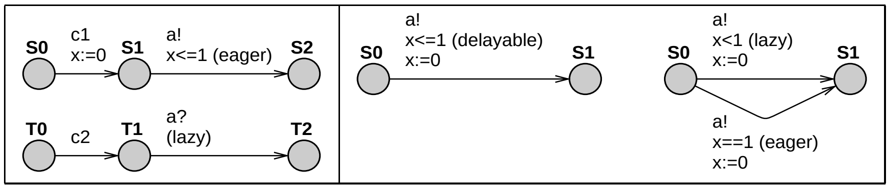


Figure 1: A TAD network (left). Delayable actions as eager/lazy actions (right).

$src(t) = l_i$ and $d(t) = eager$; and (2) $(v + \delta') \not\models g(t_i) \wedge g(t_j)$ for all $t_i \in T_i$, $t_j \in T_j$ $(i, j : 1..n, i \neq j)$ s.t. $lab(t_i) = \overline{lab(t_j)}$, $src(t_i) = l_i$, $src(t_j) = l_j$ and $d(t_i) = eager$.

where $\langle \bar{l}, v \rangle \in S$ and $\bar{l} = \langle l_1, \ldots, l_n \rangle$. A *run* is a finite or countably infinite sequence of transitions in the timed transition system.

We say that an action is *enabled* (in a given state) if its source location is in the current location vector, and its guard holds true in the current valuation. We use the term *matching* actions to refer to any pair of half actions, t and $\bar{t}$, s.t. t and $\bar{t}$ are in different components of the network and have complementary labels. We say that an action is *executable* (in a given state) if it is enabled and either is a completed action, or is a half action and there exists an enabled matching action. Matching actions must be executed simultaneously, and half actions cannot be executed autonomously.

Time-reactivity. Time-reactivity [6] is a desirable property of timed transition systems. This property holds if, from any state, either time may pass or actions can be executed. The following syntactic restriction guarantees the time-reactivity of TAD networks: For any action t, if either $d(t) = eager$ or there exists $\bar{t}$ s.t. $d(\bar{t}) = eager$, then $g(t)$ must be left-closed.[1] This restriction guarantees that delays are prevented only when eager actions can be executed (see above the semantic rule for delays). We assume, in this paper, that TAD networks satisfy this requirement.

Example. Figure 1 (left) shows a simple TAD network with two components, `S` and `T`, where `x` is a clock, `c1` and `c2` are completed actions, and `a!` and `a?` are half actions. The eager action `a!` must be executed in `S1` asap, but not later than $v(\texttt{x}) = 1$. This means that `a!` will wait for `a?` to be offered, at which point delays will be prevented and synchronization will occur. However, if `a?` is offered too late (or not at all), the network will forever remain in `S1` and a deadlock will occur, but delays will never be prevented.

Figure 1 (right) shows how combinations of eager and lazy actions may be used to represent more complex urgency conditions. For instance, so-called *delayable actions* [7] are considered urgent only when they reach their upper bounds (their guards are assumed to be right-closed). In the figure, delayable action `a!` (on the left), with guard `x<=1` is represented by one lazy action with guard `x<1` and one eager action with guard `x==1` (on the right).

3 Timed Automata in Uppaal

Uppaal is a well-known model-checker for TA, with a rich modeling language and efficient verification algorithms [4]. We describe below the subset of the language that suffices to represent TAD networks.

Urgent and *committed* locations prevent delays as soon as they are entered. In addition, whenever a committed location is entered, action interleaving is restricted to those components that are currently in committed locations.

[1] A guard g is *left-closed* if the interval $\{\delta \in \mathbb{R} \mid (v + \delta) \models g\}$ is either left-closed or left-unbounded, for all $v \in \mathbb{V}.\ v \models g$. For conjunctions of single-clock constraints, g is left-closed if $x > c$ does not occur in g, for any $x \in \mathbb{C}$ and $c \in \mathbb{N}$.

Automata synchronize on *channels*. *Binary* channels model binary and blocking synchronization (as in TAD § 2). *Broadcast* channels model a kind of asymmetric one-to-many synchronization. If b is a broadcast channel, a transition labeled with $b!$ will be executed simultaneously with all transitions labeled with $b?$ (at most one transition in each component of the network). However, the output transition ($b!$) may also be executed autonomously if no matching input transitions ($b?$) can be executed. *Urgent* channels model *asap* synchronization, i.e., synchronization on urgent channels cannot be delayed. Urgent channels may be either binary or broadcast. UPPAAL disallows clock constraints in transitions on urgent channels, and in input transitions on broadcast channels. (Transitions on urgent and broadcast channels are referred to as *urgent transitions* and *broadcast transitions*, resp.).

Preliminaries. Unless stated otherwise, we will use the notation and definitions introduced in § 2. Let $Ch = \{a, b, \ldots\}$ be a set of channels. $SL = \{a?, a! \,|\, a \in Ch\}$ is the set of synchronization labels over Ch ($\overline{a!} = a?$ and $\overline{a?} = a!$). Synchronizing transitions will be labeled with $a?, a! \in SL$ and internal transitions will be labeled with τ. Ch^{bin}, Ch^{brd}, $Ch^{\mathrm{urg}} \subseteq Ch$ resp. denote binary, broadcast and urgent channels.

Syntax and semantics. A *timed automaton* (TA) is a tuple $A = (L, l_0, Lab, T, I, C)$, where L is the set of locations; $l_0 \in L$ is the initial location; $Lab \subseteq SL \cup \{\tau\}$ is the set of labels; $T \subseteq L \times Lab \times \Phi \times 2^{\mathbb{C}} \times L$ is the set of transitions; $I : L \to \Phi$ is the invariant function; and $C \subseteq \mathbb{C}$ is the set of clocks in the automaton. Given $t = (l, a, g, r, l') \in T$, l is the source location, a is the label; g is the guard; r is the reset set and l' is the target location (resp., $src(t)$, $lab(t)$, $g(t)$, $r(t)$ and $tgt(t)$). We use $L^{\mathrm{u}}, L^{\mathrm{c}} \subseteq L$ to denote the subsets of urgent and committed locations in L (resp.). We assume that invariants are either *true* or conjunctions of upper bounds (i.e., conjunctions of simple constraints of the form $x \sim c$, where $x \in \mathbb{C}$, $\sim \in \{<, \leq\}$, and $c \in \mathbb{N}$). By convention, $I(l) = true$ for all $l \in L^{\mathrm{u}} \cup L^{\mathrm{c}}$.

A *TA network* is a tuple, $|A = \langle A_1, \ldots, A_n \rangle$, where $A_i = (L_i, l_{i,0}, Lab_i, T_i, I_i, C_i)$ is a TA ($i : 1..n$). Let $\mathbb{C} = \bigcup_{i=1}^{n} C_i$ (as in TAD networks, we admit shared clocks). The behavior of $|A$ is given by the timed transition system $(S, s_0, Lab \cup \mathbb{R}^+, T)$, where $S \subseteq (\prod_{i=1}^{n} L_i) \times \mathbb{V}$; $s_0 = \langle \bar{l}_0, v_0 \rangle$; $Lab \subseteq \{\tau\} \cup CA$ and T is the smallest set of transitions that satisfies the following conditions. ($I(\bar{l}) = \bigwedge_{i=1}^{n} I_i(l_i)$. $L^{\mathrm{u}}(\bar{l})$ and $L^{\mathrm{c}}(\bar{l})$ denote the sets of urgent and committed locations in $\bar{l}$. Given a set of indices J, we define $g_J = \bigwedge_{j \in J} g_j$, $r_J = \bigcup_{j \in J} r_j$, and we use $\bar{l}[l'_J/l_J]$ to denote substitution of l'_j for l_j in $\bar{l}$, for each $j \in J$.)

1. **(internal transition)** $\left(\langle \bar{l}, v \rangle, \tau, \langle \bar{l}\,[l'_i/l_i], r_i(v) \rangle\right) \in T$ if
$(l_i, \tau, g_i, r_i, l'_i) \in T_i$, $v \models g_i$, $r_i(v) \models I\left(\bar{l}\,[l'_i/l_i]\right)$, and $L^{\mathrm{c}}(\bar{l}) \neq \emptyset \Rightarrow l_i \in L^{\mathrm{c}}(\bar{l})$

2. **(broadcast output)** $\left(\langle \bar{l}, v \rangle, a, \langle \bar{l}\,[l'_i/l_i], r_i(v) \rangle\right) \in T$ if
$(l_i, a!, g_i, r_i, l'_i) \in T_i$, $a \in Ch^{\mathrm{brd}}$, $v \models g_i$, $r_i(v) \models I\left(\bar{l}\,[l'_i/l_i]\right)$,
$L^{\mathrm{c}}(\bar{l}) \neq \emptyset \Rightarrow l_i \in L^{\mathrm{c}}(\bar{l})$ and there is no $(l_j, a?, g_j, r_j, l'_j) \in T_j$ ($i \neq j$) s.t. $v \models g_j$ and $(r_j \cup r_i)(v) \models I\left(\bar{l}\,[l'_i/l_i][l'_j/l_j]\right)$

3. **(binary sync.)** $\left(\langle \bar{l}, v \rangle, a, \left\langle \bar{l}\,[l'_i/l_i][l'_j/l_j], (r_i \cup r_j)(v) \right\rangle\right) \in T$ if
$(l_i, a!, g_i, r_i, l'_i) \in T_i$, $(l_j, a?, g_j, r_j, l'_j) \in T_j$, $a \in Ch^{\mathrm{bin}}$, $v \models g_i \wedge g_j$,
$(r_j \cup r_i)(v) \models I\left(\bar{l}\,[l'_i/l_i][l'_j/l_j]\right)$ and $L^{\mathrm{c}}(\bar{l}) \neq \emptyset \Rightarrow \{l_i, l_j\} \cap L^{\mathrm{c}}(\bar{l}) \neq \emptyset$

4. **(broadcast sync.)** $\left(\langle \bar{l}, v \rangle, a, \langle \bar{l}\,[l'_i/l_i][l'_J/l_J], (r_i \cup r_J)(v) \rangle\right) \in T$ iff
$(l_i, a!, g_i, r_i, l'_i) \in T_i$, $a \in Ch^{\mathrm{brd}}$, and J is the maximal set of indices s.t. $J \subseteq [1..n] \setminus \{i\}$,

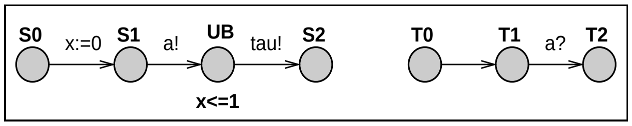


Figure 2: Urgent channels

$(l_j, a?, g_j, r_j, l'_j) \in T_j$ for all $j \in J$, $v \models g_i \wedge g_J$,
$(r_i \cup r_J)\,(v) \models I\left(\bar{l}\,[l'_i/l_i][l'_J/l_J]\right)$ and $L^{\mathrm{c}}(\bar{l}) \neq \emptyset \Rightarrow \exists\, k \in J \cup \{i\}.\ l_k \in L^{\mathrm{c}}(\bar{l})$

5. **(delays)** $\left(\langle \bar{l}, v\rangle, \delta, \langle \bar{l}, v+\delta\rangle\right) \in T$ if
$\delta \in \mathbb{R}^{+}$, $(v+\delta) \models I(\bar{l})$, $L^{\mathrm{u}}(\bar{l}) \cup L^{\mathrm{c}}(\bar{l}) = \emptyset$, and for all $\delta' \in \mathbb{R}^{+0}$, $\delta' < \delta$: (1) $(v+\delta') \not\models g(t)$ for all $t \in T_i$ $(i : 1..n)$ s.t. $lab(t) = b!, b \in Ch^{\mathrm{urg}} \cap Ch^{\mathrm{brd}}$ and $src(t) = l_i$; and (2) $(v+\delta') \not\models g(t_i) \wedge g(t_j)$ for all $t_i \in T_i$, $t_j \in T_j$ $(i, j : 1..n, i \neq j)$ s.t. $lab(t_i) = u!$, $lab(t_j) = u?$, $u \in Ch^{\mathrm{urg}} \cap Ch^{\mathrm{bin}}$, $src(t_i) = l_i$ and $src(t_j) = l_j$.

where $\langle \bar{l}, v\rangle \in S$ and $\bar{l} = \langle l_1, \ldots, l_n\rangle$. A *run* is a finite or countably infinite sequence of transitions in the timed transition system.

Delays must satisfy all component invariants, and will be prevented in urgent and committed locations, and in states where urgent transitions are enabled (either output transitions on urgent broadcast channels or matching transitions on urgent binary channels). Note that, Uppaal adopts *strong invariants*: transitions that would otherwise invalidate the invariant of the target state cannot be executed (i.e., states with invalid invariants are unreachable).[2]

Urgent actions in Uppaal. These can be modeled with invariants, urgent or committed locations, or urgent channels. The latter are the safest primitives, because delays are not prevented unless synchronization is enabled (although, not necessarily executable). However, clock constraints are disallowed in urgent transitions, and timelocks may occur due to mismatched synchronization.

By way of example, fig. 2 shows a simple TA network with two components, `S` and `T`, where `a` is an urgent channel, `tau` is an urgent broadcast channel and `x` is a clock.[3] This TA network attempts to recreate the behavior of the TAD network of fig. 1 (left). The semantics of urgent channels ensure that `a!` and `a?` will synchronize asap, but no later than $v(\texttt{x}) = 1$. This upper bound is expressed by the invariant `x<=1` in the auxiliary location `UB` (relying on Uppaal's strong invariant semantics). By semantics of urgent broadcast channels, the auxiliary `tau!` transition is executed as soon as `UB` is entered (there is no need for a matching `tau?`-transition). However, a timelock occurs if `a?` is offered when $v(\texttt{x}) > 1$; both `a!` and `a?` will be simultaneously enabled, but synchronization cannot occur because executing `a!` would invalidate the invariant in `UB`.

