FI MU

Faculty of Informatics
Masaryk University Brno

A Framework for Relating Timed Transition Systems and Preserving TCTL Model Checking

by

Lasse Jacobsen
Morten Jacobsen
Mikael H. Møller
Jiří Srba

FI MU Report Series FIMU-RS-2010-9

Publications in the FI MU Report Series are in general accessible via WWW:

`http://www.fi.muni.cz/reports/`

Further information can be obtained by contacting:

Faculty of Informatics
Masaryk University
Botanická 68a
602 00 Brno
Czech Republic

A Framework for Relating Timed Transition Systems and Preserving TCTL Model Checking

Lasse Jacobsen
Aalborg University
Selma Lagerlöfs Vej 300
9220 Aalborg East
ljacob09@student.aau.dk

Morten Jacobsen
Aalborg University
Selma Lagerlöfs Vej 300
9220 Aalborg East
mjacob09@student.aau.dk

Mikael H. Møller
Aalborg University
Selma Lagerlöfs Vej 300
9220 Aalborg East
mikael@i-dyllen.dk

Jiří Srba*
Aalborg University
Selma Lagerlöfs Vej 300
9220 Aalborg East
srba@cs.aau.dk

August 23, 2010

Abstract

Many formal translations between time dependent models have been proposed over the years. While some of them produce timed bisimilar models, others preserve only reachability or (weak) trace equivalence. We suggest a general framework for arguing when a translation preserves Timed Computation Tree Logic (TCTL) or its safety fragment. The framework works at the level of timed transition systems, making it independent of the modeling formalisms and applicable to many of the translations published in the literature. Finally, we present a novel translation from extended Timed-Arc Petri Nets to Networks of Timed Automata and using the framework argue that it preserves the full TCTL. The translation has been implemented in the verification tool TAPAAL.

*Partially supported by Institute for Theoretical Computer Science (ITI), project No. 1M0545.

1 Introduction

Time dependent formal models like Timed Automata (TA) [1], Time Petri Nets (TPN) [14] and Timed-Arc Petri Nets (TAPN) [6] have received significant attention in the theory of embedded systems. While originally developed by different communities of researchers, there has recently been devoted considerable effort towards establishing formal relationships among the different models. To this end, several translations have been developed (see e.g. [5, 6, 8, 9, 10, 11, 13, 16] or [15, 18] for a more complete overview) and some of them have been implemented in verification tools like Romeo [12], TAPAAL [9] or the TIOA Toolkit [2].

Many of these translations utilize similar tricks that allow for the simulation of one system by another. Typically, a single step in one formalism is simulated by a sequence of steps in the other. We identify a general class of translations that preserve Timed Computation Tree Logic (TCTL) (see e.g. [15]), a logic suitable for practical specification of many useful temporal properties. Our main goal is to provide a framework directly applicable to e.g. tool developers. The theory was motivated by the translations presented in [9] and [10]. Unlike much work on TCTL where only infinite alternating runs are considered [15] or the details are simply not discussed [7, 10], we consider also finite maximal runs that appear in the presence of stuck computations or time invariants (strict or nonstrict) and treat the semantics in its full generality as used in some state-of-the-art verification tools like UPPAAL [3]. This is particularly important for liveness properties. While some translations in the literature preserve some variant of timed bisimilarity [8, 10, 11, 13], other translations preserve only reachability or trace equivalence [4, 9]. Our framework allows us to argue that several such translations preserve the full TCTL or at least its safety fragment. In this report we focus only on the interleaving semantics.

To illustrate the applicability of the framework, we propose a novel, full TCTL-preserving translation from extended timed-arc Petri nets to UPPAAL networks of timed automata. Earlier translations either caused exponential blow-up in the size [8, 16, 17], preserved only safety properties [9], or where not suitable for implementation in tools due to an inefficient use of clocks and communication primitives [17]. The translation from TAPN to UPPAAL timed automata presented in this report is the first to run in polynomial time while preserving the full TCTL. We implemented the translation in the tool TAPAAL [9] and the initial experiments confirm its efficiency also in practice.

2 Preliminaries

We let $\mathbb{N}$, $\mathbb{N}_0$, $\mathbb{R}$ and $\mathbb{R}_{\geq 0}$ denote the sets of natural numbers, non-negative integers, real numbers and non-negative real numbers, respectively. A *timed transition system* (TTS) is a quadruple $T = (S, \longrightarrow, \mathcal{AP}, \mu)$ where S is a set of states (or processes), $\longrightarrow \subseteq S \times S \cup S \times \mathbb{R}_{\geq 0} \times S$ is a transition relation, $\mathcal{AP}$ is a set of atomic propositions, and $\mu : S \longrightarrow 2^{\mathcal{AP}}$ is a function assigning sets of true atomic propositions to states.

We write $s \longrightarrow s'$ whenever $(s, s') \in \longrightarrow$ and call them *discrete transitions*, and $s \xrightarrow{d} s'$ whenever $(s, d, s') \in \longrightarrow$ and call them *delay transitions*. We require that the TTS we consider satisfy the following standard axioms for delay transitions (see e.g. [5]). For all $d, d' \in \mathbb{R}_{\geq 0}$ and $s, s', s'' \in S$:

1. **Time Additivity:** if $s \xrightarrow{d} s'$ and $s' \xrightarrow{d'} s''$ then $s \xrightarrow{d+d'} s''$,
2. **Time Continuity:** if $s \xrightarrow{d+d'} s''$ then $s \xrightarrow{d} s' \xrightarrow{d'} s''$ for some s',
3. **Zero delay:** $s \xrightarrow{0} s$ for each state s, and
4. **Time Determinism:** if $s \xrightarrow{d} s'$ and $s \xrightarrow{d} s''$ then $s' = s''$.

By $s[d]$ we denote the state s' (if it exists) such that $s \xrightarrow{d} s'$ (time determinism ensures the uniqueness of $s[d]$). We write $s \longrightarrow$ if $s \longrightarrow s'$ for some $s' \in S$ and $s \not\longrightarrow$ otherwise. Similarly for $\xrightarrow{d}$. A *run* $\rho = s_0 \xrightarrow{d_0} s_0[d_0] \longrightarrow s_1 \xrightarrow{d_1} s_1[d_1] \longrightarrow s_2 \xrightarrow{d_2} \ldots$ is a (finite or infinite) alternating sequence of time delays and discrete actions.

The set of time intervals $\mathcal{I}$ is defined by the abstract syntax

$$I ::= [a, a] \mid [a, b] \mid [a, b) \mid (a, b] \mid (a, b) \mid [a, \infty) \mid (a, \infty)$$

where $a \in \mathbb{N}_0, b \in \mathbb{N}$ and $a < b$.

We shall now introduce the syntax and semantics of Timed Computation Tree Logic (TCTL). The presentation is inspired by [15]. Let $\mathcal{AP}$ be a set of *atomic propositions*. The set of TCTL formulae $\Phi(\mathcal{AP})$ over $\mathcal{AP}$ is given by

$$\varphi ::= \wp \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid E(\varphi_1\, U_I\, \varphi_2) \mid A(\varphi_1\, U_I\, \varphi_2) \mid E(\varphi_1\, R_I\, \varphi_2) \mid A(\varphi_1\, R_I\, \varphi_2)$$

where $\wp \in \mathcal{AP}$ and $I \in \mathcal{I}$. Formulae without any occurrence of the operators $A(\varphi_1\, U_I\, \varphi_2)$ and $E(\varphi_1\, R_I\, \varphi_2)$ form the *safety fragment* of TCTL.

The intuition of the until and release TCTL operators (formalized later on) is as follows: $E(\varphi_1\, U_I\, \varphi_2)$ is true if there exists a maximal run such that φ_2 eventually holds

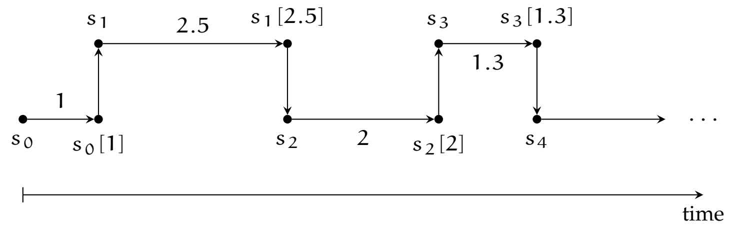

Figure 1: An illustration of the concrete run $\rho = s_0 \xrightarrow{1} s_0[1] \longrightarrow s_1 \xrightarrow{2.5} s_1[2.5] \longrightarrow s_2 \xrightarrow{2} s_2[2] \longrightarrow s_3 \xrightarrow{1.3} s_3[1.3] \longrightarrow s_4 \ldots$.

within the interval I, and until it does, φ_1 continuously holds; $E(\varphi_1 \, R_I \, \varphi_2)$ is true if there exists a maximal run such that either φ_2 always holds within the interval I or φ_1 occurred previously. As we aim to apply our framework to concrete case studies with possible tool support, we need to handle maximal runs in their full generality. Hence we have to consider all possibilities in which a run can be "stuck". In this case, we annotate the last transition of such a run with one of the three special ending symbols (denoted δ in the definition below).

A *maximal run* ρ is either

(i) an infinite alternating sequence of the form $\rho = s_0 \xrightarrow{d_0} s_0[d_0] \longrightarrow s_1 \xrightarrow{d_1} s_1[d_1] \longrightarrow s_2 \xrightarrow{d_2} s_2[d_2] \longrightarrow \ldots$, or

(ii) a finite alternating sequence of the form $\rho = s_0 \xrightarrow{d_0} s_0[d_0] \longrightarrow s_1 \xrightarrow{d_1} s_1[d_1] \longrightarrow \ldots \longrightarrow s_n \xrightarrow{\delta}$ where $\delta \in \{\infty, d_n^{\leq}, d_n^{<}\}$ for some $d_n \in \mathbb{R}_{\geq 0}$ s.t.