4 Translating TAD networks to Uppaal TA networks

In this section, we define the compositional translation of a class of TAD networks to Uppaal TA networks. The class of TAD networks is defined by the following syntactic conditions.

1. Eager actions do not share labels with lazy actions. This simplifies the presentation but does not sacrifice generality, as any TAD network can be brought into this form by a syntactic transformation (see Appendix A).

[2]In contrast, states with *weak* invariants are reachable (but delays are prevented) [7].
[3]We omit trivial invariants (*true*) and τ labels (internal transitions).

2. Outgoing transitions in the same location are guarded on the same clock (or are trivially enabled). Different clocks, and diagonal clock constraints, can be dealt with at the expense of more complex translations (see § 5 and § 6).

3. If a clock x occurs in a lower bound of an eager action in component P, then no other component Q may reset x. Also, if a clock x is reset by an upper-bounded eager action in component P, then x cannot occur in lower bounds of transitions in any other component Q. This restriction is necessary to guarantee the soundness of our translation (see Appendix B).

We believe that, the restrictions enumerated above do not compromise the practical applicability of our translation. The first restriction can be enforced on more general TAD networks by a preprocessing step (in fact, our tool implements such preprocessing § 4.4). Also, in practice (and judging from examples in the literature of timed automata verification) the second and third restrictions are satisfied by (or may be naturally enforced on) a large class of models.

4.1 The translation in examples

The translation represents lazy completed actions (in the TAD network) by internal transitions (in the resulting TA network); lazy half actions by transitions on regular channels; eager completed actions by output transitions on urgent broadcast channels; and eager half actions by transitions on urgent channels. Auxiliary locations and transitions will be added to ensure that urgent transitions are enabled only when the corresponding eager actions are enabled.

The simplest case, which is illustrated by fig. 3, corresponds to locations where all eager actions are continuously enabled (`c1` and `c2` are completed actions).[4] More interesting is the translation of eager actions with lower and upper bounds, illustrated by fig. 4. Auxiliary locations are introduced to "wait" until the current valuation reaches the action's lower bound. Then, an equivalent urgent transition is offered. We will refer to such auxiliary locations as *lb-locations*. For instance, `LB1` and `LB2` in the TA on the left, guarantee that `a!` is enabled only when $v(\mathtt{x}) \geq 1$.

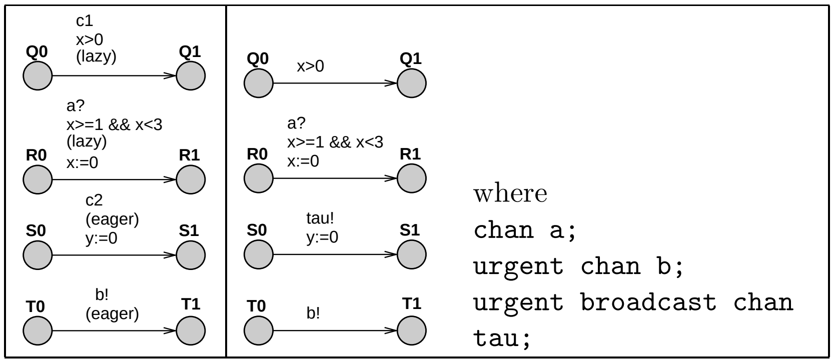
In order to disable urgent transitions after the upper bound is reached, auxiliary locations are introduced as intermediate targets with the upper bounds as invariants. We will refer to such auxiliary locations as *ub-locations*. For instance, `UB1`, in the TA in the middle, prevents `b!` from being executed when $v(\mathtt{y}) > 1$.

Upper bounds in eager actions also prompt the generation of *escape locations* (e-locations, for short). These auxiliary locations are added to disable an urgent transition after its upper bound has been reached, thus avoiding possible timelocks (fig.2). For instance, `E1` avoids the timelocks that would occur in `S0` if $v(\mathtt{y}) > 1$ and a `b?`-transition were enabled at that point. Figure 4 (right) shows the translation of eager actions with both lower and upper bounds, which requires (as expected) the use of lb- and ub-locations.

Committed locations are used when there are outgoing eager actions with lower bounds, and the source location may be entered with a valuation that satisfies the lower bound. For instance, if `R0` is entered when $v(\mathtt{x}) < 1$, the automaton must delay in `LB1` until `a!` can be offered; on the other hand, if `R0` is entered when $v(\mathtt{x}) \geq 1$, the automaton must offer `a!` immediately at `LB2`.

Figure 5 shows a more involved example, where there are many outgoing transitions in the same location. The lower bounds of eager actions `a!` and `b!` are accounted for by lb-locations `LB1` to `LB3`. The intermediate auxiliary transitions allow the TA to pass time, enabling and disabling actions depending on the current valuation. For instance, the purpose of the e-location `E1` is to disable `a!`, but it must allow the TA to offer `b!` eventually (hence, the TA may not remain in `E1` beyond $v(\mathtt{x}) = 3$).

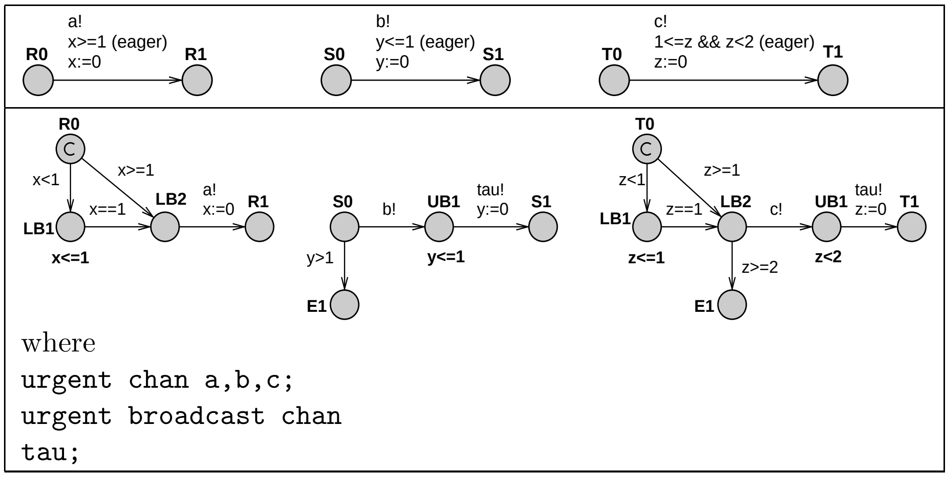
[4] The translation uses `tau!`-transitions to model urgent internal transitions in UPPAAL, where `tau` is a distinguished urgent broadcast channel that does not occur as an action label in the TAD network.



where

```
chan a;
urgent chan b;
urgent broadcast chan
tau;
```

Figure 3: Translating lazy actions and unbounded eager actions (left: TAD, right: TA)



where

```
urgent chan a,b,c;
urgent broadcast chan
tau;
```

Figure 4: Translating bounded eager actions (top: TAD, bottom: TA)

Also, the overlap between the guards of `c!` and `a!` is reflected in the TA by offering `c!` from all lb-locations where it could be enabled (`LB1` and `LB2`). In general, transitions that represent TAD actions have to be offered from all lb-locations and e-locations where they could be enabled.

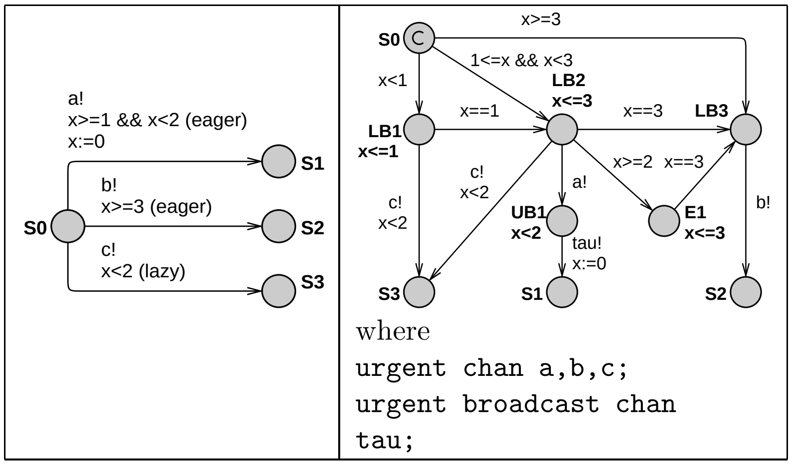
Note that, figs. 4 and 5 show that TAD models can be more concise than behaviorally equivalent TA models.

4.2 The translation, formally

Let $|A = \langle A_1, \ldots, A_n\rangle$ be a TAD network with lazy and eager actions. We define a compositional translation $\mathcal{T}$, s.t. $|B = \langle B_1, \ldots, B_n\rangle$, where $B_i = \mathcal{T}(A_i)$ for all $i : 1..n$, is an Uppaal TA network whose behavior is equivalent to $|A$.

Preliminaries. $|A = \langle A_1, \ldots, A_n\rangle$, where $A_i = (L_{A,i}, l_{A,i,0}, Lab_{A,i}, T_{A,i}, C_{A,i})$ $(i : 1..n)$. $T = \bigcup_{i=1}^{n} T_{A,i}$ and $L = \bigcup_{i=1}^{n} L_{A,i}$. T_{ea} and T_{la} are the sets of eager and lazy actions in $|A$ (resp.). $T_{\text{ea}}^{\text{lb}}, T_{\text{ea}}^{\text{ub}} \subseteq T_{\text{ea}}$ refer to lower-bounded and upper-bounded eager actions. Given $l \in L$, we use $T(l)$, $T_{\text{ea}}(l)$, $T_{\text{la}}(l)$, $T_{\text{ea}}^{\text{lb}}(l)$ and $T_{\text{ea}}^{\text{ub}}(l)$ to restrict the previous sets to the outgoing transitions in l, and we use x_l to denote the clock that guards all outgoing transitions in l.

Given $t \in T$, we assume $g(t) = x \in G$, with $G = [lb, ub)$ or $G = [lb, ub]$ for eager actions, and $G = (lb, ub)$, $G = [lb, ub)$ or $G = [lb, ub]$ for lazy actions ($lb \in \mathbb{N}$, $ub \in \mathbb{N} \cup \{\infty\}$). We use $g^l(t) = lb$ and $g^u(t) = ub$ to denote lower and upper bounds of t; and $x \in g(t)$ to denote that clock x occurs in



where
```
urgent chan a,b,c;
urgent broadcast chan
tau;
```

Figure 5: Translating multiple outgoing actions (left: TAD, right: TA)

$g(t)$. We say that t has an upper bound $(<, g^u(t))$ if $G = [g^l(t), g^u(t))$, $g^u(t) \in \mathbb{N}$ (resp. $(\leq, g^u(t))$ if $G = [g^l(t), g^u(t)]$, $g^u(t) \in \mathbb{N}$). Given an upper bound u, we define the interval $\iota(u)$ s.t. $\iota(u) = [0, ub)$ if $u = (<, ub)$ and $\iota(u) = [0, ub]$ if $u = (\leq, ub)$. We will use the following functions on sets of upper bounds.

$$\begin{aligned}
&\mathit{min}(U) = u \in U \text{ s.t. } \forall\, u' \in U.\ u \subseteq u' \\
&\mathit{next}(u, U) = u' \in U \text{ s.t. } u \subset u' \text{ and } \nexists\, u'' \in U.\ u \subset u'' \subset u'
\end{aligned}$$

TA transitions for eager and lazy actions. Action labels in half actions are represented in $|B$ by channels with the same name: labels in lazy actions with non-urgent channels and labels in eager actions with urgent channels. Lazy completed actions are represented by internal transitions. Eager completed actions are represented by `tau!`-transitions, where `tau` is a distinguished urgent broadcast channel that does not occur as a label in $|A$.

Let $t_j = (l, a_j, g_j, d_j, r_j, l') \in T$ $(j : 1..|T|)$, and loc be a location in $|B$. We define $\mathit{upp}(t_j, \mathit{loc})$ to be the set of TA transitions that represent t_j, when offered from loc.

$$\mathit{upp}(t_j, \mathit{loc}) = \begin{cases} \{\, (\mathit{loc}, \mathit{sync}_j, g_j, r_j, l') \,\} & \text{if } t_j \in T_{\text{la}} \\ \{\, (\mathit{loc}, \mathit{sync}_j, \mathit{true}, r_j, l') \,\} & \text{if } t_j \in T_{\text{ea}} \setminus T_{\text{ea}}^{\text{ub}} \\ \{\, (\mathit{loc}, \mathit{sync}_j, \mathit{true}, \emptyset, l^j_{\text{ub}}),\ (l^j_{\text{ub}}, \texttt{tau!}, \mathit{true}, r_j, l') \,\} & \text{if } t_j \in T_{\text{ea}}^{\text{ub}} \end{cases}$$

where $\mathit{sync}_j = \texttt{tau!}$ if t_j is an eager completed action, $\mathit{sync}_j = \tau$ if t_j is a lazy completed action and $\mathit{sync}_j = a_j$ otherwise.