   – if $\delta = \infty$ then $s_n \xrightarrow{d} s_n[d]$ for all $d \in \mathbb{R}_{\geq 0}$,

   – if $\delta = d_n^{\leq}$ then $s_n \overset{d}{\not\longrightarrow}$ for all $d > d_n$ and $s_n \xrightarrow{d_n} s_n[d_n]$ s.t. $s_n[d_n] \not\longrightarrow$, and

   – if $\delta = d_n^{<}$ then $s_n \overset{d}{\not\longrightarrow}$ for all $d \geq d_n$, and there exists d_s, $0 \leq d_s < d_n$, such that for all d, $d_s \leq d < d_n$, we have $s_n \xrightarrow{d} s_n[d]$ and $s_n[d] \not\longrightarrow$.

By *MaxRuns*(T, s) we denote the set of maximal runs in a TTS T starting at s.

Intuitively, the three conditions in case (ii) describe all possible ways in which a finite run can terminate. First, a run can end in a state where time diverges. The other two cases define a run which ends in a state from which no discrete transition is allowed after some time delay, but time cannot diverge either (typically caused by the presence of invariants in the model). These cases differ in whether the bound on the maximal time delay can be reached or not.

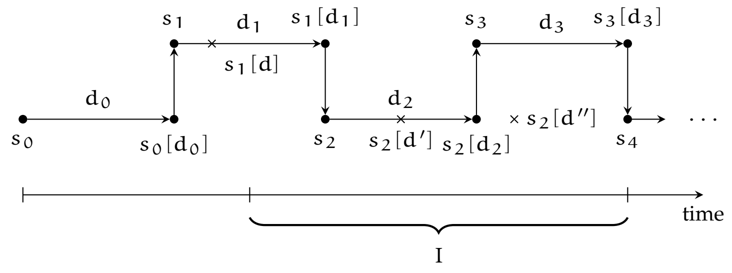

Figure 2: An illustration of a run $\rho = s_0 \xrightarrow{d_0} s_0[d_0] \longrightarrow s_1 \xrightarrow{d_1} s_1[d_1] \longrightarrow s_2 \xrightarrow{d_2} s_2[d_2] \longrightarrow \dots$.

Figure 1 illustrates part of the maximal run $\rho = s_0 \xrightarrow{1} s_0[1] \longrightarrow s_1 \xrightarrow{2.5} s_1[2.5] \longrightarrow s_2 \xrightarrow{2} s_2[2] \longrightarrow s_3 \xrightarrow{1.3} s_3[1.3] \longrightarrow s_4 \longrightarrow \dots$. Note that actions take zero time units and that, although not shown in this example, time delays can be zero so it is possible to do multiple actions in succession without any time progression in between. Further, there is no special meaning as to whether the arrow for an action goes up or down, this is simply to keep the figure small.

Let us now introduce some notation for a given maximal run $\rho = s_0 \xrightarrow{d_0} s_0[d_0] \longrightarrow s_1 \xrightarrow{d_1} s_1[d_1] \longrightarrow s_2 \xrightarrow{d_2} \dots$. First, $r(i,d)$ denotes the total time elapsed from the beginning of the run up to some delay $d \in \mathbb{R}_{\geq 0}$ after the i'th discrete transition. Formally, $r(i,d) = \left(\sum_{j=0}^{i-1} d_j\right) + d$. Second, we define a predicate $\mathit{valid}_\rho : \mathbb{N} \times \mathbb{R}_{\geq 0} \times \mathcal{I} \to \{\mathit{true}, \mathit{false}\}$ such that $\mathit{valid}_\rho(i,d,I)$ checks whether the total time for reaching the state $s_i[d]$ in ρ belongs to the time interval *I*, formally

$$\mathit{valid}_\rho(i,d,I) = \begin{cases} d \leq d_i \wedge r(i,d) \in I & \text{if } d_i \in \mathbb{R}_{\geq 0} \\ r(i,d) \in I & \text{if } d_i = \infty \\ d \leq d_n \wedge r(i,d) \in I & \text{if } d_i = d_n^{\leq} \\ d < d_n \wedge r(i,d) \in I & \text{if } d_i = d_n^{<} . \end{cases}$$

Figure 2 illustrates a run $\rho = s_0 \xrightarrow{d_0} s_0[d_0] \longrightarrow s_1 \xrightarrow{d_1} s_1[d_1] \longrightarrow s_2 \xrightarrow{d_2} s_2[d_2] \longrightarrow \dots$ and three points (marked with $\times$). Note that although time delays appear identical in the figure they can be of different length (even zero). Let us now give some example of the application of the $\mathit{valid}_\rho(i,d,I)$ function. We see that $\mathit{valid}_\rho(1,d,I)$ is false because $s_1[d]$ lies outside the interval I. Similarly, $\mathit{valid}_\rho(2,d'',I)$ is false because $s_2[d'']$ is not a part of the run (since $d'' > d_2$). Finally, $\mathit{valid}_\rho(2,d',I)$ is true because $s_2[d']$ is a part of the run and within I.

Next, we define a function $\mathit{history}_\rho : \mathbb{N} \times \mathbb{R}_{\geq 0} \to 2^{\mathbb{N}\times\mathbb{R}_{\geq 0}}$ s.t. $\mathit{history}_\rho(i, d)$ returns the set of pairs (j, d') that constitute all states $s_j[d']$ in ρ preceding $s_i[d]$, formally $\mathit{history}_\rho(i, d) = \{(j, d') \mid 0 \leq j < i \wedge 0 \leq d' \leq d_j\} \cup \{(i, d') \mid 0 \leq d' < d\}$.

Now we can define the satisfaction relation $s \models \varphi$ for a state $s \in S$ in a TTS $T = (S, \longrightarrow, \mathcal{AP}, \mu)$ and a TCTL formula φ.

$$
\begin{array}{ll}
s \models \wp & \text{iff } \wp \in \mu(s) \\
s \models \neg\varphi & \text{iff } s \not\models \varphi \\
s \models \varphi_1 \wedge \varphi_2 & \text{iff } s \models \varphi_1 \text{ and } s \models \varphi_2 \\
s \models E(\varphi_1\, U_I\, \varphi_2) & \text{iff } \exists \rho \in \mathit{MaxRuns}(T, s)\,. \\
& \quad \exists i \geq 0\,.\, \exists d \in \mathbb{R}_{\geq 0}\,.\, [\mathit{valid}_\rho(i, d, I) \wedge s_i[d] \models \varphi_2 \wedge \\
& \quad \forall (j, d') \in \mathit{history}_\rho(i, d)\,.\, s_j[d'] \models \varphi_1\Big] \\
s \models E(\varphi_1\, R_I\, \varphi_2) & \text{iff } \exists \rho \in \mathit{MaxRuns}(T, s)\,. \\
& \quad \forall i \geq 0\,.\, \forall d \in \mathbb{R}_{\geq 0}\,.\, \mathit{valid}_\rho(i, d, I) \Rightarrow \\
& \quad \Big[s_i[d] \models \varphi_2 \vee \exists (j, d') \in \mathit{history}_\rho(i, d)\,.\, s_j[d'] \models \varphi_1\Big]
\end{array}
$$

The operators $A(\varphi_1\, U_I\, \varphi_2)$ and $A(\varphi_1\, R_I\, \varphi_2)$ are defined analogously by replacing the quantification $\exists \rho \in \mathit{MaxRuns}(T, s)$ with $\forall \rho \in \mathit{MaxRuns}(T, s)$.

Figure 3 illustrates the satisfaction of an until formula and Figure 4 illustrates the satisfaction of a release formula. The x-axis indicates elapsed time (which is why discrete actions are drawn as vertical lines: they take zero time units). The figure illustrates where the two subformulae φ_1 and φ_2 must hold.

In particular, notice that there are four possible ways for a release formula to be satisfied. First, φ_1 may have occurred in the past (outside the interval), which releases φ_2, effectively ensuring that φ_2 need not hold in the interval I at all. Second, φ_2 may not be released, which means that it must hold continuously within the entire interval I. Third, φ_2 can hold continuously in the interval I, until some point in the interval where $\varphi_1 \wedge \varphi_2$ holds, thereby releasing φ_2. Finally, φ_2 can hold continuously in the interval I until the run deadlocks.

As expected, the until and release operators are dual.

Lemma 2.1. *Let* $T = (S, \longrightarrow, \mathcal{AP}, \mu)$ *be a TTS and* $s \in S$. *Then* $s \models A(\varphi_1\, R_I\, \varphi_2)$ *iff* $s \models \neg E(\neg\varphi_1\, U_I\, \neg\varphi_2)$, *and* $s \models A(\varphi_1\, U_I\, \varphi_2)$ *iff* $s \models \neg E(\neg\varphi_1\, R_I\, \neg\varphi_2)$.

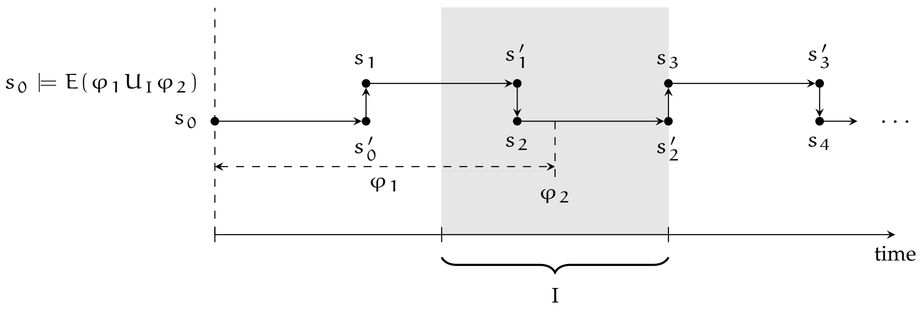

Figure 3: Illustration of a run satisfying an until formula.