Auxiliary locations and transitions. For any $l \in L$, the lower bounds of eager actions in l partition the reachable valuations of x_l in l into intervals:

$$\Gamma(l) = \{\, [0, \mathit{lb}_1), [\mathit{lb}_1, \mathit{lb}_2), \ldots, [\mathit{lb}_m, \infty) \,\}$$

where (1) $\mathit{lb}_k < \mathit{lb}_{k+1}$ for all $k : 1..(m-1)$; (2) $\forall\, k : 1..m.\ \exists\, t \in T_{\text{ea}}^{\text{lb}}(l).\ g^l(t) = \mathit{lb}_k$; and (3) $\forall\, t \in T_{\text{ea}}^{\text{lb}}(l).\ \exists\, k : 1..m.\ g^l(t) = \mathit{lb}_k$.

Let ι_k denote the k-th interval in the sequence, $[\mathit{lb}_{k-1}, \mathit{lb}_k)$ for $k : 1..(m+1)$ (by convention, $\mathit{lb}_0 = 0$ and $\mathit{lb}_{m+1} = \infty$). For any such interval ι, $T_{\text{offer}}(l, \iota) \subseteq T(l)$ is the set of transitions in l that are enabled in ι.

$$T_{\text{offer}}(l, \iota) = \{\, t \in T(l) \mid g(t) = x_l \in G,\ G \cap \iota \neq \emptyset \,\}$$

Let $T^{\text{ub}}_{\text{offer}}(l, \iota) \subseteq T_{\text{offer}}(l, \iota) \cap T^{\text{ub}}_{\text{ea}}(l)$ be the subset of eager actions offered during ι, which have an upper bound in ι.

$$T^{\text{ub}}_{\text{offer}}(l, \iota) = \{\ t \in T_{\text{offer}}(l, \iota) \cap T^{\text{ub}}_{\text{ea}}(l) \mid g^u(t) \in \iota\ \}$$

Let $UB(l, \iota)$ denote the set of all different upper bounds of actions in $T^{\text{ub}}_{\text{offer}}(l, \iota)$.

Locations in $|A$ will be represented in $|B$ by committed locations with the same name (unless $T^{\text{lb}}_{\text{ea}}(l) = \emptyset$). In addition, for each interval ι_k, $k : 1..|\Gamma(l)|$, the translation will generate the following auxiliary locations.

1. one lb-location, l^k_{lb};
2. one ub-location, l^j_{ub} (for each $t_j \in T^{\text{ub}}_{\text{ea}}(l)$, $j : 1..|T|$); and
3. one e-location, $l^{k,u}_{\text{esc}}$ (for each $u \in UB(l, \iota_k)$).

The translation will also generate the following auxiliary transitions.

1. from l to l^k_{lb},
2. from l^k_{lb} to l^{k+1}_{lb},
3. from l^k_{lb} to $l^{k,u}_{\text{esc}}$ (for $u = min(UB(l, \iota_k))$),
4. from $l^{k,u}_{\text{esc}}$ to $l^{k,u'}_{\text{esc}}$ (for $u' = next(u, UB(l, \iota_k))$); and
5. from $l^{k,u}_{\text{esc}}$ to l^{k+1}_{lb} (for each $u \in UB(l, \iota_k)$).

Finally, from every lb-location and e-location (and from the initial source location, if this is not a committed location), the translation will generate equivalent TA transitions to map the eager and lazy actions that may be enabled in the associated intervals (using the function *upp*()).

Translating components. For each TAD $A_i = (L_{A,i}, l_{A,i,0}, Lab_{A,i}, T_{A,i}, C_{A,i})$ $(i : 1..n)$, the translation generates the TA B_i, which is defined as follows.

$$B_i = \mathcal{T}(A_i) = (L_{B,i}, l_{B,i,0}, Lab_{B,i}, T_{B,i}, I_{B,i}, C_{B,i})$$

where:

- $L_{B,i} = L_{A,i} \ \cup\ L^{\text{lb}}_{A,i} \ \cup\ L^{\text{esc}}_{A,i} \ \cup\ L^{\text{ub}}_{A,i}$, where

$$\begin{aligned} L^{\text{lb}}_{A,i} &= \{\, l^k_{\text{lb}} \mid l \in L_{A,i},\ k : 1..|\Gamma(l)| \,\} \\ L^{\text{esc}}_{A,i} &= \{\, l^{k,u}_{\text{esc}} \mid l \in L_{A,i},\ k : 1..|\Gamma(l)|,\ \iota_k \in \Gamma(l),\ u \in UB(l, \iota_k) \,\} \\ L^{\text{ub}}_{A,i} &= \{\, l^j_{\text{ub}} \mid l \in L_{A,i},\ j : 1..|T|,\ t_j \in T^{\text{ub}}_{\text{ea}}(l) \,\} \end{aligned}$$

- $L^{\text{u}}_{B,i} = \emptyset$, $L^{\text{c}}_{B,i} = \{l \in L_{A,i} \mid T^{\text{lb}}_{\text{ea}}(l) \neq \emptyset\}$
- $l_{B,i,0} = l_{A,i,0}$
- $Lab_{B,i}$ is the smallest set of labels that satisfies the following conditions.

$$\begin{array}{ll} a \in Lab_{B,i} \cap (Ch^{\text{bin}} \setminus Ch^{\text{urg}}) & \text{if } \exists\, t \in T_{\text{la}} \cap HA.\ lab(t) = a \\ u \in Lab_{B,i} \cap Ch^{\text{bin}} \cap Ch^{\text{urg}} & \text{if } \exists\, t \in T_{\text{ea}} \cap HA.\ lab(t) = u \\ \texttt{tau} \in Lab_{B,i} \cap Ch^{\text{brd}} \cap Ch^{\text{urg}} & \text{if } T_{\text{ea}} \cap CA \neq \emptyset \\ \tau \in Lab_{B,i} & \text{if } T_{\text{la}} \cap CA \neq \emptyset \end{array}$$

- $C_{B,i} = C_{A,i}$
- $I_{B,i}$ is the invariant function, which is defined as follows.

$$\begin{array}{lll}
I_{B,i}(l) & = \mathit{true} & \text{if } \ l \in L_{A,i} \setminus L^{\mathrm{c}}_{B,i} \\
I_{B,i}(l^k_{\mathrm{lb}}) & = x_l \le lb_k & \text{if } \ l \in L_{A,i}, \ k : 1..|\Gamma(l)|, \ [lb_{k-1}, lb_k) \in \Gamma(l) \\
I_{B,i}(l^{k,u}_{\mathrm{esc}}) & = I_{B,i}(l^k_{\mathrm{lb}}) & \text{if } \ l \in L_{A,i}, \ k : 1..|\Gamma(l)|, \ \iota_k \in \Gamma(l), \ u \in UB(l, \iota_k) \\
I_{B,i}(l^j_{\mathrm{ub}}) & = x_j < g^u_j & \text{if } \ l \in L_{A,i}, \ j : 1..|T|, \ g_j = [g^l_j, g^u_j), \ g^u_j \in \mathbb{N} \\
I_{B,i}(l^j_{\mathrm{ub}}) & = x_j \le g^u_j & \text{if } \ l \in L_{A,i}, \ j : 1..|T|, \ g_j = [g^l_j, g^u_j], \ g^u_j \in \mathbb{N}
\end{array}$$

- $T_{B,i}$ is the smallest set of transitions that satisfies the following conditions.

$$\begin{array}{lll}
(l, \tau, x_l \in \iota_k, \emptyset, l^k_{\mathrm{lb}}) \in T_{B,i} & \text{if} & l \in L_{A,i}, \ k : 1..|\Gamma(l)| \\
(l^k_{\mathrm{lb}}, \tau, x_l = lb_k, \emptyset, l^{k+1}_{\mathrm{lb}}) \in T_{B,i} & \text{if} & l \in L_{A,i}, \ k : 1..(|\Gamma(l)| - 1) \\
(l^k_{\mathrm{lb}}, \tau, x_l \in \iota_k \setminus \iota(u), \emptyset, l^{k,u}_{\mathrm{esc}}) \in T_{B,i} & \text{if} & l \in L_{A,i}, \ k : 1..|\Gamma(l)|, \ \iota_k \in \Gamma(l), \\
 & & u = \mathit{min}(UB(l, \iota_k)) \\
(l^{k,u}_{\mathrm{esc}}, \tau, x_l \in \iota_k \setminus \iota(u), \emptyset, l^{k,u'}_{\mathrm{esc}}) \in T_{B,i} & \text{if} & l \in L_{A,i}, \ k : 1..|\Gamma(l)|, \ \iota_k \in \Gamma(l), \\
 & & u \in UB(l, \iota_k), \\
 & & u' = \mathit{next}(u, UB(l, \iota_k)) \\
(l^{k,u}_{\mathrm{esc}}, \tau, x_l = lb_k, \emptyset, l^{k+1}_{\mathrm{lb}}) \in T_{B,i} & \text{if} & l \in L_{A,i}, \ (k : 1..|\Gamma(l)| - 1), \\
 & & \iota_k \in \Gamma(l), u \in UB(l, \iota_k) \\
\mathit{upp}(t, l^k_{\mathrm{lb}}) \subseteq T_{B,i} & \text{if} & l \in L_{A,i}, \ k : 1..|\Gamma(l)|, \ \iota_k \in \Gamma(l), \\
 & & t \in T_{\mathrm{offer}}(l, \iota_k) \\
\mathit{upp}(t, l^{k,u}_{\mathrm{esc}}) \subseteq T_{B,i} & \text{if} & l \in L_{A,i}, \ k : 1..|\Gamma(l)|, \ \iota_k \in \Gamma(l), \\
 & & u \in UB(l, \iota_k), \ t \in T_{\mathrm{offer}}(l, \iota_k), \\
 & & g(t) = x_l \in G, \ G \cap (\iota_k \setminus \iota(u)) \neq \emptyset \\
\mathit{upp}(t, l) \subseteq T_{B,i} & \text{if} & l \in L_{A,i}, \ t \in T(l), \ T^{\mathrm{lb}}_{\mathrm{ea}}(l) = \emptyset
\end{array}$$

The following theorems state the correctness and complexity of the translation (detailed proofs can be found in Appendix B).

THEOREM **4.1. (Bisimulation)** *Let $|A$ be a TAD network and $|B = \mathcal{T}(|A)$ the resulting TA network. There exists a strongly timed bisimulation [10] between $|A$ and $|B$, which abstracts over auxiliary TA transitions.*

THEOREM **4.2. (Linear complexity)** *Let $|A$ be a TAD network and $|B = \mathcal{T}(|A)$ the resulting TA network. The size of $|B$ is proportional to the size of $|A$.*

4.3 Verification

Correctness properties on the TAD network can be expressed as queries written in UPPAAL's requirement language [4] (a subset of TCTL [16]). Equivalent properties on the TA network are obtained by substitution of equivalent locations in the original query. This is defined as follows.

Let $|A = \langle A_1, \ldots, A_n \rangle$ be a TAD network and $|B = \langle B_1, \ldots, B_n \rangle$ the resulting TA network. Substitution of equivalent locations is given by the function θ, [5]

$$\begin{array}{rl}
\theta(A_i.l) = \bigvee(& \{ B_i.l \} \cup \\
 & \{ B_i.l^k_{\mathrm{lb}} \mid k : 0..|\Gamma(l)| \} \cup \\
 & \{ B_i.l^{k,u}_{\mathrm{esc}} \mid k : 1..|\Gamma(l)|, \ u \in UB(l, \iota_k) \} \cup \\
 & \{ B_i.\mathit{loc}^j_{\mathrm{ub}} \mid (\mathit{loc}^j_{\mathrm{ub}}, \texttt{tau!}, \mathit{true}, r_j, l) \in T_{B,i}, \ \mathit{loc} \in L_{A,i}, \ j : 1..|T| \})
\end{array}$$

[5]The state formula $A_i.l$ holds whenever A_i is currently in location l.

THEOREM **4.3.** *Let $|A$ be a TAD network and $|B = \mathcal{T}(|A)$ the resulting TA network. For any formula F in* UPPAAL*'s requirement language, the following holds.*

$$|A \models F \quad \textit{iff} \quad |B \models F\left[\theta(A_i.l)/A_i.l\right]_{A_i.l \in F}$$

Proof. Follows from strong timed-bisimulation between $|A$ and $|B$, and preservation of TCTL properties by strong-timed bisimulation [21]. □

For instance, the property on the TAD network of Fig. 5, *is action* `b!` *eventually enabled?*, can be expressed as $F = \exists\Diamond(\mathtt{S0} \wedge \mathtt{x} \geq 3)$. The equivalent property on the TA network is then $F' = \exists\Diamond((\mathtt{S0} \vee \mathtt{LB1} \vee \mathtt{LB2} \vee \mathtt{LB3} \vee \mathtt{E1}) \wedge \mathtt{x} \geq 3)$.

4.4 The TAD2TA tool

We implemented a tool that receives a TAD network and generates the equivalent TA network, which can be displayed, simulated and verified in UPPAAL. TAD networks themselves can be built using UPPAAL's GUI and most of its modeling facilities (e.g., parameterized templates and data variables). Correctness properties can be expressed in UPPAAL's requirements language, which the tool then translates to equivalent formulae in terms of the TA network (§ 4.3). Diagnostic traces are presented in terms of the TA network, and the tool generates mappings to identify the original locations and transitions in the TAD network (the integration with UPPAAL's simulator is subject of ongoing work).

We tested the tool on a number of TAD models, which we constructed from TA models of academic examples (e.g., Fisher's mutex protocol and the train-gate problem) and case studies such as [9, 23, 14, 22, 18, 15]. In general, we obtained TAD models that were conceptually simpler than the originals (with the guarantee of being time-reactive by construction). In same cases, the TAD models were also more faithful to the system (e.g., the behavior of some timeouts constructs, which could only be approximated in the TA model of [9], was represented exactly in the TAD model). On the other hand, the design of TA models may benefit from features of UPPAAL that are not supported by our translation. For instance, TAD models cannot easily express the interleaving semantics of committed locations, and our translation disallows variables in clock constraints.

Table 1 compares the size of the TAD models with that of the generated UPPAAL TA networks (translation times were negligible). The tool implements a number of optimizations that avoid the generation of redundant auxiliary locations and transitions (e.g., eager actions in location l, which are guarded on clock x, admit simpler translations if x is reset whenever l is entered). In addition, graph layout algorithms were implemented to help the visualization of generated TA networks. More details on the tool's architecture, its input language, and the models listed in Table 1, can be found in the tool's website.[6]

5 Multiple outgoing eager actions guarded on different clocks

This section presents a translation from Sparse TAD to TA networks in Uppaal, for the case where actions in the same location may be guarded on different clocks (assuming, however, that all other syntactic restrictions apply). This is rarely the case in practice, but we want to show how the translation of § 4 can be extended to deal with more complex TAD models. The translation presented here, $\mathcal{T}_{\mathrm{mclocks}}$, extends the definition of $\mathcal{T}$ (§ 4) to deal with multiple clocks.

For example, fig 6 (left) shows a fragment of a TAD component with two eager half actions from `L1`, `a!` and `b!`, with guards $\mathtt{x} \in [1, 2)$ and $\mathtt{y} \in [3, 4)$. There is not enough information to determine

[6]`http://www.cs.kent.ac.uk/people/staff/rsg5/TAD2TAtool/TAD2TAwpage.htm`

Table 1: Translation of TAD models to UPPAAL TA networks. $|L|$ ($|L'|$) and $|T|$ ($|T'|$) denote the number of locations and transitions in the input (output) model (resp.).

TAD model	($\|L\|$, $\|T\|$)	($\|L'\|$, $\|T'\|$)	($\|L'\|/\|L\|$, $\|T'\|/\|T\|$)
gbox_tad.xml	(65, 84)	(78,102)	(1.20, 1.21)
bocdpFIXED_tad.xml	(70, 130)	(126, 189)	(1.8, 1.45)
csmacd_kronos_tad.xml	(9, 18)	(27, 46)	(3, 2.56)
fischer_tad.xml	(4, 5)	(4, 5)	(1, 1)
lipsync_tad.xml	(50, 53)	(82, 97)	(1.64, 1.83)
train-gate-410_tad.xml	(10, 13)	(15, 21)	(1.5, 1.62)
zeroconffull2007_tad.xml	(14, 25)	(37, 66)	(2.64, 2.75)
2doors_tad.xml	(8, 11)	(11, 16)	(1.38, 1.45)
bmp_tad.xml	(19, 30)	(27, 51)	(1.42, 1.7)
windcar_tad.xml	(7, 9)	(7, 9)	(1, 1)
WSN_tad.xml	(16, 19)	(18, 26)	(1.13, 1.37)
interrupt_tad.xml	(8, 10)	(8, 10)	(1, 1)
bridge_tad.xml	(8, 9)	(8, 11)	(1, 1.22)

v(`x`) at `L1`	v(`y`) at `L1`	Offer `a!`	Offer `b!`	Location in the TA component
< 1	< 3	×	×	`LB1`
≥ 1	< 3	✓	×	`LB2`
< 1	≥ 3	×	✓	`LB3`
≥ 1	≥ 3	✓	✓	`LB4`

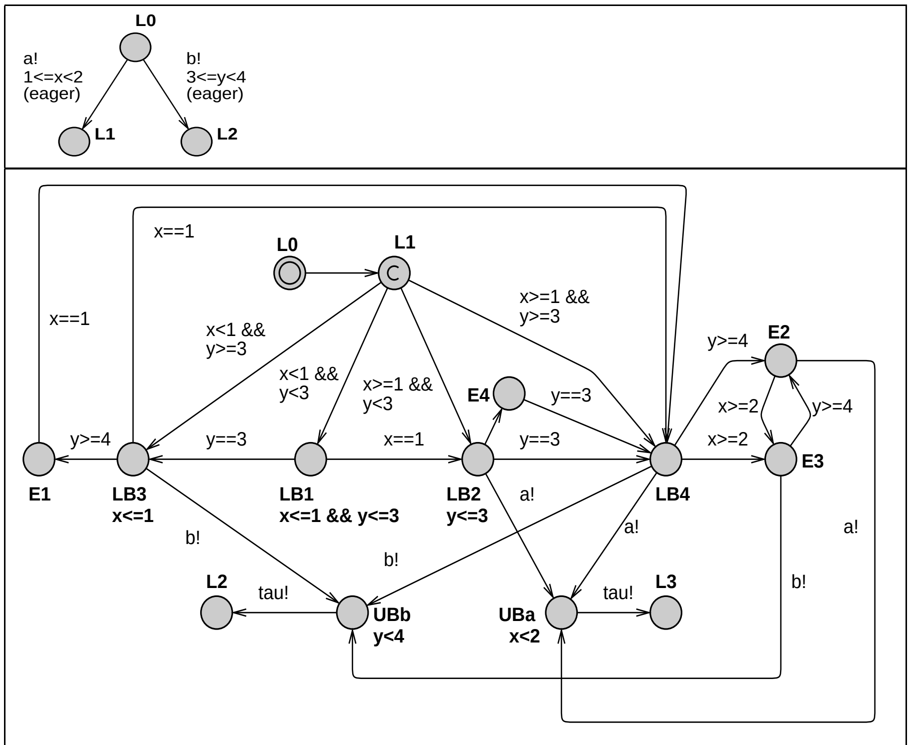
Table 2: Relative valuations to consider in the translation of the TAD shown in Fig. 6 (left). ✓: the transition must be offered. ×: the transition must not be offered. - : the location is not shown.

the relation between the values of `x` and `y` when `L1` is entered, hence the translation must consider all possibilities in order to offer the appropriate urgent transition, `a!` or `b!`, at the right time. Table 2 shows the different possible valuations when `L1` is entered. Outgoing actions are grouped w.r.t. the clocks occurring in the their guards. For each group, the lower bounds of eager actions induce a partition of the time-line into a sequence of consecutive intervals. Finally, the set of valuations at `L1` is partitioned according to all possible combinations of intervals (one for each clock). Every such combination is represented in the TA by an auxiliary location (fig 6, right). As usual, we consider lb-locations to enforce eager actions' lower-bounds, ub-locations to enforce eager actions' upper bounds and escape locations to avoid timelocks due to urgent transitions.

5.1 Formal Definition

Let $|A = \langle A_1, \ldots, A_n \rangle$ be a TAD network. We define a translation, $\mathcal{T}_{\text{mclocks}}$, which takes any component TAD, A_i, $i : 1..n$, and produces an equivalent TA, $B = \mathcal{T}_{\text{mclocks}}(A)$. Notations and definitions will be borrowed from § 4, unless stated otherwise.

Preeliminaries. We assume that $|A$ satisfies the restrictions enumerated in § 4, except that outgoing transitions at the same location can be guarded on different clocks (although, we assume that at most one clock occurs in any guard). Let $C(l) = \{x_1, \ldots, x_{|C(l)|}\}$ be the set of clocks referred to by actions in location l.



where:

```
urgent chan a, b;
urgent broadcast chan tau;
```

Figure 6: Combinatorial explosion due multiple outgoing eager actions on different clocks (top: TAD, bottom: TA)

The lower bounds of eager actions in l partition the reachable values of each $x \in C(l)$ into a set

$$\Gamma(l, x) = \{[0, lb_1), [lb_1, lb_2), \ldots, [lb_m, \infty)\}$$

where (1) $lb_k < lb_{k+1}$ for all $k : 1..(m-1)$; (2) $\forall\, k : 1..m.\ \exists\, t \in T^{\text{lb}}_{\text{ea}}(l).\ g^l(t) = lb_k$; and (3) $\forall\, t \in T^{\text{lb}}_{\text{ea}}(l).\ \exists\, k : 1..m.\ g^l(t) = lb_k$.

Let $UB(l, \iota, x)$ be the set of distinct upper bounds of actions in $T^{\text{lb}}_{\text{ea}}(l)$, whose guards refer to $x \in C(l)$ and are satisfied in interval ι. Let $s(l) = |C(l)|$ and $s(l, h) = |\Gamma(l, x_h)|$ $(h : 1..s(l))$. We define,

$$\begin{array}{ll} \mathit{indices}(l) & = \{\ (k_1, \ldots, k_{s(l)}) \mid\ \ [lb_{k_h - 1}, lb_{k_h}) \in \Gamma(l, x_h), \\ & \qquad\qquad\qquad\qquad\ \ k_h : 1..|s(l, x_h)|,\ \ x_h \in |C(l)|,\ \ h : 1..s(l)\ \} \end{array}$$

For any tuple of indices $\kappa = (k_1, \ldots, k_r)$ and $h : 1..r$, we define $\kappa_h = k_h$ (the h-th element of κ) and $\iota(\kappa, h) = [lb_{\kappa_h - 1}, lb_{\kappa_h})$ (the unique interval corresponding to κ_h). We also define the set of actions which are offered at any ι_h,

$$T_{\text{offer}}(l, \kappa) = \{t \in T \mid g(t) = x_h \in G,\ \ G \in \iota(\kappa, h), h : 1..|s(l)|\ \}$$

Let $T^{\text{ub}}_{\text{offer}}(l, \iota, x)$ be the subset of urgent actions in l that are guarded on clock x and have an upper bound in interval ι.

$$T^{\text{ub}}_{\text{offer}}(l, \iota, x) = \{\ t \in T_{\text{ea}}(l) \mid g^u(t) \in \iota\ \}$$

As in § 4, the translation generates lb-locations to enforce eager actions's lower bounds, ub-locations to enforce eager actions' upper bounds and escape locations to avoid timelocks due to urgent transitions. These auxiliary locations, plus the necessary interconnecting auxiliary transitions, are formalized below.

For each TAD $A_i = (L_{A,i}, l_{A,i,0}, \mathit{Lab}_{A,i}, T_{A,i}, C_{A,i})$ $(i : 1..n)$, the translation generates the TA B_i, which is defined as follows.

$$B_i = \mathcal{T}(A_i) = (L_{B,i}, l_{B,i,0}, \mathit{Lab}_{B,i}, T_{B,i}, I_{B,i}, C_{B,i})$$

where

- $L_{B,i} = L_{A,i}\ \cup\ L^{\text{lb}}_{A,i}\ \cup\ L^{\text{esc}}_{A,i}\ \cup\ L^{\text{ub}}_{A,i}$, where

$$\begin{array}{ll} L^{\text{lb}}_{A,i} & = \{\ l^{\kappa}_{\text{lb}} \mid l \in L_{A,i},\ \ \kappa \in \mathit{indices}(l)\} \\ L^{\text{esc}}_{A,i} & = \{\ l^{\kappa,x,u}_{\text{esc}} \mid l \in L_{A,i},\ \ \kappa \in \mathit{indices}(l),\ \ u \in UB(l, \iota(\kappa, h), x),\ \ x \in C(l)\ \} \\ L^{\text{ub}}_{A,i} & = \{\ l^{j}_{\text{ub}} \mid l \in L_{A,i},\ \ t_j \in T^{\text{ub}}_{\text{ea}}(l),\ \ j : 1..|T|\ \} \end{array}$$

- $L^{\text{u}}_{B,i} = \emptyset$, $L^{\text{c}}_{B,i} = \{l \in L_{A,i} \mid T^{\text{lb}}_{\text{ea}}(l) \neq \emptyset\}$
- $l_{B,i,0} = l_{A,i,0}$
- $\mathit{Lab}_{B,i}$ is the smallest set of labels that satisfies the following conditions.

$$\begin{array}{ll} a \in \mathit{Lab}_{B,i} \cap (\mathit{Ch}^{\text{bin}} \setminus \mathit{Ch}^{\text{urg}}) & \text{if}\ \ \exists\, t \in T_{\text{la}} \cap \mathit{HA}.\ \mathit{lab}(t) = a \\ u \in \mathit{Lab}_{B,i} \cap \mathit{Ch}^{\text{bin}} \cap \mathit{Ch}^{\text{urg}} & \text{if}\ \ \exists\, t \in T_{\text{ea}} \cap \mathit{HA}.\ \mathit{lab}(t) = u \\ \texttt{tau} \in \mathit{Lab}_{B,i} \cap \mathit{Ch}^{\text{brd}} \cap \mathit{Ch}^{\text{urg}} & \text{if}\ \ T_{\text{ea}} \cap \mathit{CA} \neq \emptyset \\ \tau \in \mathit{Lab}_{B,i} & \text{if}\ \ T_{\text{la}} \cap \mathit{CA} \neq \emptyset \end{array}$$

- $C_{B,i} = C_{A,i}$

- $I_{B,i}$ is the invariant function, which is defined as follows.

$$\begin{array}{lll}
I_{B,i}(l) & = \mathit{true} & \text{if } l \in L_{A,i} \setminus L^{\mathrm{c}}_{B,i} \\
I_{B,i}(l^{\kappa}_{\mathrm{lb}}) & = \bigwedge_{h=1}^{s(l)} x_h \leq lb_{k_h} & \text{if } l^{\kappa}_{\mathrm{lb}} \in L_{B,i},\ x_h \in C(l),\ \iota(\kappa,h) = [lb_{k_h-1}, lb_{k_h}) \\
I_{B,i}(l^{\kappa,x,u}_{\mathrm{esc}}) & = I_{B,i}(l^{\kappa}_{\mathrm{lb}}) & \text{if } l^{\kappa}_{\mathrm{lb}}, l^{\kappa,x,u}_{\mathrm{esc}} \in L_{B,i} \\
I_{B,i}(l^{j}_{\mathrm{ub}}) & = x_j < g^u_j & \text{if } l^{j}_{\mathrm{ub}} \in L_{B,i},\ g_j = [g^l_j, g^u_j),\ g^u_j \in \mathbb{N} \\
I_{B,i}(l^{j}_{\mathrm{ub}}) & = x_j \leq g^u_j & \text{if } l^{j}_{\mathrm{ub}} \in L_{B,i},\ g_j = [g^l_j, g^u_j],\ g^u_j \in \mathbb{N}
\end{array}$$

- $T_{B,i}$ is the smallest set of transitions that satisfies the following conditions.