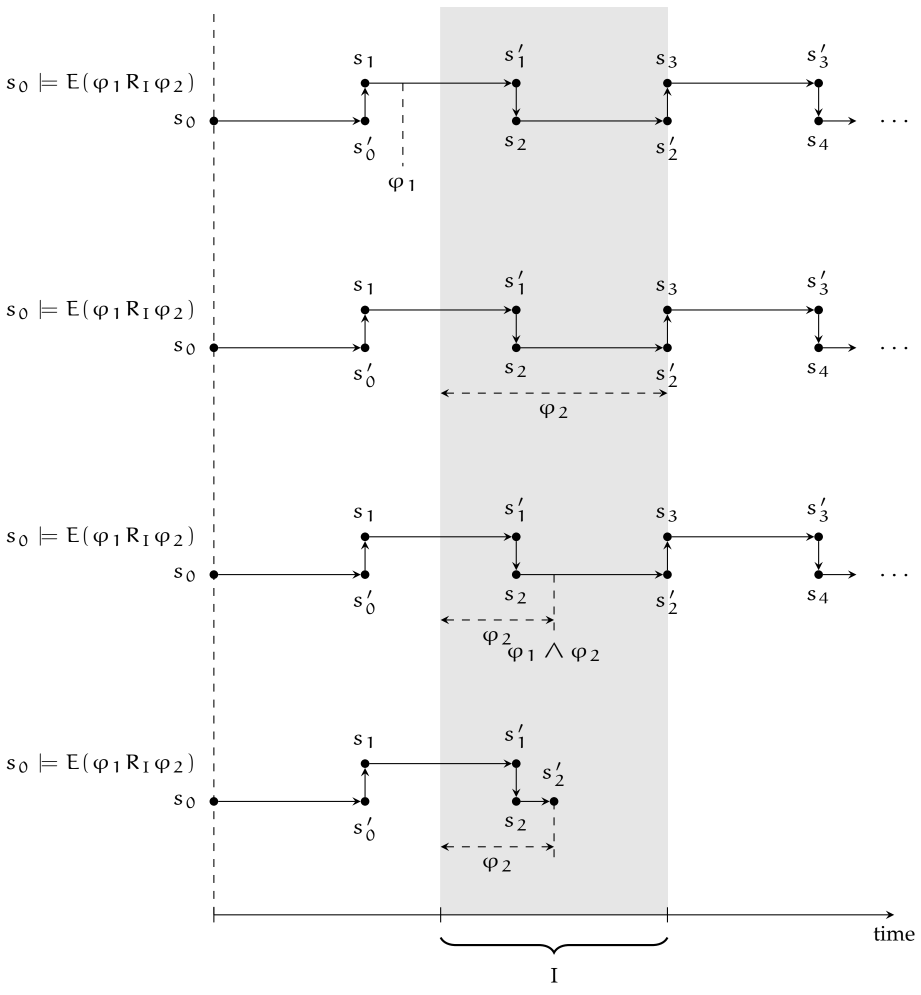

Figure 4: Illustration of runs satisfying a release formula.

Proof. We first argue for the release operator.
(Release $\Rightarrow$): Assume that $s \models A(\varphi_1 \, R_I \, \varphi_2)$. We will show that $s \models \neg E(\neg\varphi_1 \, U_I \, \neg\varphi_2)$. Assume by contradiction that $s \models E(\neg\varphi_1 \, U_I \, \neg\varphi_2)$. This means that there exists a maximal run ρ starting from s such that there exists an $i \geq 0$ and a $d \in \mathbb{R}_{\geq 0}$ such that $\mathit{valid}_\rho(i, d, I)$ is true, $s_i[d] \models \neg\varphi_2$ and for all $(j, d') \in \mathit{history}_\rho(i, d)$ it holds that $s_j[d'] \models \neg\varphi_1$.
This is a contradiction to the assumption that $s \models A(\varphi_1 \, R_I \, \varphi_2)$ which by definition means that for all maximal runs ρ' starting from s, all $i \geq 0$ and all $d \in \mathbb{R}_{\geq 0}$ it holds that $\mathit{valid}_{\rho'}(i, d, I)$ implies that either $s_i[d] \models \varphi_2$ or there exists a $(j, d') \in \mathit{history}_{\rho'}(i, d)$ such that $s_j[d'] \models \varphi_1$. Thus, it follows that $s \models A(\varphi_1 \, R_I \, \varphi_2)$ implies $s \models \neg E(\neg\varphi_1 \, U_I \, \neg\varphi_2)$.
(Release $\Leftarrow$): Assume that $s \models \neg E(\neg\varphi_1 \, U_I \, \neg\varphi_2)$. This means that for all maximal runs ρ starting from s, all $i \geq 0$ and all $d \in \mathbb{R}_{\geq 0}$ it holds that either $\mathit{valid}_\rho(i, d, I)$ is not true or $s_i[d] \not\models \neg\varphi_2$ or there exist a $(j, d') \in \mathit{history}_\rho(i, d)$ such that $s_j[d'] \not\models \neg\varphi_1$. By removing the double negations we see that this matches exactly the definition of $s \models A(\varphi_1 \, R_I \, \varphi_2)$ which says that for all maximal runs ρ starting from s, all $i \geq 0$ and all $d \in \mathbb{R}_{\geq 0}$ it holds that if $\mathit{valid}_\rho(i, d, I)$ is true, then either $s_i[d] \models \varphi_2$ or there exists a $(j, d') \in \mathit{history}_\rho(i, d)$ such that $s_j[d'] \models \varphi_1$. Thus, we have $\neg E(\neg\varphi_1 \, U_I \, \neg\varphi_2)$ implies $s \models A(\varphi_1 \, R_I \, \varphi_2)$.

Now we shall argue for the until operator.
(Until $\Rightarrow$): Assume that $s \models A(\varphi_1 \, U_I \, \varphi_2)$. We will show that $s \models \neg E(\neg\varphi_1 \, R_I \, \neg\varphi_2)$. Assume by contradiction that $s \models E(\neg\varphi_1 \, R_I \, \neg\varphi_2)$, this means that there exists a maximal run ρ starting from s s.t. for all $i \geq 0$ and for all $d \in \mathbb{R}_{\geq 0}$ if $\mathit{valid}_\rho(i, d, I)$ is true, then it holds that $s_i[d] \models \neg\varphi_2$ or there exist a $(j, d') \in \mathit{history}_\rho(i, d)$ s.t. $s_j[d'] \models \neg\varphi_1$. This is a contradiction to the assumption that $s \models A(\varphi_1 \, U_I \, \varphi_2)$, which by definition means that for every maximal run ρ' starting from s there exists and $i \geq 0$ and a $d \in \mathbb{R}_{\geq 0}$ such that $\mathit{valid}_{\rho'}(i, d, I)$ is true, $s_i[d] \models \varphi_2$ and for all $(j, d') \in \mathit{history}_{\rho'}(i, d)$ it holds that $s_j[d'] \models \varphi_1$. Then it follows that $s \models \neg E(\neg\varphi_1 \, R_I \, \neg\varphi_2)$ and thus $s \models A(\varphi_1 \, U_I \, \varphi_2)$ implies $s \models \neg E(\neg\varphi_1 \, R_I \, \neg\varphi_2)$
(Until $\Leftarrow$): Assume by contraposition that $s \not\models A(\varphi_1 \, U_I \, \varphi_2)$. By definition this means that there exists a maximal run ρ starting from s s.t. for all $i \geq 0$ and all $d \in \mathbb{R}_{\geq 0}$ either $\mathit{valid}_\rho(i, d, I)$ is not true or $s_i[d] \models \neg\varphi_2$ or there exist a $(j, d') \in \mathit{history}_\rho(i, d)$ s.t. $s_j[d'] \models \neg\varphi_1$. This matches exactly the definition of $s \models E(\neg\varphi_1 \, R_I \, \neg\varphi_2)$ which says that there exists a maximal run ρ starting from s s.t. for all $i \geq 0$ and all $d \in \mathbb{R}_{\geq 0}$ if $\mathit{valid}_\rho(i, d, I)$ is true then either $s_i[d] \models \neg\varphi_2$ or there exists $(j, d') \in \mathit{history}_\rho(i, d)$ s.t. $s_j[d'] \models \neg\varphi_1$. Thus we have that $s \models \neg E(\neg\varphi_1 \, R_I \, \neg\varphi_2)$ implies $s \models A(\varphi_1 \, U_I \, \varphi_2)$. $\square$

3 Framework Description

In this section, we shall present a general framework for arguing when a simulation of one time dependent system by another preserves satisfiability of TCTL formulae. We define the notion of one-by-many correspondence, a relation between two TTSs A and B. If A is in one-by-many correspondence with B then every transition in A can be simulated by a sequence of transitions in B. Further, every TCTL formula φ can be algorithmically translated into a formulate $tr(\varphi)$ s.t. $A \models \varphi$ iff $B \models tr(\varphi)$. In the rest of this section, we shall use A and B to refer to the original and the translated system, respectively.

3.1 One-By-Many Correspondence

As the system B is simulating a single transition of A by a sequence of transitions, the systems A and B are comparable only in the states before and after this sequence was performed. We say that B is *stable* in such states and introduce a fresh atomic proposition called *stable* to explicitly identify this situation. States that do not satisfy the proposition *stable* are called *intermediate states*. We now define three conditions that B should possess in order to apply to our framework. The third condition is optional and necessary only for the preservation of liveness TCTL properties. A TTS $(S, \to, \mathcal{AP}, \mu)$ s.t. $stable \in \mathcal{AP}$ is

- *delay-implies-stable* if for any $s \in S$, it holds that $s \xrightarrow{d}$ for some $d > 0$ implies $s \models stable$,
- *delay-preserves-stable* if for any $s \in S$ such that $s \models stable$, if $s \xrightarrow{d} s[d]$ then $s[d] \models stable$ for all $d \in \mathbb{R}_{\geq 0}$, and
- *eventually-stable* if for any $s_0 \in S$ such that $s_0 \models stable$ and for any infinite sequence of discrete transitions $\rho = s_0 \longrightarrow s_1 \longrightarrow s_2 \longrightarrow s_3 \longrightarrow s_4 \longrightarrow \ldots$ or any finite nonempty sequence of discrete transitions $\rho = s_0 \longrightarrow s_1 \longrightarrow \cdots \longrightarrow s_n \not\longrightarrow$ there exists an index $i \geq 1$ such that $s_i \models stable$. We call such a sequence a *maximal discrete sequence*.