1. $l \xrightarrow{\tau,\ \bigwedge_{h=1}^{s(l)} x_h \in \iota(\kappa,h),\ \emptyset} l^{\kappa}_{\mathrm{lb}} \in T_{B,i}$ — for all $l, l^{\kappa}_{\mathrm{lb}} \in L_{B,i}$

2. $l^{\kappa}_{\mathrm{lb}} \xrightarrow{\tau,\ x_h = lb_{k_h},\ \emptyset} l^{\kappa'}_{\mathrm{lb}} \in T_{B,i}$ — for all $l^{\kappa}_{\mathrm{lb}}, l^{\kappa'}_{\mathrm{lb}} \in L_{B,i},\ \iota(\kappa,h) = [lb_{k_h-1}, lb_{k_h})$
$\kappa = (k_1, \ldots, k_h, \ldots, k_r)$,
$\kappa' = (k_1, \ldots, k_h + 1, \ldots, k_r)$

3. $\mathit{upp}(t, l^{\kappa}_{\mathrm{lb}}) \subseteq T_{B,i}$ — for all $l^{\kappa}_{\mathrm{lb}} \in L_{B,i},\ t \in T_{\mathrm{offer}}(l,\kappa)$

4. $l^{\kappa}_{\mathrm{lb}} \xrightarrow{\tau,\ x_h \in \iota(\kappa,h) \setminus \iota(u),\ \emptyset} l^{\kappa,x_h,u}_{\mathrm{esc}} \in T_{B,i}$ — for all $l^{\kappa}_{\mathrm{lb}}, l^{\kappa,u}_{\mathrm{esc}} \in L_{B,i}$,
$u = \mathit{min}(\mathit{UB}(l, \iota(\kappa,h), x_h))$,
$h : 1..|s(l)|$

5. $l^{\kappa,x_h,u}_{\mathrm{esc}} \xrightarrow{\tau,\ x_h \in \iota(\kappa,h) \setminus \iota(u),\ \emptyset} l^{\kappa,x_h,u'}_{\mathrm{esc}} \in T_{B,i}$ — for all $l^{\kappa}_{\mathrm{lb}}, l^{\kappa,x_h,u}_{\mathrm{esc}} \in L_{B,i}$,
$u' = \mathit{next}(u, \mathit{UB}(l, \iota(\kappa,h), x_h))$,
$h : 1..|s(l)|$

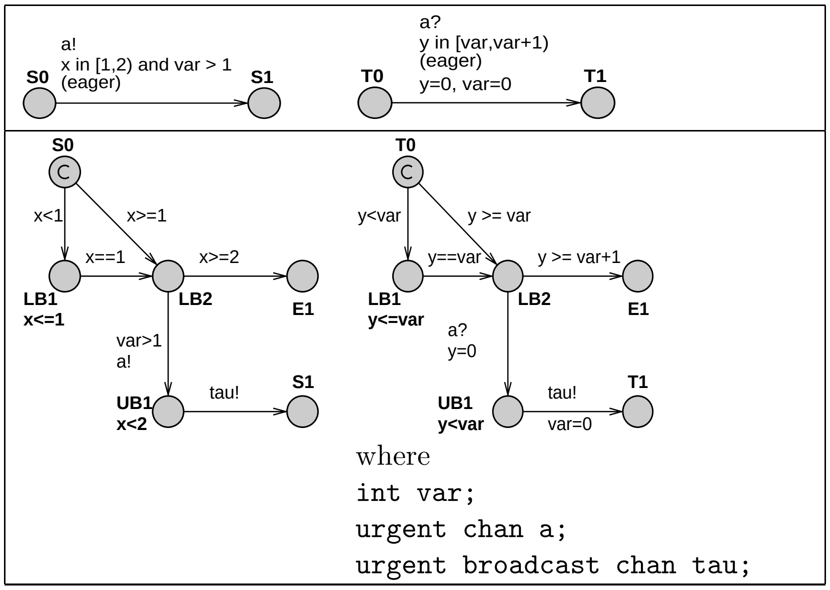
6. $l^{\kappa,x_h,u}_{\mathrm{esc}} \xrightarrow{\tau,\ x_h = lb_{k_h},\ \emptyset} l^{\kappa'}_{\mathrm{lb}} \in T_{B,i}$ — for all $l^{\kappa,x_h,u}_{\mathrm{esc}}, l^{\kappa'}_{\mathrm{lb}} \in L_{B,i}$,
$\iota(\kappa,h) = [lb_{k_h-1}, lb_{k_h})$,
$\kappa = (k_1, \ldots, k_h, \ldots, k_r)$,
$\kappa' = (k_1, \ldots, k_h + 1, \ldots, k_r)$

7. $\mathit{upp}(t, l^{\kappa,x_h,u}_{\mathrm{esc}}) \subseteq T_{B,i}$ — for all $l^{\kappa,x_h,u}_{\mathrm{esc}} \in L_{B,i}$,
$t \in T_{\mathrm{offer}}(l,\kappa)$,
$g(t) = x_q \in G,\ G \cap \iota(\kappa,q) \neq \emptyset$,
$h, q : 1..|s(l)|$

8. $\mathit{upp}(t, l) \subseteq T_{B,i}$ — for all $t \in T(l)$ and $l \in L_{A,i}$ s.t. $T_{\mathrm{ea}}(l) = \emptyset$

5.2 Correctness and Complexity

The translation $\mathcal{T}_{\mathrm{mclocks}}$ generates TA networks that are strongly timed bisimilar to the original TAD networks (abstracting over the auxiliary transitions introduced by the translation). However, in this case, the resulting TA networks may be exponentially bigger than the input TAD networks. We omit the proofs of these claims, which can be extrapolated from those in Appendix B.



```
int var;
urgent chan a;
urgent broadcast chan tau;
```

Figure 7: A translation of eager actions guarded on shared variables (top:TAD, bottom:TA)

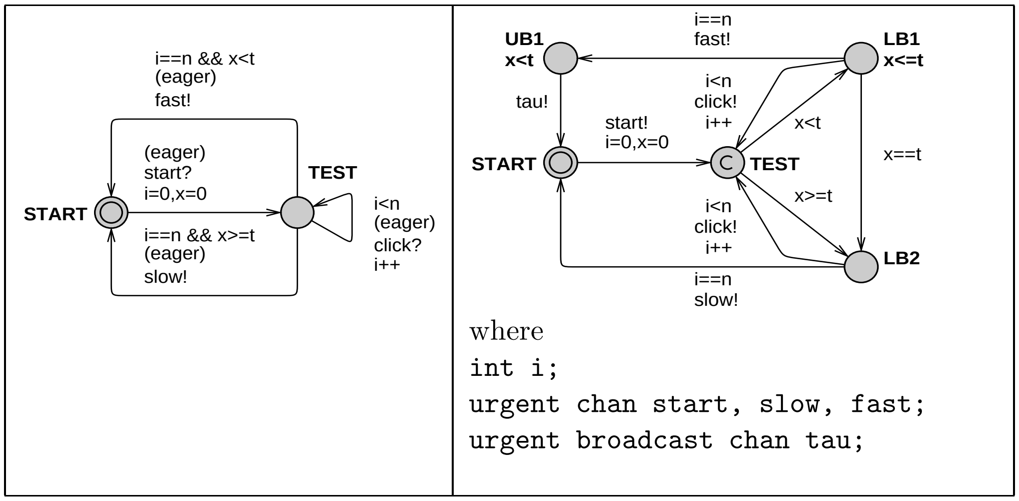
6 Data variables and diagonal constraints

This paper has focused on simple TAD models, where communication is achieved through synchronization and clock constraints refer to single clocks. However, it is not difficult to deal with richer TAD models where actions may be guarded with shared variables and diagonal constraints (and where shared variables can be updated as a side effect of actions). This can be done as follows.

Accommodating data variables. Uppaal allows data constraints on urgent (and non-urgent) transitions, hence the translation can accommodate guards that involve both clock and data constraints at no extra cost. As we did for shared clocks in § 4, we require that any data variable that is updated in some component cannot occur in the guard of any eager action of other components. The translation is driven entirely by clock constraints on eager actions (or lazy actions with eager matches). Note that, whenever the guard of an eager action can be expressed as a conjunction $g = \phi_c \wedge \phi_d$, where ϕ_c is a clock constraint (where data variables may occur) and ϕ_d is a data constraint (where clock variables do not occur), ϕ_d may simply be copied as the guard of every urgent transition that represents the eager action.

This is illustrated in fig. 7. The `S`-component includes the eager action `a?`, which is guarded on clock `x` and shared (integer) variable `var`. Given that `var` does not constrain `x`, the translation is driven only by the term `x in [1,2)` and `var>1` is simply added as a guard in the urgent transition `a!`. Compare this with the translation of eager action `a?` in the `T`-component. Here `var` bounds the clock `y` and therefore occurs in invariants and guards of auxiliary locations and transitions. The translation of both eager actions in components `S` and `T` follows the pattern described in § 4.

As another example of the use of data variables on eager actions, fig.8 shows a TAD model of a click-speed test. The test is meant to determine the relative speed of a person clicking a mouse. The subject is considered to be fast (signaled with `fast!`) if `n` clicks occur in less than `t` time-units, otherwise the subject is considered to be slow (signaled with `slow!`). Again, the translation is driven by the clock constraints on eager actions (`x<t` and `x>=t`). As data constraints (`i<n` and `i==n`) do not bound the clock `x`, the translation simply attaches these constraints as guards of urgent transitions.



where

```
int i;
urgent chan start, slow, fast;
urgent broadcast chan tau;
```

Figure 8: A speed test (left:TAD, right:TA)

Accommodating diagonal constraints. The translation can be applied to actions with diagonal constraints by considering clock differences as a single clock. For instance, a constraint such as $x - y > 1$ can be interpreted as $z > 1$, where z is a fresh clock. Then, the TAD network is translated and z is substituted with $x - y$ in the resulting TA network, as a last step.

7 Conclusions and Future Work

We presented a compositional translation of TAD networks to Uppaal TA networks. The generated TA networks are strongly timed-bisimilar to the TAD networks and exhibit a worst-case linear increase in size. We described an implementation that allows Uppaal to aid the design and verification of TAD networks. Compositionality was achieved thanks to the asap-synchronization semantics of Uppaal's urgent channels. In addition, the strong invariant interpretation adopted in Uppaal proved necessary to faithfully express urgent actions with upper bounds. Asap-synchronization can also be obtained over non-urgent channels, although not as concisely (e.g., urgent channels can be expressed by non-urgent channels with the addition of shared Boolean variables and conditional invariants). Nonetheless, at least in principle, similar compositional translations could be obtained for other model-checkers for TA.

As future work, we would like to obtain further integration between our TAD2TA tool and Uppaal's GUI. In particular, we would like to extend Uppaal's simulator to work directly on the input TAD network (currently, the user may only simulate the equivalent TA network). In addition, it would be interesting to compare the syntactic translation of TAD to TA, as presented in this paper, against a direct implementation of time-constrained urgent transitions on TA (the support for clock-constraints on urgent transitions may be available in forthcoming Uppaal versions).

Acknowledgements. We are grateful to Howard Bowman, John Håkansson, Frits Vaandrager and members of his research group at Radboud University Nijmegen, and Kim Larsen and members of his research group at Aalborg University, for fruitful discussions on this work.

References

[1] L. Aceto, P. Bouyer, A. Burgueño, and K. Larsen. The power of reachability testing for timed automata. *Theoretical Computer Science*, 1-3(300):411–475, 2003.

[2] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[3] R. Barbuti and L. Tesei. Timed automata with urgent transitions. *Acta Informatica*, 40(5), March 2004.

[4] G. Behrmann, A. David, and K. Larsen. A tutorial on Uppaal. In *SFM-RT 2004*, LNCS 3185, pages 200–236. Springer, 2004.

[5] H. C. Bohnenkamp, H. Hermanns, and J-P. Katoen. MOTOR: The MODEST tool environment. In *TACAS'07*, LNCS 4424, pages 500–504. Springer, 2007.

[6] S. Bornot and J. Sifakis. On the composition of hybrid systems. In *Hybrid Systems: Computation and Control*, LNCS 1386, pages 49–63. Springer, 1998.

[7] S. Bornot, J. Sifakis, and S. Tripakis. Modeling urgency in timed systems. In *Proc. of COMPOS 1997*, LNCS 1536, pages 103–129. Springer, 1998.

[8] H. Bowman. Time and action lock freedom properties for timed automata. In *Proceedings of FORTE 2001*, pages 119–134. Kluwer Academic, 2001.

[9] H. Bowman, G. Faconti, J.-P. Katoen, D. Latella, and M. Massink. Automatic verification of a lip synchronization protocol using UPPAAL. *Formal Aspects of Computing*, 10(5-6):550–575, August 1998.

[10] H. Bowman and R. Gomez. *Concurrency Theory, Calculi and Automata for Modelling Untimed and Timed Concurrent Systems.* Springer, January 2006.

[11] H. Bowman and R. Gomez. How to stop time stopping. *Formal Aspects of Computing*, 18(4):459–493, December 2006.

[12] M. Bozga, S. Graf, Ileana Ober, Iulian Ober, and J. Sifakis. The IF toolset. In *SFM-RT 2004*, LNCS 3185, pages 237–267. Springer, 2004.

[13] P. D'Argenio, H. Hermanns, J-P. Katoen, and R. Klaren. MoDeST - a modelling and description language for stochastic timed systems. In *Proc. of PAPM-PROBMIV 2001*, LNCS 2165, pages 87–104. Springer, 2001.

[14] B. Gebremichael, F. Vaandrager, and M. Zhang. Analysis of the zeroconf protocol using Uppaal. In *EMSOFT '06*, pages 242–251. ACM Press, 2006.

[15] K. Havelund, A. Skou, K. G. Larsen, and K. Lund. Formal modeling and analysis of an audio/video protocol: an industrial case study using Uppaal. In *IEEE Real-Time Systems Symposium, RTSS '97*, pages 2–13. IEEE Computer Society, 1997.

[16] T. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111(2):193–244, 1994.

[17] D. Kozen. Results on the propositional mu-calculus. *Theoretical Computer Science*, 27:333–354, 1983.

[18] M. Lindahl, P. Pettersson, and W. Yi. Formal design and analysis of a gearbox controller. *Software Tools for Technology Transfer (STTT)*, 3(3):353–368, 2001.

[19] F. Moller and C. Tofts. A temporal calculus of communicating systems. In *Proc. of CONCUR 1990*, pages 401–415. Springer-Verlag New York, Inc., 1990.

[20] J. Sifakis and S. Yovine. Compositional specification of timed systems. In *Proc. of STACS'96*, LNCS 1046, pages 347–359. Springer-Verlag, 1996.

[21] S. Tripakis and S. Yovine. The analysis of timed systems using time-abstracting bisimulations. *Formal Methods in System Design*, 18(1):25–68, 2001.

[22] F. Vaandrager and A. de Groot. Analysis of a biphase mark protocol with Uppaal and PVS. *Formal Aspects of Computing*, 18(4):433–458, 2006.

[23] S. Yovine. Kronos: A verification tool for real-time systems. *International Journal of Software Tools for Technology Transfer*, 1(1-2):123–133, 1997.

A Avoiding eager/lazy synchronization

The following steps transform any TAD network with eager and lazy actions into an equivalent (bisimilar) network where eager and lazy actions do not share labels.[7] Let $u, u' : CA \to CA$ be relabeling functions, s.t. for any $a \in CA$, $u(a)$ and $u'(a)$ are distinct from each other and distinct from any label used in the input TAD network.

1. Any lazy action that has only eager matches is replaced by an eager action with the same source and target locations, label, guard and reset set.

2. All labels of eager half actions are consistently renamed. For any $a \in CA$ that occurs in the label of an eager half action, a is renamed to a fresh label $u(a)$.[8]

3. For any lazy input action t_l labeled with $a?$, s.t. there is an eager output action t_e labeled with $u(a)!$, we add a new eager input action t'_l labeled with $u(a)?$, with the same source and target locations, guard and reset set as t_l.

4. For any lazy output action t_l labeled with $a!$, s.t. there is an eager action t_e labeled with $u(a)?$, we add a new eager output action t'_l labeled with $u'(a)!$, with the same source and target locations, guard and reset set as t_l, and a new eager output action t'_e labeled with $u'(a)?$, with the same source and target locations, guard and reset set as t_e.

Avoiding name clashes. Note that, different labels are needed (u vs u') to avoid unwanted synchronization between added input/output eager actions. The problem of using $u = u'$ is illustrated by fig. 9. The TAD network (on the left) has four components, `Q`, `R`, `S` and `T`. Note that synchronization between `Q` and `R` may occur at any time (`a!` and `a?` are lazy actions in `Q` and `R`), while both `S` and `T` will demand synchronization to occur asap (`a!` and `a?` are eager actions in `S` and `T`). In the TAD network in the middle, we have renamed eager actions `a!` and `a?` to `ua!` and `ua?` (assuming $u(\mathtt{a}) = \mathtt{ua}$). But the behavior of this TAD network differs from that of the original: As soon as either `S1` or `T1` are entered, delays in `R0` and `Q0` will be prevented because synchronization on `ua` is possible and urgent. This has the side effect of enforcing urgent synchronization also between the lazy actions `a!` and `a?`. In contrast (fig. 9, right), a correct translation ensures that different urgent channels are used to represent synchronization between: (a) input lazy or eager actions and their output eager matches ($u(\mathtt{a}) = \mathtt{ua}$), and (b) output lazy actions and their input eager matches ($u'(\mathtt{a}) = \mathtt{ua\_o}$).

A.1 Formal definition

For any set of transitions T in a TAD network, let $T_{\text{la}}, T_{\text{ea}} \subseteq T$ be the sets of all lazy and eager actions in T, resp. Let $|A = \langle A_1, \ldots, A_n \rangle$ be a TAD network, defined as in § 2, where $A_i = (L_i, l_{i,0}, Lab_i, T_i, C_i)$, $i : 1..n$. Let $T = \bigcup_{i=1}^{n} T_i$. Let $LE \subseteq T$ be the set of all lazy half actions without matching lazy actions,

$$LE = \{\, t \in T_{\text{la}} \mid lab(t) \in HA \;\wedge\; \forall\, t' \in T.\ t||t' \Rightarrow t' \in T_{\text{ea}} \,\}$$

We define $|A_1 = \langle A_{1,1}, \ldots, A_{n,1} \rangle$, where $A_{i,1} = (L_i, l_{i,0}, Lab_i, T_{i,1}, C_i)$, $i : 1..n$, and

$$T_{i,1} = (T_{i,1} \setminus LE) \cup \{\, (l, a, g, eager, r, l') \mid (l, a, g, lazy, r, l') \in T_i \cap LE \,\}$$

[7] We do not claim these steps to be optimal.

[8] We say that $a \in CA$ occurs in $act \in Act$ if $act \in \{a, a?, a!\}$.

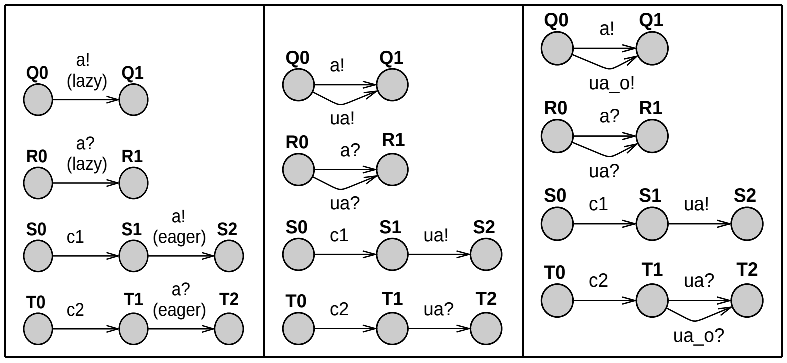


Figure 9: A conflict between multiple lazy actions with eager matches

Let $T^1 = \bigcup_{i=1}^{n} T_{i,1}$. We define $|A_2 = \langle A_{1,2}, \ldots, A_{n,2} \rangle$, where $A_{i,2} = (L_i, l_{i,0}, Lab_i, T_{i,2}, C_i)$, $i : 1..n$, and

$$\begin{aligned} T_{i,2} = \; & (T_{i,1} \setminus \{\, t \in T^1_{\mathrm{ea}} \mid lab(t) \in HA \,\}) \cup \\ & \{\, (l, urg(a), g, eager, r, l') \mid (l, a, g, eager, r, l') \in T_{i,1},\; a \in HA \,\} \end{aligned}$$

where $urg(a!) = u(a)!$ and $urg(a?) = u(a)?$. Let $T^2 = \bigcup_{i=1}^{n} T_{i,2}$. We define $|A_3 = \langle A_{1,3}, \ldots, A_{n,3} \rangle$, where $A_{i,3} = (L_i, l_{i,0}, Lab_i, T_{i,3}, C_i)$, $i : 1..n$, and

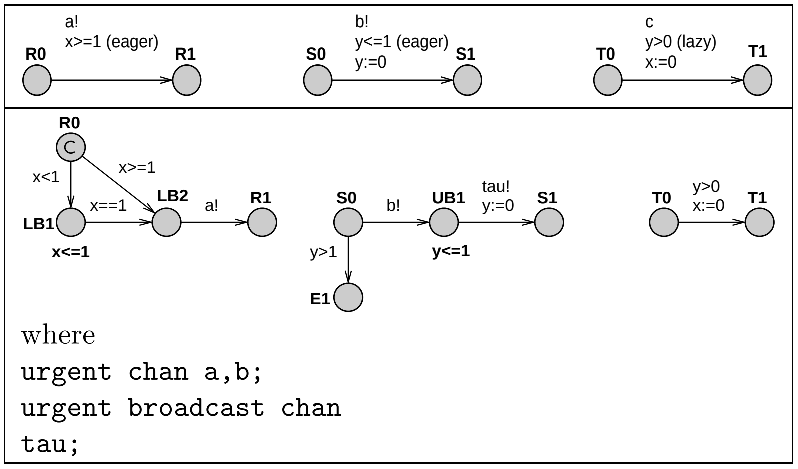
$$\begin{aligned} T_{i,3} = \; & T_{i,2} \cup \\ & \{\, (l, u(a)?, g, eager, r, l') \mid && (l, a?, g, lazy, r, l') \in T_{i,2}, \\ & && \exists\, t \in T^2_{\mathrm{ea}} \setminus T_{i,2}.\; lab(t) = u(a)! \,\} \cup \\ & \{\, (l, u'(a)!, g, eager, r, l') \mid && (l, a!, g, lazy, r, l') \in T_{i,2}, \\ & && \exists\, t \in T^2_{\mathrm{ea}} \setminus T_{i,2}.\; lab(t) = u(a)? \,\} \cup \\ & \{\, (l, u'(a)?, g, eager, r, l') \mid && (l, u(a)?, g, eager, r, l') \in T_{i,2}, \\ & && \exists\, t \in T^2_{\mathrm{la}} \setminus T_{i,2}.\; lab(t) = a! \,\} \end{aligned}$$

Let $normalize(|A) = |A_3$ as defined above.

THEOREM **A.1.** *Let $|A$ be a TAD network, defined as in § 2, and $|A' = normalize(|A)$. Then, the following holds.*

1. *Eager and lazy actions in $|A'$ do not share labels. Formally, if T is the set of transitions in $|A'$, then*
$$\forall\, t \in T_{\mathrm{ea}}, t' \in T_{\mathrm{la}}.\; lab(t) \neq lab(t')$$
2. *$|A$ and $|A'$ are strongly timed-bisimilar*

Proof. We will only offer an informal argument here. Consider the networks produced from $|A$ by each intermediate step of the transformation, $|A_1$, $|A_2$ and $|A_3 = normalize(|A)$, as defined above. It is easy to see that $|A_1$ has the same behavior than $|A$ and all lazy actions in $|A_1$ are either completed actions or match both lazy and eager actions. $|A_2$ preserves completed actions, lazy/lazy and eager/eager synchronization. Finally, $|A_3$ recovers synchronization between lazy input actions and eager output actions, and between lazy output actions and eager input actions. This is achieved by adding eager actions which are equivalent to the corresponding lazy actions. □



where

```
urgent chan a,b;
urgent broadcast chan
tau;
```

Figure 10: The role of concurrent resets (top: TAD network, bottom: TA network)

B Proofs

Here we prove that the translation $\mathcal{T}$, as defined in § 4, generates timed-bisimilar networks with a linear increase in size. Notation has been borrowed from § 2, 3 and 4 (unless stated otherwise).

Restrictions on concurrent resets. Let us recall here the syntactic restrictions on clock resets that were imposed on the input TAD networks § 4.

> If a clock x occurs in a lower bound of an eager action in component P, then no other component Q may reset x. Also, if a clock x is reset by an upper-bounded eager action in component P, then x cannot occur in lower bounds of transitions in any other component Q.

We mentioned that the soundness of the translation may be compromised if such restrictions were not met by the TAD network; Figure 10 illustrates the issues.

One of the problems is related to the use of lb-locations to represent eager actions with lower bounds. Note that, if `x` were concurrently reset by another component while execution is in `LB2`, e.g. by `c` in `T0`, the urgent transition `a!` would be enabled at $v(\texttt{x}) < 1$, violating the lower bound of the original eager action `a!`. Thus, the generated TA network would exhibit a behavior that is not possible in the TAD network.

The other problem is related to the non-atomic representation of eager actions with upper bounds. The internal transition from `T0` to `T1` in the TA network (which represents the lazy action `c` in the TAD network) could be enabled when `UB1` is entered, thus it could be executed immediately after the urgent transition `b!` but before the `tau!` transition (which collectively, but not atomically, represent the eager action `b!` in the TAD network). However, the immediate execution of `c` after `b!` is not possible in the TAD network, because `b!` resets `y` and `c` has a lower bound on `y`. Again, the generated TA network would not be equivalent to the TAD network.

Note that, the restrictions on clock resets guarantee that the above scenarios cannot occur.