We write $s \rightsquigarrow s'$ if there is an alternating sequence $s = s_0 \longrightarrow s_1 \xrightarrow{0} s_1 \longrightarrow s_2 \xrightarrow{0} s_2 \longrightarrow \cdots \xrightarrow{0} s_{n-1} \longrightarrow s_n = s'$ such that $s \models stable$, $s' \models stable$, and $s_j \not\models stable$ for $1 \leq j \leq n-1$.

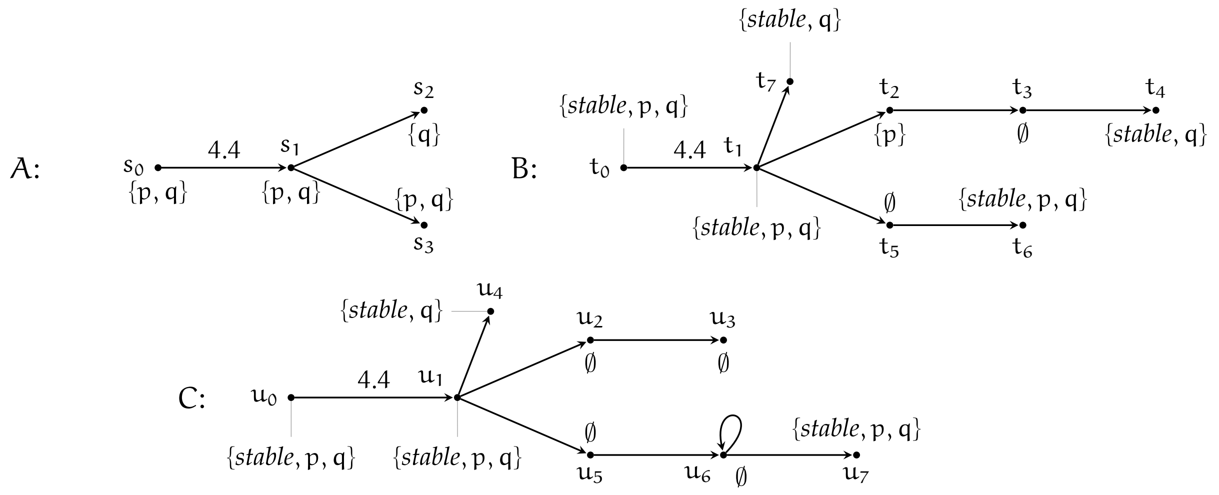

Figure 5: Three TTSs such that $s_0 \mathrel{\risingdotseq}_c t_0$ and $s_0 \risingdotseq u_0$.

Remark 3.1. *For technical convenience, we introduced zero delays in the definition of $\rightsquigarrow$ in order to preserve the alternating nature of the sequence. Note that this is not restrictive as for any $s \in S$ we always have $s \xrightarrow{0} s$.*

Definition 3.2. *Let $A = (S, \rightarrow_A, \mathcal{AP}_A, \mu_A)$ and $B = (T, \rightarrow_B, \mathcal{AP}_B, \mu_B)$ be two TTSs s.t. stable $\in \mathcal{AP}_B$ and B is* delay-implies-stable *and* delay-preserves-stable *TTS. A relation $\mathcal{R} \subseteq S \times T$ is a* one-by-many correspondence *if there exists a function $\mathrm{tr}_p : \mathcal{AP}_A \longrightarrow \mathcal{AP}_B$ such that whenever $s\,\mathcal{R}\,t$ then*

1. *$t \models stable$,*
2. *$s \models \wp$ iff $t \models \mathrm{tr}_p(\wp)$ for all $\wp \in \mathcal{AP}_A$,*
3. *if $s \longrightarrow s'$ then $t \rightsquigarrow t'$ and $s'\,\mathcal{R}\,t'$,*
4. *if $s \xrightarrow{d} s[d]$ then $t \xrightarrow{d} t[d]$ and $s[d]\,\mathcal{R}\,t[d]$ for all $d \in \mathbb{R}_{\geq 0}$,*
5. *if $t \rightsquigarrow t'$ then $s \longrightarrow s'$ and $s'\,\mathcal{R}\,t'$ and*
6. *if $t \xrightarrow{d} t[d]$ then $s \xrightarrow{d} s[d]$ and $s[d]\,\mathcal{R}\,t[d]$ for all $d \in \mathbb{R}_{\geq 0}$.*

If B is moreover an eventually-stable *TTS, then we say that $\mathcal{R}$ is a* complete one-by-many correspondence. *We write $s \risingdotseq t$ (resp. $s \risingdotseq_c t$) if there exists a relation $\mathcal{R}$ which is a one-by-many correspondence (resp. a complete one-by-many correspondence) such that $s\,\mathcal{R}\,t$.*

Example 3.3. *Consider the TTSs* A, B *and* C *in Figure 5 where the sets of propositions for* A, B *and* C *are* $\mathcal{AP}_A = \{p, q\}$ *and* $\mathcal{AP}_B = \mathcal{AP}_C = \{p, q, \mathit{stable}\}$*. Then* $\{(s_0[d], t_0[d]) \mid 0 \le d \le 4.4\} \cup \{(s_1, t_1), (s_2, t_4), (s_3, t_6), (s_2, t_7)\}$ *is a complete one-by-many correspondence which implies that* $s_0 \mathrel{\underline{\underline{\triangleright}}}_c t_0$ *and* $\{(s_0[d], u_0[d]) \mid 0 \le d \le 4.4\} \cup \{(s_1, u_1), (s_2, u_4), (s_3, u_7)\}$ *is a one-by-many correspondence which implies that* $s_0 \mathrel{\underline{\underline{\triangleright}}} u_0$*. Notice that the system* C *is not* eventually-stable *since the two maximal discrete sequences* $u_1 \longrightarrow u_5 \longrightarrow u_6 \longrightarrow u_6 \longrightarrow u_6 \longrightarrow u_6 \longrightarrow \cdots$ *and* $u_1 \longrightarrow u_2 \longrightarrow u_3$ *do not contain any stable states except for* u_1*.*

Consider now the maximal run $\rho = s_0 \xrightarrow{4.4} s_1 \longrightarrow s_2 \xrightarrow{0^{\le}}$ *in the system* A*. This run witnesses that* $s_0 \models E(\neg q \, R_{[3,5]} \, q)$*. Similarly, the maximal run* $\rho' = t_0 \xrightarrow{4.4} t_1 \rightsquigarrow t_4 \xrightarrow{0^{\le}}$ *witnesses that* $t_0 \models E((\neg q \wedge \mathit{stable}) \, R_{[3,5]} \, (q \vee \neg \mathit{stable}))$*.* □

Before we can prove the main theorem, we need to introduce some notation.

Definition 3.4. *Let* $A = (S, \rightarrow_A, \mathcal{AP}_A, \mu_A)$ *and* $B = (T, \rightarrow_B, \mathcal{AP}_B, \mu_B)$ *be two TTSs, where* $\mathit{stable} \in \mathcal{AP}_B$*. For two finite alternating runs* ρ *in* A *and* ρ' *in* B *of the form*

$$\rho = s_0 \xrightarrow{d_0} s_0[d_0] \longrightarrow s_1 \xrightarrow{d_1} s_1[d_1] \longrightarrow \cdots \longrightarrow s_n \xrightarrow{d_n} s_n[d_n]$$
$$\rho' = t_0 \xrightarrow{d_0} t_0[d_0] \rightsquigarrow t_1 \xrightarrow{d_1} t_1[d_1] \rightsquigarrow \cdots \rightsquigarrow t_n \xrightarrow{d_n} t_n[d_n]$$

we write $\rho \mathrel{\underline{\underline{\triangleright}}} \rho'$ *if* $s_i[d] \mathrel{\underline{\underline{\triangleright}}} t_i[d]$ *for all* $i \le n$ *and all* $d \le d_i$*.*

Now we define when two maximal runs are related w.r.t. $\mathrel{\underline{\underline{\triangleright}}}$.

Definition 3.5. *Let* $A = (S, \rightarrow_A, \mathcal{AP}_A, \mu_A)$ *and* $B = (T, \rightarrow_B, \mathcal{AP}_B, \mu_B)$ *be two TTSs, where* $\mathit{stable} \in \mathcal{AP}_B$*. For two maximal runs* ρ *in* A *and* ρ' *in* B *we write* $\rho \mathrel{\underline{\underline{\triangleright}}} \rho'$ *if*

- ρ *is an infinite maximal run*

  $$\rho = s_0 \xrightarrow{d_0} s_0[d_0] \longrightarrow s_1 \xrightarrow{d_1} s_1[d_1] \longrightarrow s_2 \xrightarrow{d_2} s_2[d_2] \longrightarrow \ldots,$$

  ρ' *is an infinite maximal run*

  $$\rho' = t_0 \xrightarrow{d_0} t_0[d_0] \rightsquigarrow t_1 \xrightarrow{d_1} t_1[d_1] \rightsquigarrow t_2 \xrightarrow{d_2} t_2[d_2] \rightsquigarrow \ldots,$$

  and $s_i[d] \mathrel{\underline{\underline{\triangleright}}} t_i[d]$ *for all* $i \ge 0$ *and all* $d \le d_i$*, or*

- ρ *is a finite maximal run of the form*

  $$\rho = s_0 \xrightarrow{d_0} s_0[d_0] \longrightarrow s_1 \xrightarrow{d_1} s_1[d_1] \longrightarrow s_2 \xrightarrow{d_2} s_2[d_2] \longrightarrow \ldots \longrightarrow s_n \xrightarrow{\delta},$$

  ρ' *is a finite maximal run of the form*

  $$\rho' = t_0 \xrightarrow{d_0} t_0[d_0] \rightsquigarrow t_1 \xrightarrow{d_1} t_1[d_1] \rightsquigarrow t_2 \xrightarrow{d_2} t_2[d_2] \rightsquigarrow \ldots \longrightarrow t_n \xrightarrow{\delta},$$

  for some $\delta \in \{\infty, d_n^{\le}, d_n^{<}\}$ *such that,*

- $s_i[d] \Bumpeq t_i[d]$ *for all* $i < n$ *and all* $d \leq d_i$,
- $s_n[d] \Bumpeq t_n[d]$ *for all* $d \in \mathbb{R}_{\geq 0}$ *if* $\delta = \infty$,
- $s_n[d] \Bumpeq t_n[d]$ *for all* $d \leq d_n$ *if* $\delta = d_n^{\leq}$ *and*
- $s_n[d] \Bumpeq t_n[d]$ *for all* $d < d_n$ *if* $\delta = d_n^{<}$.