Preliminaries. Let $|A$ be the TAD network and $|B = \mathcal{T}(|A)$ the resulting TA network. Let $TTS_A = (S_A, s_{0,A}, Lab_A \cup \mathbb{R}^+, T_{S,A})$ and $TTS_B = (S_B, s_{0,B}, Lab_B \cup \mathbb{R}^+, T_{S,B})$ be the respective timed transition systems. (Here, for the sake of readability, we will use $s \xrightarrow{a} s'$ to denote action transitions, $s \xrightarrow{\delta} s'$ to denote delay transitions). Let T_A, T_B, L_A and L_B be the set of all transitions and locations of $|A$ and

$|B$, resp. Let $L_B^{\text{ub}} \subseteq L_B$ be the set of all ub-locations in $|B$. For any run ρ, $\mathit{last}(\rho)$ is the first state of ρ; $\mathit{last}(\rho)$ is the last state in ρ (if ρ is finite); $\mathit{delay}(\rho) = \Sigma_{\{i\in\mathbb{N}|\gamma_i\in\mathbb{R}^+\}}\gamma_i$ is the sum of all delays in ρ; and $\mathit{Edges}(\rho)$ is the set of all automata transitions that participate in any action transition in ρ.

The equivalence between TAD and TA locations is given by the relation $\equiv_{\text{loc}} \subseteq L_A \times L_B$, s.t. $l \equiv_{\text{loc}} l'$ iff

$$\begin{aligned} l' \in\ & \{\, l \,\} \cup \\ & \{\, l_{\text{lb}}^k \mid k : 1..|\Gamma(l)| \,\} \cup \\ & \{\, l_{\text{esc}}^{k,u} \mid k : 1..|\Gamma(l)|,\ \iota_k \in \Gamma(l),\ u \in \mathit{UB}(l, \iota_k) \,\} \cup \\ & \{\, \mathit{loc}_{\text{ub}}^j \mid (\mathit{loc}_{\text{ub}}^j, \texttt{tau!}, \mathit{true}, r_j, l) \in T_B,\ \mathit{loc} \in L_A,\ j : 1..|T_A| \,\} \end{aligned}$$

The equivalence between TAD and TA states is given by the relation $\equiv_{\text{state}} \subseteq S_A \times S_B$, s.t. $\langle \bar{l}, v\rangle \equiv_{\text{state}} \langle \bar{l}', v'\rangle$ iff

$$v = R(\bar{l}')(v') \ \wedge\ \forall\, i : 1..n.\ \bar{l}[i] \equiv_{\text{loc}} \bar{l}'[i]$$

where $R(\bar{l}')$ is the set of all clocks that are reset in tau!-transitions whose source locations belong to $\bar{l}'$, i.e.,

$$R(\bar{l}') = \bigcup \{\, r(t) \mid t \in T_B,\ \mathit{src}(t) \in L_B^{\text{ub}},\ \exists\, i : 1..n.\ \bar{l}'[i] = \mathit{src}(t) \,\}$$

Note that, two states, $\langle \bar{l}, v\rangle$ in the TAD network and $\langle \bar{l}', v'\rangle$ in the TA network, are considered equivalent, $\langle \bar{l}, v\rangle \equiv_{\text{state}} \langle \bar{l}', v'\rangle$, when their location vectors are equivalent ($\forall\, i : 1..n.\ \bar{l}[i] \equiv_{\text{loc}} \bar{l}'[i]$) and the valuations coincide after executing all tau!-transitions from ub-locations in $\bar{l}'$, if any such transition exists ($v = R(v')$). This takes into account the transient states represented by ub-locations, which split the otherwise atomic execution of those urgent transitions in the TA network that represent eager actions with upper bounds in the TAD network (§ 4.2).

Let $AT_B \subseteq T_B$ be the set of auxiliary transitions in the TA network,

$$\begin{aligned} AT_B =\ & \{\, t_B \in T_B \mid \nexists\, t_A \in T_A,\ l_B \in L_B.\ t_B \in \mathit{upp}(t_A, l_B) \,\} \cup \\ & \{\, t_B \in T_B \mid \exists\, l, l' \in L_B,\ j : 1..|T_A|.\ t_B = (l_{\text{ub}}^j, \texttt{tau!}, \mathit{true},\ r_j, l') \,\} \end{aligned}$$

Let R_B be the set of all runs exhibited by $|B$. Let $\Rightarrow\ \subseteq S_B \times S_B$ be the transition relation that abstracts over auxiliary transitions,

$$\begin{aligned} s \overset{a}{\Longrightarrow} s' \text{ iff } \exists\, s_1, s_2 \in S_B.\quad & s \xrightarrow{\tau}{}^* s_1 \xrightarrow{a} s_2 \xrightarrow{\tau}{}^* s' \ \wedge \\ & (\mathit{Edges}(s \xrightarrow{\tau}{}^* s_1) \cup \mathit{Edges}(s_2 \xrightarrow{\tau}{}^* s')) \subseteq AT_B \\ s \overset{\delta}{\Longrightarrow} s' \text{ iff } \exists\, \rho \in R_B.\quad & \mathit{first}(\rho) = s \ \wedge\ \mathit{last}(\rho) = s' \ \wedge \\ & \mathit{Edges}(\rho) \subseteq AT_B \ \wedge\ \mathit{delay}(\rho) = \delta \end{aligned}$$

The equivalence between TAD and TA transitions is given by the relation $\equiv_{\text{edge}} \subseteq T_A \times T_B$, s.t., $\mathit{act}_A \equiv_{\text{edge}} \mathit{act}_B$ iff

$$\forall\, t_B \in \mathit{Edges}(\mathit{act}_B) \setminus AT_B.\ \exists\, t_A \in \mathit{Edges}(\mathit{act}_A),\ l \in L_B.\ t_B \in \mathit{upp}(t_A, l)$$

Next, we introduce some auxiliary lemmas.

LEMMA **B.1.** *Let $s_A \equiv_{\text{state}} s_B$. Any state s'_B that is immediately reachable from s_B via a sequence of auxiliary transitions is also equivalent to s_A.*

$$\forall\, s'_B \in S_B.\ s_B \xrightarrow{\tau}{}^* s'_B \ \wedge\ \mathit{Edges}(s_B \xrightarrow{\tau}{}^* s'_B) \subseteq AT_B \ \Rightarrow\ s_A \equiv_{\text{state}} s'_B$$

Proof. Let $s_A = \langle \bar{l}_A, v_A \rangle$ and $s_B = \langle \bar{l}_B, v_B \rangle$, $s_A \equiv_{\text{state}} s_B$. By definition of $\equiv_{\text{state}}$, $\forall\, i : 1..n.\ \bar{l}_A[i] \equiv_{\text{loc}} \bar{l}_B[i]$ and $v_A = R(\bar{l}_B)(v_B)$. By def. of $\mathcal{T}$, from s_B, a sequence of auxiliary transitions with zero accumulated delay may only reach location vectors $\bar{l}'_B$ that are equivalent to $\bar{l}_A$, and the clocks that are reset in the sequence are those which are reset by `tau!`-transitions from ub-locations, say $R' \subseteq R(\bar{l}_B)$. Then, any such $\bar{l}'_B$ is reached with a valuation $v'_B = R'(v_B)$. Hence, $\forall\, i : 1..n.\ \bar{l}_A[i] \equiv_{\text{loc}} \bar{l}'_B[i]$ and $v_A = (R(\bar{l}_B) \setminus R')(v'_B) = R(\bar{l}'_B)(v'_B)$. By definition, $s_A \equiv_{\text{state}} s'_B$. □

LEMMA **B.2.** *Let $l_A \in L_A$, $t_A \in T(l_A)$ and v s.t. $v \models g(t_A)$. Let $l_B \in L_B$ s.t. $l_A \equiv_{\text{loc}} l_B$. If v is reachable in l_B then $t_B \in upp(t_A, l_B)$, the TA transition that represents t_A, is either offered in l_B or in some other auxiliary location that is immediately reachable from l_B by a sequence of auxiliary transitions.*

Proof. By definition of $upp(t_A, l_B)$, $g(t_B) = g(t_A)$ or $g(t_B) = true$. By definition of $\mathcal{T}$, if $g(t_B) = true$ but $g(t_B) \neq g(t_A)$, then t_A must be an eager action, t_B must be the urgent transition representing t_A, and $I(tgt(t_B))$ correctly enforces the upper bound in t_A (if any). Hence, a proof by contradiction must necessarily assume that $l_B \neq src(t_B)$ and that $src(t_B)$ cannot be reached immediately from l_B by a sequence of auxiliary transitions.

Then, by definition of $\mathcal{T}$, l_B must be an auxiliary location which may only be entered with valuations v' s.t. $v'(x_{l_A}) > c$ (for some $c \in \mathbb{N}, c > 0$), but where $v(x_{l_A}) < c_1$ was reached in l_B, after l_B was entered and while execution remained in l_B. This would imply that x_{l_A} was reset by some other component B_j, $j \neq i$ while execution remained in l_B. Note that, also by definition of $\mathcal{T}$, l_B implies the existence of an eager action in l_A with a lower bound on x_{l_A}. Then, there must exist a transition in A_j that resets x_{l_A}, where x_{l_A} occurs in the lower bound of an eager action in A_i. This violates a syntactic requirement on the TAD network (contradiction). □

LEMMA **B.3.** *Let $s_A = \langle \bar{l}_A, v_A \rangle \in S_A$ and $s_B = \langle \bar{l}_B, v_B \rangle \in S_B$ s.t. $s_A \equiv_{\text{state}} s_B$. For any TAD transition t_A that is offered in s_A, the TA transition t_B that represents t_A is offered in some $s'_B = \langle \bar{l}'_B, v'_B \rangle$ that is immediately reachable from s_B via a sequence of auxiliary transitions.*

Conversely, for any TA transition t_B that is offered in s_B, s.t. t_B represents a given TAD transition t_A, t_A is offered in s_A.

Proof. ($\Rightarrow$) Let $t_A \in T(l_A)$, where $\bar{l}_A[i] = l_A$ for some $i : 1..n$ and $v_A \models g(t_A)$. By definition of $\mathcal{T}$, TA transitions that represent TAD transitions are not offered from ub-locations. Then, w.l.o.g., consider the state $s'_B = \langle \bar{l}'_B, v'_B \rangle$ that is reached from s_B by executing (in any order) all the `tau!`-transitions in ub-locations in s_B. By definition of $\equiv_{\text{state}}$, $\bar{l}'_B$ does not contain ub-locations and $v'_B = v_A$. Hence, Lemma B.2 guarantees that a transition t_B, which represents t_A, will be offered at some state s''_B which is immediately reachable from s_B by a sequence of auxiliary transitions.

($\Leftarrow$) Let $t_B \in T(l_B)$, where $\bar{l}_B[i] = l_B$ for some $i : 1..n$, and $v_B \models g(t_B)$ and $r(t_B)(v_B) \models I(tgt(t_B))$, s.t. there exists some TAD transition t_A that is represented by t_B. Thus, $t_B \in upp(t_A, l_B)$.

By contradiction, assume that either $src(t_A) \notin \bar{l}_A$ or $v_A \not\models g(t_A)$ (i.e., t_A is not offered in s_A). By definition of $\equiv_{\text{state}}$, $\equiv_{\text{loc}}$ and $upp(t_A, l_B)$, and because $src(t_B) \in \bar{l}_B$, $src(t_A) \in \bar{l}_A$. Then, it must be the case that $v_A \not\models g(t_A)$.

By definition of $\equiv_{\text{state}}$, and because $s_B \equiv_{\text{state}} s_A$, the only difference between v_B and v_A is in the value of clocks that are reset by `tau!`-transitions from ub-locations in $\bar{l}_B$. That is, $v_A = R(\bar{l}_B)(v_B)$. We show that a contradiction arises from our assumption that $v_B \models t_B$ and $r(t_B)(v_B) \models I(tgt(t_B))$ but $v_A \not\models g(t_A)$.

There are two cases to consider, depending on the form of $g(t_B)$. By definition of $upp(t_A, l_B)$, either $g(t_B) = g(t_A)$ or $g(t_B) = true$.

Case 1. Assume that $g(t_B) = g(t_A)$, $v_B \models t_B$ and $r(t_B)(v_B) \models I(tgt(t_B))$ but $v_A \not\models g(t_A)$. Then $g(t_A)$

must contain a lower bound n on some clock $x \in R(\bar{l}_B)$ s.t. $v_B(x) \geq n$ and x is reset by a `tau!`-transition from an ub-location in some other component B_j, $j \neq i$. By definition of $\mathcal{T}$, there must be some eager action with upper bound in $A_j, j \neq i$ that resets x, where x occurs in the lower bound of a transition in A_i (t_A). This violates a syntactic requirement on the TAD network (contradiction).

Case 2. Now, assume that $g(t_B) = \mathit{true}$, $g(t_B) \neq g(t_A)$, $v_B \models t_B$ and $r(t_B)(v_B) \models I(\mathit{tgt}(t_B))$ but $v_A \not\models g(t_A)$. Note that, t_B must be an urgent transition representing an eager action t_A. By definition of $\mathcal{T}$, the upper bound in t_A (if any) is represented by the invariant in the ub-location $\mathit{tgt}(t_B)$, in which case $r(t_B) = \emptyset$ ($r(t_A)$ is copied to the `tau!`-transition from $\mathit{tgt}(t_B)$). Hence, $r(t_B)(v_B) \models I(\mathit{tgt}(t_B))$ implies $v_B \models I(\mathit{tgt}(t_B))$, which implies $v_B \models x \leq n$ for an upper bound $x \leq n$ in $g(t_A)$ (or $v_B \models x < n$ for an upper bound $x < n$ in $g(t_A)$). In other words, v_B does not invalidate the upper bound in $g(t_A)$. In turn, because v_A may differ from v_B only in that $v_B(x) > v_A(x) = 0$ (for some clock $x \in R(\bar{l}_B)$), v_A does not invalidate the upper bound in $g(t_A)$. Next, we prove that v_A does not invalidate the lower bound in $g(t_A)$, either.