For the rest of this part let us fix two TTS $A = (S, \rightarrow_A, \mathcal{AP}_A, \mu_A)$ and $B = (T, \rightarrow_B, \mathcal{AP}_B, \mu_B)$ such that *stable* $\in \mathcal{AP}_B$ and B has the properties *delay-implies-stable* and *delay-preserves-stable*.

Lemma 3.6. *Let* $s_0 \in S$ *and* $t_0 \in T$ *be such that* $s_0 \Bumpeq t_0$. *Then there exists a finite run*

$$\rho = s_0 \xrightarrow{d_0} s_0[d_0] \longrightarrow s_1 \xrightarrow{d_1} s_1[d_1] \longrightarrow \cdots \longrightarrow s_n \xrightarrow{d_n} s_n[d_n]$$

in A *if and only if there exists a finite run*

$$\rho' = t_0 \xrightarrow{d_0} t_0[d_0] \leadsto t_1 \xrightarrow{d_1} t_1[d_1] \leadsto \cdots \leadsto t_n \xrightarrow{d_n} t_n[d_n]$$

in B *such that* $\rho \Bumpeq \rho'$.

Proof. ($\Rightarrow$): Assume that there exists a run $\rho = s_0 \xrightarrow{d_0} s_0[d_0] \longrightarrow s_1 \xrightarrow{d_1} s_1[d_1] \longrightarrow \cdots \longrightarrow s_n \xrightarrow{d_n} s_n[d_n]$ in A. By induction on i and using condition 3 and 4 of Definition 3.2 we can construct a run $\rho' = t_0 \xrightarrow{d_0} t_0[d_0] \leadsto t_1 \xrightarrow{d_1} t_1[d_1] \leadsto \cdots \leadsto t_n \xrightarrow{d_n} t_n[d_n]$ in B.

($\Leftarrow$): Assume that there exists a run $\rho' = t_0 \xrightarrow{d_0} t_0[d_0] \leadsto t_1 \xrightarrow{d_1} t_1[d_1] \leadsto \cdots \leadsto t_n \xrightarrow{d_n} t_n[d_n]$ in B. By induction on i and using condition 5 and 6 of Definition 3.2 we can construct a run $\rho = s_0 \xrightarrow{d_0} s_0[d_0] \longrightarrow s_1 \xrightarrow{d_1} s_1[d_1] \longrightarrow \cdots \longrightarrow s_n \xrightarrow{d_n} s_n[d_n]$ in A. $\square$

Notice that Lemma 3.6 deals only with finite runs and moreover requires that the run in B ends in a stable state (see definition of $\leadsto$ and the property of TTS *delay-preserves-stable* in the main text). However, there may still exist finite runs in B for which there is no related run in A. The next lemma considers maximal runs and it is a consequence of the definition of complete one-by-many correspondence.

Lemma 3.7. *Let* $s_0 \in S$ *and* $t_0 \in T$ *be such that* $s_0 \Bumpeq_c t_0$. *Then there exists a maximal run* $\rho \in$ *MaxRuns*(A, s_0) *if and only if there exists a maximal run* $\rho' \in$ *MaxRuns*(B, t_0) *such that* $\rho \Bumpeq_c \rho'$.

Proof. ($\Rightarrow$): We shall prove this lemma in two steps, first for infinite maximal runs and then for finite maximal runs. Let

$$\rho = s_0 \xrightarrow{d_0} s_0[d_0] \longrightarrow s_1 \xrightarrow{d_1} s_1[d_1] \longrightarrow s_2 \xrightarrow{d_2} s_2[d_2] \longrightarrow \dots,$$

be any infinite maximal run in A. Since $s_0 \overset{\rightarrow}{=}_c t_0$, we have by induction on the index i of states and using condition 3 and 4 of Definition 3.2 that there exists an infinite maximal run ρ' in B of the form

$$\rho' = t_0 \xrightarrow{d_0} t_0[d_0] \rightsquigarrow t_1 \xrightarrow{d_1} t_1[d_1] \rightsquigarrow t_2 \xrightarrow{d_2} t_2[d_2] \rightsquigarrow \dots,$$

where $s_i[d] \overset{\rightarrow}{=}_c t_i[d]$ for all $i \geq 0$ and all $d \leq d_i$. Thus, $\rho \overset{\rightarrow}{=}_c \rho'$.

Now let

$$\rho = s_0 \xrightarrow{d_0} s_0[d_0] \longrightarrow s_1 \xrightarrow{d_1} s_1[d_1] \longrightarrow s_2 \xrightarrow{d_2} s_2[d_2] \longrightarrow \dots \longrightarrow s_n \xrightarrow{\delta},$$

be a finite maximal run in A where $\delta \in \{\infty, d_n^{\leq}, d_n^{<}\}$.

Since $s_0 \overset{\rightarrow}{=}_c t_0$, we shall construct a finite maximal run in B of the form

$$\rho' = t_0 \xrightarrow{d_0} t_0[d_0] \rightsquigarrow t_1 \xrightarrow{d_1} t_1[d_1] \rightsquigarrow t_2 \xrightarrow{d_2} t_2[d_2] \rightsquigarrow \dots \rightsquigarrow t_n \xrightarrow{\delta}.$$

Using Lemma 3.6 and noticing that the lemma also works with runs ending with a discrete action, it follows that from the prefix of ρ up to and including s_n (referred to as $\rho_{\textit{prefix}}$), we can construct the prefix of ρ' up to and including t_n (referred to as $\rho'_{\textit{prefix}}$) such that $\rho_{\textit{prefix}} \overset{\rightarrow}{=}_c \rho'_{\textit{prefix}}$.

We shall now handle the final part of ρ' according to the value of δ in ρ.

$\delta = \infty$ In this case, $s_n \xrightarrow{d} s_n[d]$ for all $d \in \mathbb{R}_{\geq 0}$. It follows from condition 4 of Definition 3.2 that $t_n \xrightarrow{d} t_n[d]$ such that $s_n[d] \overset{\rightarrow}{=}_c t_n[d]$ for all $d \in \mathbb{R}_{\geq 0}$. Hence $\rho \overset{\rightarrow}{=}_c \rho'$.

$\delta =\leq$ In this case, $s_n \not\xrightarrow{d}$ for all $d > d_n$ and $s_n \xrightarrow{d_n} s_n[d_n]$ such that $s_n[d_n] \not\longrightarrow$. It follows from condition 6 of Definition 3.2 that $t_n \not\xrightarrow{d}$ for $d > d_n$. Since $s_n \xrightarrow{d_n} s_n[d_n]$, it follows from condition 4 of Definition 3.2 that $t_n \xrightarrow{d_n} t_n[d_n]$ such that $s_n[d_n] \overset{\rightarrow}{=}_c t_n[d_n]$. Since $s_n[d_n] \not\longrightarrow$, it follows from condition 5 of 3.2 that $t_n[d_n] \not\rightsquigarrow$. By the *delay-implies-stable* and *eventually-stable* properties of B, it follows that $t_n[d] \longrightarrow$ iff $t_n[d] \rightsquigarrow$ and thus, $t_n[d_n] \not\longrightarrow$. Hence $\rho \overset{\rightarrow}{=}_c \rho'$.

$\delta = d_n^{<}$ In this case, $s_n \not\xrightarrow{d}$ for all $d \geq d_n$ and there exists a $d_s < d_n$ such that for all d, $d_s \leq d < d_n$, we have that $s_n \xrightarrow{d} s_n[d]$ and $s_n[d] \not\longrightarrow$. It follows from condition 6 of Definition 3.2 that $t_n \overset{d}{\not\longrightarrow}$ for $d \geq d_n$. Since delays are similar in the two runs, it follows from condition 4 of Definition 3.2 that $t_n \xrightarrow{d} t_n[d]$ such that $s_n[d] \sqsupseteq_c t_n[d]$ and from condition 5 of Definition 3.2 that $t_n[d] \not\rightsquigarrow$ for all $d_s \leq d < d_n$. Moreover, by the *delay-implies-stable* and *eventually-stable* properties of B, it follows that $t_n[d] \longrightarrow$ iff $t_n[d] \rightsquigarrow$, hence $t_n[d] \not\longrightarrow$ for all $d_s \leq d < d_n$. Hence $\rho \sqsupseteq_c \rho'$.

($\Leftarrow$): We shall also prove this direction in two steps, first for infinite maximal runs and then for finite maximal runs. Assume that there exists an infinite maximal run in B of the form

$$\rho' = t_0 \xrightarrow{d_0} t_0[d_0] \longrightarrow t_1 \xrightarrow{d_1} t_1[d_1] \longrightarrow t_2 \xrightarrow{d_2} t_2[d_2] \longrightarrow \cdots$$

where intermediate states are also indexed.

Let $j_1 < j_2 < j_3 < \ldots$ be all the indices such that $t_{j_i} \models$ *stable* for all $i > 0$. Since B is an *eventually-stable* TTS there are infinitely many stable states in ρ'. Moreover, since B is a *delay-implies-stable* TTS, it follows that if $t_i \not\models$ *stable* then $d_i = 0$ for all $i > 0$. Finally, because B is also a *delay-preserves-stable* TTS, it follows that whenever $t_i \models$ *stable* then $t_i[d_i] \models$ *stable* for all $i \geq 0$. These properties in turn imply that we can write ρ' in the form

$$\rho' = t_0 \xrightarrow{d_0} t_0[d_0] \rightsquigarrow t_{j_1} \xrightarrow{d_{j_1}} t_{j_1}[d_{j_1}] \rightsquigarrow t_{j_2} \xrightarrow{d_{j_2}} t_{j_2}[d_{j_2}] \rightsquigarrow \cdots$$

Now, following the similar strategy as in the ($\Rightarrow$) case, we can construct a infinite maximal run $\rho \in \mathit{MaxRuns}(A, s_0)$ such that $\rho \sqsupseteq_c \rho'$.

Now assume that there exists a finite maximal run in B of the form

$$\rho' = t_0 \xrightarrow{d_0} t_0[d_0] \longrightarrow t_1 \xrightarrow{d_1} t_1[d_1] \longrightarrow t_2 \xrightarrow{d_2} t_2[d_2] \longrightarrow \cdots \longrightarrow t_n \xrightarrow{\delta}$$

where intermediate states are also indexed and $\delta = \{\infty, d_n^{<}, d_n^{\leq}\}$.

Let $j_1 < j_2 < j_3 < \ldots < j_k$ be all the indices such that $t_{j_i} \models$ *stable* for all $0 < i \leq k$. Since B is an *eventually-stable* TTS, we know that $j_k = n$. Further, because B is a *delay-implies-stable* and *delay-preserves-stable* TTS, we can write ρ' in the form

$$\rho' = t_0 \xrightarrow{d_0} t_0[d_0] \rightsquigarrow t_{j_1} \xrightarrow{d_{j_1}} t_{j_1}[d_{j_1}] \rightsquigarrow \cdots \rightsquigarrow t_{j_k} \xrightarrow{\delta} .$$

Now, following the similar strategy as in the ($\Rightarrow$) case, we can construct a finite maximal run $\rho \in \mathit{MaxRuns}(A, s_0)$ which also ends with $\xrightarrow{\delta}$ such that $\rho \sqsupseteq_c \rho'$. $\square$

Now we translate TCTL formulae. Let $\mathcal{AP}_A$ and $\mathcal{AP}_B$ be sets of atomic propositions such that *stable* $\in \mathcal{AP}_B$ and let $tr_p : \mathcal{AP}_A \longrightarrow \mathcal{AP}_B$ be a function translating atomic propositions. We define $\mathrm{tr} : \Phi(\mathcal{AP}_A) \to \Phi(\mathcal{AP}_B)$ as follows:

$$\begin{aligned}
\mathrm{tr}(\wp) &= \mathrm{tr}_p(\wp)\\
\mathrm{tr}(\neg\varphi_1) &= \neg\mathrm{tr}(\varphi_1)\\
\mathrm{tr}(\varphi_1 \wedge \varphi_2) &= \mathrm{tr}(\varphi_1) \wedge \mathrm{tr}(\varphi_2)\\
\mathrm{tr}(E(\varphi_1\, U_I\, \varphi_2)) &= E((\mathrm{tr}(\varphi_1) \vee \neg \mathit{stable})\, U_I\, (\mathrm{tr}(\varphi_2) \wedge \mathit{stable}))\\
\mathrm{tr}(A(\varphi_1\, U_I\, \varphi_2)) &= A((\mathrm{tr}(\varphi_1) \vee \neg \mathit{stable})\, U_I\, (\mathrm{tr}(\varphi_2) \wedge \mathit{stable}))\\
\mathrm{tr}(E(\varphi_1\, R_I\, \varphi_2)) &= E((\mathrm{tr}(\varphi_1) \wedge \mathit{stable})\, R_I\, (\mathrm{tr}(\varphi_2) \vee \neg \mathit{stable}))\\
\mathrm{tr}(A(\varphi_1\, R_I\, \varphi_2)) &= A((\mathrm{tr}(\varphi_1) \wedge \mathit{stable})\, R_I\, (\mathrm{tr}(\varphi_2) \vee \neg \mathit{stable}))
\end{aligned}$$

We are now ready to state the main result of this section.

Theorem 3.8. *Let* $A = (S, \to_A, \mathcal{AP}_A, \mu_A)$ *and* $B = (T, \to_B, \mathcal{AP}_B, \mu_B)$ *be two TTSs such that stable* $\in \mathcal{AP}_B$ *and let* $s_0 \in S$ *and* $t_0 \in T$. *If* $s_0 \sqsupseteq_c t_0$ *then for any TCTL formula* φ, $s_0 \models \varphi$ *if and only if* $t_0 \models \mathrm{tr}(\varphi)$. *If* $s_0 \sqsupseteq t_0$ *then the claim holds only for any formula* φ *from the safety fragment of TCTL.*

Proof. We shall prove this theorem by structural induction on φ. By Lemma 2.1 it is sufficient to handle the operators $\wp$, $\neg\varphi$, $\varphi_1 \wedge \varphi_2$, $E(\varphi_1\, U_I\, \varphi_2)$ and $E(\varphi_1\, R_I\, \varphi_2)$.

- $\varphi = \wp$, $\varphi = \neg\varphi_1$, and $\varphi = \varphi_1 \wedge \varphi_2$ follow trivially from the induction hypothesis.
- $\varphi = E(\varphi_1\, U_I\, \varphi_2)$:

  $(\Rightarrow)$: Assume that $s_0 \models_A E(\varphi_1\, U_I\, \varphi_2)$. Thus there exists a maximal run $\rho \in \mathit{MaxRuns}(A, s_0)$ that witnesses φ. This implies that there exists a prefix of ρ of the form

  $$\begin{aligned}\rho_{\mathit{prefix}} = s_0 \xrightarrow{d_0} s_0[d_0] \longrightarrow s_1 \xrightarrow{d_1} s_1[d_1] \longrightarrow \cdots \longrightarrow s_{n-1}\\ \xrightarrow{d_{n-1}} s_{n-1}[d_{n-1}] \longrightarrow s_n \xrightarrow{d} s_n[d],\end{aligned}$$

  such that $\mathit{valid}_\rho(n, d, I)$, $s_n[d] \models_A \varphi_2$ and $s_k[d'] \models_A \varphi_1$ for all $(k, d') \in \mathit{history}_\rho(n, d)$. By Lemma 3.6 there exists a run ρ'_{prefix} in B of the form

  $$\begin{aligned}\rho'_{\mathit{prefix}} = t_0 \xrightarrow{d_0} t_0[d_0] \rightsquigarrow t_1 \xrightarrow{d_1} t_1[d_1] \rightsquigarrow \cdots \rightsquigarrow t_{n-1}\\ \xrightarrow{d_{n-1}} t_{n-1}[d_{n-1}] \rightsquigarrow t_n \xrightarrow{d} t_n[d],\end{aligned}$$

  such that $\rho_{\mathit{prefix}} \sqsupseteq_c \rho'_{\mathit{prefix}}$.

We want to show that $t_0 \models_B E((tr(\varphi_1) \vee \neg \mathit{stable})\, U_I\, (tr(\varphi_2) \wedge \mathit{stable}))$. Because $s_n[d] \mathrel{\underline{\underline{\triangleright}}}_c t_n[d]$ and $s_n[d] \models_A \varphi_2$, we have by the induction hypothesis that $t_n[d] \models_B tr(\varphi_2)$ and from condition 1 of Definition 3.2 we have that $t_n[d] \models_B \mathit{stable}$. Let j be the index corresponding to the occurrence of t_n in the alternating sequence which unfolds the $\rightsquigarrow$ steps. Because time delays are equivalent in the two runs, it follows that $\mathit{valid}_{\rho'}(j, d, I)$ and moreover, for any pair $(k, d') \in \mathit{history}_{\rho'}(j, d)$ either,

- $t_k[d']$ is an intermediate state and thus $t_k[d'] \models_B \neg \mathit{stable}$, or
- $t_k[d']$ is a stable state. From the construction of ρ'_{prefix} it follows that there exists a pair $(k', d') \in \mathit{history}_\rho(n, d)$ such that $s_{k'}[d'] \mathrel{\underline{\underline{\triangleright}}}_c t_k[d']$. By the induction hypothesis and the fact that $s_{k'}[d'] \models_A \varphi_1$, it follows that $t_k[d'] \models_B tr(\varphi_1)$.

This means that $t_k[d'] \models_B tr(\varphi_1) \vee \neg \mathit{stable}$.

Thus any maximal run $\rho' \in \mathit{MaxRuns}(B, t_0)$ that extends ρ'_{prefix} witnesses $tr(\varphi)$, meaning that $t_0 \models_B E((tr(\varphi_1) \vee \neg \mathit{stable})\, U_I\, (tr(\varphi_2) \wedge \mathit{stable}))$.

$(\Leftarrow)$: Assume $t_0 \models_B E((tr(\varphi_1) \vee \neg \mathit{stable})\, U_I\, (tr(\varphi_2) \wedge \mathit{stable}))$. Thus there exists a maximal run ρ' in B that witnesses $tr(\varphi)$. By the fact that B is a *delay-implies-stable* and *delay-preserves-stable* TTS there exists a prefix of ρ' of the form

$$\rho'_{\mathit{prefix}} = t_0 \xrightarrow{d_0} t_0[d_0] \rightsquigarrow t_1 \xrightarrow{d_1} t_1[d_1] \rightsquigarrow \cdots \rightsquigarrow t_{n-1} \xrightarrow{d_{n-1}} t_{n-1}[d_{n-1}] \rightsquigarrow t_n \xrightarrow{d} t_n[d],$$

such that $\mathit{valid}_{\rho'}(j, d, I)$, $t_n[d] \models_B tr(\varphi_2) \wedge \mathit{stable}$ and $t_k[d'] \models_B tr(\varphi_1) \vee \neg \mathit{stable}$ for all $(k, d') \in \mathit{history}_{\rho'}(j, d)$ where j is the index corresponding to the occurrence of t_n in the alternating sequence which unfolds the $\rightsquigarrow$ steps as before.

By Lemma 3.6 there exists a run ρ_{prefix} in A of the form

$$\rho_{\mathit{prefix}} = s_0 \xrightarrow{d_0} s_0[d_0] \longrightarrow s_1 \xrightarrow{d_1} s_1[d_1] \longrightarrow \cdots \longrightarrow s_{n-1} \xrightarrow{d_{n-1}} s_{n-1}[d_{n-1}] \longrightarrow s_n \xrightarrow{d} s_n[d],$$

such that $\rho_{\mathit{prefix}} \mathrel{\underline{\underline{\triangleright}}}_c \rho'_{\mathit{prefix}}$.