Assume that t_A has a lower bound. By definition of $\mathcal{T}$, the source location of t_B must have been entered (in the current run) in some $s^e_B = \langle \bar{l}^e_B, v^e_B \rangle$ where $v^e_B \models x \geq n$ for a lower bound $x \geq n$ in $g(t_A)$ (or $v^e_B \models x > n$ for a lower bound $x > n$ in $g(t_A)$). Now, if s^e_B is reached while traversing the sequence of auxiliary transitions from s_B to s'_B, then $v_A = R(\bar{l}^e_B)(v^e_B)$ and there must be some eager action with upper bound in $A_j, j \neq i$ that resets x, where x occurs in the lower bound of a transition in A_i (t_A). This violates a syntactic requirement on the TAD network (contradiction). On the other hand, s^e_B might have been reached before s_B and x was reset by some transition in $B_j, j \neq i$, which was executed between s^e_B and s_B. This would imply the existence of transition in A_j that resets x, where x occurs in the lower bound of an eager action in A_i (t_A). Again, this violates a syntactic requirement on the TAD network (contradiction).

□

LEMMA **B.4.** *From equivalent states, $|A$ and $|B$ offer equivalent actions (abstracting over auxiliary transitions in $|B$). Formally, $s_A \equiv_{\text{state}} s_B$ implies,*

1. $\forall\, a \in \mathit{Lab}_A,\ s'_A \in S_A.\ \mathit{act}_A = s_A \xrightarrow{a} s'_A \ \Rightarrow$
   $\exists\, b \in \mathit{Lab}_B,\ s'_B \in S_B.\ \mathit{act}_B = s_B \overset{b}{\Longrightarrow} s'_B \ \wedge\ s'_A \equiv_{\text{state}} s'_B \ \wedge\ \mathit{act}_A \equiv_{\text{edge}} \mathit{act}_B$
2. $\forall\, b \in \mathit{Lab}_B,\ s'_B \in S_B.\ \mathit{act}_B = s_B \overset{b}{\Longrightarrow} s'_B \ \Rightarrow$
   $\exists\, a \in \mathit{Lab}_A,\ s'_A \in S_A.\ \mathit{act}_A = s_A \xrightarrow{a} s'_A \ \wedge\ s'_A \equiv_{\text{state}} s'_B \ \wedge\ \mathit{act}_A \equiv_{\text{edge}} \mathit{act}_B$

Proof. Follows from Lemma B.3 (equivalent states offer equivalent transitions) and definition of $\equiv_{\text{state}}$ and $\equiv_{\text{edge}}$ (the execution of equivalent transitions, from equivalent states, yields equivalent next states).
□

LEMMA **B.5.** *Let $s_A \equiv_{\text{state}} s_B$. If $s_A \xrightarrow{\delta} s'_A \in T_{S,A}$ and $s_B \overset{\delta}{\Longrightarrow} s'_B \in T_{S,B}$ for some $\delta \in \mathbb{R}$, then $s'_A \equiv_{\text{state}} s'_B$.*

Proof. Let $s_A = \langle \bar{l}_A, v_A \rangle$ and $s_B = \langle \bar{l}_B, v_B \rangle$. By definition of $\equiv_{\text{state}}$, $v_A = v_B$ and $\bar{l}_A[i] \equiv_{\text{loc}} \bar{l}_B[i]$ for any $i : 1..n$. By definition of $\equiv_{\text{loc}}$, for any $i : 1..n$ and $\bar{l}_A[i] = l$, $l_B[i]$ may be a location of the form l, l^k_{lb}, $l^{k,u}_{\text{esc}}$ or $\mathit{loc}^j_{\text{ub}}$ (for some $k : 1..|\Gamma(l)|$, $\iota_k \in \Gamma(l)$, $u \in \mathit{UB}(l, \iota_k)$, $\mathit{loc} \in L_A$, $j : 1..|T_A|$ and $(\mathit{loc}^j_{\text{ub}}, \texttt{tau!}, \mathit{true}, r_j, l) \in T_B$).

By hypothesis, for all $i : 1..n$, there is a run $\bar{l}_B[i]$ that only involves auxiliary transitions and has a cumulative duration of δ time units. By construction of the TAD network, such a run may only lead to some location l' of the form l, $l^{k'}_{\text{lb}}$ or $l^{k',u'}_{\text{esc}}$ (for some $k' : 1..|\Gamma(l)|$, $k' \geq k$ and $u' \in \mathit{UB}(l, \iota_{k'})$).

Now, we know that $s'_A = \langle \bar{l}_A, v_A + \delta \rangle$ and $s'_B = \langle \bar{l}'_B, v_A + \delta \rangle$, where $\bar{l}_A[i] = l$ implies that $\bar{l}'_B[i] = l'$ as explained above $(i : 1..n)$. Therefore, by definition of $\equiv_{\text{state}}$, $s_A \equiv_{\text{state}} s'_B$. □

LEMMA **B.6.** *From equivalent states, $|A$ and $|B$ may pass the same amount of time (abstracting over auxiliary transitions in $|B$). Formally, $s_A \equiv_{\text{state}} s_B$ implies,*

1. $\forall\, \delta \in \mathbb{R},\, s'_A \in S_A.\; s_A \xrightarrow{\delta} s'_A \;\Rightarrow\; \exists\, s'_B.\; s_B \overset{\delta}{\Longrightarrow} s'_B \;\wedge\; s'_A \equiv_{\text{state}} s'_B \;\wedge$
2. $\forall\, \delta \in \mathbb{R},\, s'_B \in S_B.\; s_B \overset{\delta}{\Longrightarrow} s'_B \;\Rightarrow\; \exists\, s'_A \in S_A.\; s_A \xrightarrow{\delta} s'_A \;\wedge\; s'_A \equiv_{\text{state}} s'_B$

Proof. Here we will prove statement 1; the proof of statement 2 can be constructed in a similar way. Assume that δ time units can pass from s_A and lead to s'_A. We will prove, by contradiction, that δ time units can also pass from s_B and lead to s'_B s.t. $s'_A \equiv_{\text{state}} s'_B$, possibly along a number of auxiliary locations.

Let $s_A = \langle \bar{l}_A, v\rangle \;\equiv_{\text{state}}\; s_B = \langle \bar{l}_B, v\rangle$ be the equivalent states and $s_A \xrightarrow{\delta} (s_A + \delta) \in T_{S,A}$, $\delta \in \mathbb{R}$ be a delay transition in the TAD network. Then,

(†) No eager action may participate of any transition $(s_A + \delta') \xrightarrow{a} s'_A \in T_{S,A}$, $\delta' \in \mathbb{R}$, $\delta' < \delta$.

By contradiction, assume that there is no equivalent delay transition in the TA network, i.e.

(‡) $s_B \overset{\delta}{\Longrightarrow} s'_B \notin T_{S,B}$

Then, either (a) $s_B \overset{\delta}{\Longrightarrow} s''_B \in T_{S,B}$ for some $s''_B \in S_B$ s.t. $(s_A + \delta, s''_B) \notin \equiv_{\text{state}}$, or (b) there are $s''_B \in S_B$ and $\delta' < \delta$ s.t. $s_B \overset{\delta'}{\Longrightarrow} s''_B \in T_{S,B}$ and either (b.1) there is some urgent transition t that may be performed in s''_B or (b.2) some invariant's upper bound is satisfied in s''_B (or a committed location is in s''_B) but no auxiliary transition is enabled in s''_B. By Lemma B.5, the case (a) is not possible. By Lemmas B.5 and B.4, the case (b.1) implies the existence of some $s''_A \in S_A$, $s''_A = s_A + \delta'$ s.t. $s_B \overset{\delta'}{\Longrightarrow} s''_A \in T_{S,A}$ and the eager action represented by t may be performed in s''_A. This contradicts (†). Finally, by construction of the TA network, case (b.2) is not possible either. Note that, we reach a contradiction by assuming (‡); hence, the lemma holds. $\square$

Theorem 4.1 (Bisimulation) Let $|A$ be a TAD network and $|B = \mathcal{T}(|A)$ the resulting TA network. $|A$ and $|B$ are strongly timed-bisimilar [10].

Proof. The relation $\equiv_{\text{state}}$ is a strong timed-bisimulation between $|A$ and $|B$ that abstracts over auxiliary transitions. Formally, $s_A \equiv_{\text{state}} s_B$ implies

1. $\forall\, a \in Lab_A,\, s'_A \in S_A.\; act_A = s_A \xrightarrow{a} s'_A \;\Rightarrow$
   $\exists\, b \in Lab_B,\, s'_B \in S_B.\; act_B = s_B \overset{b}{\Longrightarrow} s'_B \;\wedge\; s'_A \equiv_{\text{state}} s'_B \;\wedge\; act_A \equiv_{\text{edge}} act_B$
2. $\forall\, b \in Lab_B,\, s'_B \in S_B.\; act_B = s_B \overset{b}{\Longrightarrow} s'_B \;\Rightarrow$
   $\exists\, a \in Lab_A,\, s'_A \in S_A.\; act_A = s_A \xrightarrow{a} s'_A \;\wedge\; s'_A \equiv_{\text{state}} s'_B \;\wedge\; act_A \equiv_{\text{edge}} act_B$
3. $\forall\, \delta \in \mathbb{R},\, s'_A \in S_A.\; s_A \xrightarrow{\delta} s'_A \;\Rightarrow\; \exists\, s'_B.\; s_B \overset{\delta}{\Longrightarrow} s'_B \;\wedge\; s'_A \equiv_{\text{state}} s'_B$
4. $\forall\, \delta \in \mathbb{R},\, s'_B \in S_B.\; s_B \overset{\delta}{\Longrightarrow} s'_B \;\Rightarrow\; \exists\, s'_A \in S_A.\; s_A \xrightarrow{\delta} s'_A \;\wedge\; s'_A \equiv_{\text{state}} s'_B$

This follows from Lemmas B.4 and B.6. $\square$

Theorem 4.2 (Linear complexity) Let $|A$ be a TAD network and $|B = \mathcal{T}(|A)$ the resulting TA network. The size of $|B$ is proportional to the size of $|A$.

Proof. Let $|A$ by represented (for the sake of a worst-case analysis) by a TAD network that satisfies the following conditions:

- All transitions in $|A$ are eager half actions.
- All transitions in $|A$ in the same location have different lower bounds. Then, for every location l with m outgoing transitions there will be $m+1$ lb-locations $|B$: l_{lb}^k, $k : 1..(m+1)$. These locations will be reachable from l by $m+1$ auxiliary transitions.
- All transitions in $|A$ in the same location have different and strict upper bounds. Then, for every transition $t_j = (l, act_j, g_j, d_j, r_j, l')$, with upper bound $(<, ub_j)$, $|B$ will contain an ub-location l_{ub}^j (with $I(l_{\mathrm{ub}}^j) < ub_j$) and a transition $(l_{\mathrm{ub}}^j, \mathtt{tau!}, \mathit{true}, r_j, l')$.
- Upper bounds of transitions in $|A$ in the same location are greater than or equal to the upper bound of the transition with the greatest lower bound. Then, in $|B$, (a) the urgent transitions (as generated by $\mathit{upp}()$) which are offered at l_{lb}^k, $k : 2..m$, are also offered at any l_{lb}^q, $q : 3..m+1$, (b) for every transition t_j with upper bound $(<, ub_j)$ there is an e-location $l_{\mathrm{esc}}^{m+1,(<,ub_j)}$, which offers transitions equivalent to t'_j, for all $ub'_j > ub_j$, and which connects to the next e-location $l_{\mathrm{esc}}^{m+1,(<,ub'_j)}$ via $(l_{\mathrm{esc}}^{m+1,(<,ub_j)}, \tau, x_l \geq ub_j, \emptyset, l_{\mathrm{esc}}^{m+1,(<,ub'_j)})$ (with $(<, ub'_j) = \mathit{next}((<, ub_j), \mathit{UB}(l, \iota_{m+1})))$.

Let the size of $|A$ be given by (c_l, c_t), where c_l is the number of locations and c_t is the number of transitions. Let *out* be the maximum number of outgoing transitions in any location in $|A$. The number of locations in $|B$ is bounded from above by

$$\underbrace{c_l}_{1} + \underbrace{(c_t + c_l)}_{2} + \underbrace{c_t}_{3} + \underbrace{c_t}_{4} = 2c_l + 3c_t$$

where the terms correspond to, respectively, (1) all committed (source) locations, (2) all lb-locations l_{lb}^k, (3) all ub-locations l_{ub}^j and (4) all e-locations $l_{\mathrm{esc}}^{m+1,(<,ub_j)}$. The number of transitions in $|B$ is bounded from above by

$$\underbrace{(c_t + c_l)}_{1} + \underbrace{c_t}_{2} + \underbrace{c_t}_{3} + \underbrace{out.c_t}_{4} + \underbrace{out.c_t}_{5} + \underbrace{c_t}_{6} = c_l + (2out + 4)c_t$$

where the terms correspond to, respectively, (1) all auxiliary transitions from l to l_{lb}^k, (2) all auxiliary transitions from l_{lb}^k to l_{lb}^{k+1}, (3) all auxiliary `tau!`-transitions from l_{ub}^j to the target locations, (4) an upper bound on all urgent transitions representing the TAD transitions, copied from every l_{lb}^k, (5) an upper bound on all urgent transitions representing the TAD transitions, copied from every e-location and (6) all auxiliary transitions from one e-location to the next. $\square$