We want to show that $s_0 \models_A E(\varphi_1\, U_I\, \varphi_2)$. Because $s_n[d] \mathrel{\underline{\underline{\triangleright}}}_c t_n[d]$ and $t_n[d] \models_B tr(\varphi_2) \wedge \mathit{stable}$, we have by the induction hypothesis that $s_n[d] \models_A \varphi_2$. Since time delays are equivalent in the two runs, it follows that $\mathit{valid}_\rho(n, d, I)$. Further, for any pair $(k', d') \in \mathit{history}_\rho(n, d)$, it follows that there exists a $(k, d') \in \mathit{history}_{\rho'}(j, d)$ such that $s_{k'}[d'] \supseteq_c t_k[d']$. By the induction hypothesis, it follows that $s_{k'}[d'] \models_A \varphi_1$ because $t_k[d'] \models_B \mathrm{tr}(\varphi_1)$ and $t_k[d'] \models_B \mathit{stable}$.

It then follows that any maximal run $\rho \in \mathit{MaxRuns}(A, s_0)$ starting with ρ_{prefix} witnesses φ, thus $s_0 \models_A E(\varphi_1 \, U_I \, \varphi_2)$.

- $\varphi = E(\varphi_1 \, R_I \, \varphi_2)$:

  $(\Rightarrow)$: Assume that $s_0 \models_A E(\varphi_1 \, R_I \, \varphi_2)$. By assumption, there exists a maximal run (infinite or finite)

  $$\rho = s_0 \xrightarrow{d_0} s_0[d_0] \longrightarrow s_1 \xrightarrow{d_1} s_1[d_1] \longrightarrow s_2 \xrightarrow{d_2} s_2[d_2] \longrightarrow \cdots$$

  in A that witnesses φ. This means that for all $i \geq 0$ and all $d \in \mathbb{R}_{\geq 0}$ if $\mathit{valid}_\rho(i, d, I)$ is true then either $s_i[d] \models_A \varphi_2$ or there exists a $(k, d') \in \mathit{history}_\rho(i, d)$ s.t. $s_k[d'] \models_A \varphi_1$.

  By Lemma 3.7 it follows that there exists a maximal run

  $$\rho' = t_0 \xrightarrow{d_0} t_0[d_0] \rightsquigarrow t_1 \xrightarrow{d_1} t_1[d_1] \rightsquigarrow t_2 \xrightarrow{d_2} t_2[d_2] \rightsquigarrow \cdots \tag{1}$$

  in B such that $\rho \supseteq_c \rho'$ (see Definition 3.5).

  We want to show that $t_0 \models_B E((\mathrm{tr}(\varphi_1) \wedge \mathit{stable}) \, R_I \, (\mathrm{tr}(\varphi_2) \vee \neg \mathit{stable}))$. For all $i \geq 0$ and all $d \in \mathbb{R}_{\geq 0}$, if $\mathit{valid}_{\rho'}(i, d, I)$ is true then either

  - $t_i[d]$ is an intermediate state and then $t_i[d] \models_B \neg\mathit{stable}$, or
  - $t_i[d]$ is a stable state. This means that $t_i[d]$ has a distinguished index in Equation 1. Let h be the index of $t_i[d]$ in Equation 1. It follows from the construction of ρ' that $s_h[d] \supseteq_c t_h[d]$. There are two cases to consider.
    * either $s_h[d] \models_A \varphi_2$ and then by the induction hypothesis it follows that $t_h[d] \models_B \mathrm{tr}(\varphi_2)$, or
    * there exists $(\ell', d') \in \mathit{history}_\rho(h, d)$ such that $s_{\ell'}[d'] \models_A \varphi_1$. From the construction of ρ' it follows that there exists a pair $(\ell, d') \in \mathit{history}_{\rho'}(i, d)$ such that $s_{\ell'}[d'] \supseteq_c t_\ell[d']$. By the induction hypothesis, it follows that $t_\ell[d'] \models_B \mathrm{tr}(\varphi_1) \wedge \mathit{stable}$.

  This in turn means that $t_0 \models_B E((\mathrm{tr}(\varphi_1) \wedge \mathit{stable}) \, R_I \, (\mathrm{tr}(\varphi_2) \vee \neg\mathit{stable}))$.

($\Leftarrow$) : Assume that $t_0 \models_B \mathsf{E}((tr(\varphi_1) \wedge \mathit{stable})\, \mathsf{R}_I\, (tr(\varphi_2) \vee \neg \mathit{stable}))$. Hence there exists a maximal run (infinite or finite) in B that witnesses $tr(\varphi)$ and since B is a *delay-implies-stable*, *delay-preserves-stable* and *eventually-stable* TTS, the run has the following form:

$$\rho' = t_0 \xrightarrow{d_0} t_0[d_0] \rightsquigarrow t_1 \xrightarrow{d_1} t_1[d_1] \rightsquigarrow t_2 \xrightarrow{d_2} t_2[d_2] \rightsquigarrow \cdots$$

This means that for all $i \geq 0$ and all $d \in \mathbb{R}_{\geq 0}$ if $\mathit{valid}_{\rho'}(i, d, I)$ is true then either $t_i[d] \models_B tr(\varphi_2) \vee \neg \mathit{stable}$ or there exists a $(k, d') \in \mathit{history}_{\rho'}(i, d)$ s.t. $t_k[d'] \models_B tr(\varphi_1) \wedge \mathit{stable}$.

By Lemma 3.7 it follows that there exists a maximal run

$$\rho = s_0 \xrightarrow{d_0} s_0[d_0] \longrightarrow s_1 \xrightarrow{d_1} s_1[d_1] \longrightarrow s_2 \xrightarrow{d_2} s_2[d_2] \longrightarrow \cdots$$

in A such that $\rho \mathrel{\supseteq\!\!\!\!=_c} \rho'$.

We want to show that $s_0 \models_A \mathsf{E}(\varphi_1\, \mathsf{R}_I\, \varphi_2)$. For all $i \geq 0$ and all $d \in \mathbb{R}_{\geq 0}$, we have that $s_i[d] \mathrel{\supseteq\!\!\!\!=_c} t_i[d]$ and whenever $\mathit{valid}_\rho(i, d, I)$ is true then

- either $t_i[d] \models_B tr(\varphi_2)$ and then by the induction hypothesis we have that $s_i[d] \models_A \varphi_2$, or
- there exists some $(\ell', d') \in \mathit{history}_{\rho'}(j, d)$ where j is the index corresponding to the occurrence of t_i in the alternating sequence which unfolds the $\rightsquigarrow$ steps, such that $t_{\ell'}[d'] \models_B tr(\varphi_1) \wedge \mathit{stable}$. From the construction of ρ, it follows that there exists a pair $(\ell, d') \in \mathit{history}_\rho(i, d)$ such that $s_\ell[d'] \mathrel{\supseteq\!\!\!\!=_c} t_{\ell'}[d']$. By the induction hypothesis, it follows that $s_\ell[d'] \models_A \varphi_1$.

This in turn means that $s_0 \models_A \mathsf{E}(\varphi_1\, \mathsf{R}_I\, \varphi_2)$.

Observe that for the case of $\mathsf{E}(\varphi_1\, \mathsf{U}_I\, \varphi_2)$ (dually for the case $\mathsf{A}(\varphi_1\, \mathsf{R}_I\, \varphi_2)$), we used Lemma 3.6 which requires only a one-by-many correspondence. On the other hand, to prove the case of $\mathsf{E}(\varphi_1\, \mathsf{R}_I\, \varphi_2)$ (dually, $\mathsf{A}(\varphi_1\, \mathsf{U}_I\, \varphi_2)$) we used the *eventually-stable* property and Lemma 3.7, which requires a complete one-by-many correspondence. Hence, if the relation is only a one-by-many correspondence then the theorem works only for the safety fragment. □

Example 3.9. *The reason we need a complete one-by-many correspondence to preserve the full TCTL can be illustrated by considering systems* A *and* C *in Figure 5 where*

$\{(s_0, u_0), (s_1, u_1), (s_2, u_4), (s_3, u_7)\}$ *is a one-by-many correspondence between states in* A *and* C. *In this particular example,* $s_0 \models_A A(p\, \mathsf{U}_{[3,5]}\, q)$ *but* $u_0 \not\models_B \mathrm{tr}(A(p\, \mathsf{U}_{[3,5]}\, q)) = A((p \vee \neg \mathit{stable})\, \mathsf{U}_{[3,5]}\, (q \wedge \mathit{stable}))$. *Both of the following maximal runs*

$$\rho = u_0 \xrightarrow{4.4} u_1 \longrightarrow u_2 \xrightarrow{0} u_2 \longrightarrow u_3 \xrightarrow{0^{\leq}}$$
$$\rho' = u_0 \xrightarrow{4.4} u_1 \longrightarrow u_5 \xrightarrow{0} u_5 \longrightarrow u_6 \xrightarrow{0} u_6 \longrightarrow u_6 \xrightarrow{0} u_6 \longrightarrow u_6 \xrightarrow{0} \cdots$$

in C *are counter examples to* $\mathrm{tr}(A(p\, \mathsf{U}_{[3,5]}\, q))$. □

3.2 Overall Methodology

We finish this section by recalling the steps needed in order to apply the framework to a particular translation between two time-dependent systems. Assume that we designed an algorithm that for a given system A constructs a system B together with the notion of stable states in the system B.

1. Show that B is a *delay-implies-stable* and *delay-preserves-stable* TTS (and optionally an *eventually-stable* TTS).

2. Define a proposition translation function $\mathrm{tr}_p : \mathcal{AP}_A \longrightarrow \mathcal{AP}_B$.

3. Define a relation $\mathcal{R}$ and show that it fulfills conditions 1–6 of Definition 3.2.

Theorem 3.8 now allows us to conclude that the translation preserves the full TCTL (or its safety fragment if $\mathcal{R}$ is only a one-by-many correspondence).

4 Translation from Bounded TAPN to NTA

This section describes a translation from extended timed-arc Petri nets to networks of timed automata (NTA). We start with the definitions of the models.

4.1 Extended Timed-Arc Petri Nets

We shall now define timed-arc Petri nets with invariants, inhibitor arcs and transport arcs. Recall the set of time intervals $\mathcal{I}$ defined in Section 2. The predicates $r \in I$ for $I \in \mathcal{I}$ and $r \in \mathbb{R}_{\geq 0}$ are defined in the expected way. By $\mathcal{I}_{\mathrm{Inv}}$ we denote the subset of $\mathcal{I}$ of intervals containing 0 and call them *invariant intervals*.

Definition 4.1. *A* timed-arc Petri net with invariants, inhibitor arcs and transport arcs *(TAPN) is a tuple* $N = (P, T, F, c, F_{tarc}, c_{tarc}, F_{inhib}, c_{inhib}, \iota)$ *where*

- P *is a finite set of* places*,*
- T *is a finite set of* transitions *such that* $P \cap T = \emptyset$*,*
- $F \subseteq (P \times T) \cup (T \times P)$ *is a set of* normal arcs*,*
- $c : F|_{P \times T} \longrightarrow \mathcal{I}$ *assigns time intervals to arcs from places to transitions,*
- $F_{tarc} \subseteq (P \times T \times P)$ *is a set of* transport arcs *that satisfy for all* $(p, t, p') \in F_{tarc}$ *and all* $r \in P$*:* $(p, t, r) \in F_{tarc} \Rightarrow p' = r$*,* $(r, t, p') \in F_{tarc} \Rightarrow p = r$*,* $(p, t) \notin F$*, and* $(t, p') \notin F$*,*
- $c_{tarc} : F_{tarc} \longrightarrow \mathcal{I}$ *is a function assigning time intervals to transport arcs,*
- $F_{inhib} \subseteq P \times T$ *is a set of* inhibitor arcs *satisfying for all* $(p, t) \in F_{inhib}$ *and all* $p' \in P$*:* $(p, t) \notin F$ *and* $(p, t, p') \notin F_{tarc}$*,*
- $c_{inhib} : F_{inhib} \longrightarrow \mathcal{I}$ *assigns time intervals to inhibitor arcs, and*
- $\iota : P \longrightarrow \mathcal{I}_{Inv}$ *is a function assigning* invariants *to places.*

The *preset* of $t \in T$ is defined as ${}^\bullet t = \{p \in P \mid (p, t) \in F \vee \exists p' \in P \,.\, (p, t, p') \in F_{tarc}\}$ and the *postset* of t is $t^\bullet = \{p \in P \mid (t, p) \in F \vee \exists p' \in P \,.\, (p', t, p) \in F_{tarc}\}$.

A *marking* on a TAPN N is a function $M : P \longrightarrow \mathcal{B}(\mathbb{R}_{\geq 0})$, where $\mathcal{B}(\mathbb{R}_{\geq 0})$ is the set of finite multisets of non-negative real numbers s.t. for every place $p \in P$ and every token $x \in M(p)$ it holds that $x \in \iota(p)$. The set of all markings on N is denoted by $\mathcal{M}(N)$. Note that in TAPN each token has its own age. A *marked* TAPN is a pair (N, M_0) where N is a TAPN and M_0 is an *initial marking* on N with all tokens of age 0. A transition $t \in T$ is *enabled* in marking M if

- for all $p \in {}^\bullet t$ s.t. $(p, t) \in F$ there is a token x of an age in the time interval on the arc from p to t: $\forall p \in {}^\bullet t$ s.t. $(p, t) \in F \,.\, \exists x \in M(p) \,.\, x \in c(p, t)$,
- for all $p \in {}^\bullet t$ s.t. $(p, t, p') \in F_{tarc}$ the age of the token x in p satisfies the invariant at p': $\forall p \in {}^\bullet t$ s.t. $(p, t, p') \in F_{tarc} \,.\, \exists x \in M(p) \,.\, x \in c_{tarc}(p, t, p') \wedge x \in \iota(p')$,
- for all $p \in P$ s.t. $(p, t) \in F_{inhib}$ there is no token with age in the interval on the inhibitor arc: $\forall p \in P$ s.t. $(p, t) \in F_{inhib} \,.\, \neg\exists x \in M(p) \,.\, x \in c_{inhib}(p, t)$.

Definition 4.2 (Firing Rule). *If* t *is enabled in a marking* M *then it can be* fired *producing a marking* M' *defined as* $M'(p) = (M(p) \setminus C_t^-(p)) \cup C_t^+(p)$ *for all* $p \in P$ *where*

- *for every* $p \in P$ *such that* $(p,t) \in F$
  $C_t^-(p) = \{x\}$ *where* $x \in M(p)$ *and* $x \in c(p,t)$*,*

- *for every* $p \in P$ *such that* $(t,p) \in F$
  $C_t^+(p) = \{0\}$*, and*

- *for every* $p, p' \in P$ *such that* $(p,t,p') \in F_{tarc}$
  $C_t^-(p) = \{x\} = C_t^+(p')$ *where* $x \in M(p)$*,* $x \in c_{tarc}(p,t,p')$ *and* $x \in \iota(p')$*, and*

- *in all other cases we set the above sets to* $\emptyset$*.*

Note that there may be multiple choices for $C_t^-(p)$ *and* $C_t^+(p)$ *and the minus and union operators are interpreted over multisets.*

Definition 4.3 (Time Delay). *A* time delay $d \in \mathbb{R}_{\geq 0}$ *is allowed in a marking* M *if* $(x + d) \in \iota(p)$ *for all* $p \in P$ *and all* $x \in M(p)$*, i.e. by delaying* d *time units no tokens violate the invariants on places. By delaying* d *time units we reach a marking* M' *defined as* $M'(p) = \{x + d \mid x \in M(p)\}$ *for all* $p \in P$*.*

A TAPN $N = (P, T, F, c, F_{tarc}, c_{tarc}, F_{inhib}, c_{inhib}, \iota)$ generates a TTS $T(N) = (\mathcal{M}(N), \longrightarrow, \mathcal{AP}, \mu)$ where states are markings on N, $M \longrightarrow M'$ if by firing some transition t in marking M we reach the marking M', and $M \xrightarrow{d} M'$ if by delaying d time units in marking M we get to marking M'. The set of atomic propositions $\mathcal{AP}$ and the labeling function μ are defined as $\mathcal{AP} \overset{\text{def}}{=} \{(p \bowtie n) \mid p \in P, n \in \mathbb{N}_0 \text{ and } \bowtie \in \{<, \leq, =, \geq, >\}\}$ and $\mu(M) \overset{\text{def}}{=} \{(p \bowtie n) \mid |M(p)| \bowtie n \text{ and } \bowtie \in \{<, \leq, =, \geq, >\}\}$. The idea here is that the proposition $(p \bowtie n)$ is true in a marking M if and only if the number of tokens in the place p satisfies the constraint with respect to n.

4.2 Networks of Timed Automata

We shall now introduce networks of timed automata in the UPPAAL style [3]. UPPAAL timed automata can perform handshake and broadcast communication and manipulate finite data structures. We define only those features that are needed for our translation, namely broadcast communication and integer variables (used only for counting). These features are only a syntactic sugar and the expressive power is identical to the timed automata model by Alur and Dill [1].

Let $C = \{c_1, c_2, \ldots\}$ be a finite set of real-valued *clocks*. A *clock constraint* (or guard) is a boolean expression defined by the abstract syntax: $g_1, g_2 ::= \mathit{true} \mid c \bowtie n \mid g_1 \wedge g_2$ where $c \in C$, $n \in \mathbb{N}_0$ and $\bowtie \in \{\leq, <, ==, >, \geq\}$. For *invariant clock constraints*, we require $\bowtie \in \{\leq, <\}$. The set of all clock constraints and invariant clock constraints over C are denoted by $\mathcal{G}(C)$ and $\mathcal{G}_{\mathrm{Inv}}(C)$, respectively.

A (clock) *valuation* is a function $\nu : C \to \mathbb{R}_{\geq 0}$ that for every clock $c \in C$ returns the value of c. Let $d \in \mathbb{R}_{\geq 0}$. We define the valuation $(\nu + d)$ after delaying d time units by $(\nu + d)(c) = \nu(c) + d$ for every $c \in C$. Let $r \subseteq C$. We define the valuation $\nu[r]$ after the clocks in r are reset by $\nu[r](c) = 0$ if $c \in r$ and $\nu[r](c) = \nu(c)$ if $c \in C \setminus r$. The satisfaction relation $\nu \models g$ (i.e. when a valuation satisfies a guard g) is defined in the natural way.

We will now define the concept of integer variables. Let X be a finite set of *integer variables*. The set VE(X) of arithmetic expressions over X is given by the abstract syntax $\mathit{expr} ::= m \mid x{+}{+} \mid x{-}{-}$ where $m \in \mathbb{Z}$ and $x \in X$. The set VG(X) of *variable guards* is a boolean combination of the predicates $\mathit{expr} \bowtie \mathit{expr}$ where $\mathit{expr} \in VE(X)$ and $\bowtie \in \{<, \leq, ==, \geq, >\}$.

Variable *assignments* are expressions of the form $x := \mathit{expr}$ where $x \in X$ and $\mathit{expr} \in VE(X)$. The set of all variables assignments over X is denoted by VA(X). A set $\mathcal{A} \subseteq VA(X)$ of variable assignments is called *non-conflicting* if for every $x \in X$ whenever $(x := \mathit{expr}_1) \in \mathcal{A}$ and $(x := \mathit{expr}_2) \in \mathcal{A}$ then $\mathit{expr}_1 = \mathit{expr}_2$.

Finally, we will define a *variable valuation* as a total mapping $z : X \longrightarrow \mathbb{Z}$ that for a variable $x \in X$ returns its current value. This mapping is naturally extended to all variable expressions in VE(X). The satisfaction relation $z \models \phi$ is true if the variable guard $\phi \in VG(X)$ evaluates to true under the valuation z. Let $\mathcal{A}$ be a finite non-conflicting set of variable assignments and let z be a variable valuation. We define $z[\mathcal{A}]$ as a variable valuation updated with the assignments from $\mathcal{A}$ by $z[\mathcal{A}](x) = z(\mathit{expr})$ if $(x := \mathit{expr}) \in \mathcal{A}$ and $z[\mathcal{A}](x) = z(x)$ otherwise.

We can now define the notion of a timed automaton.

Definition 4.4 (Timed Automaton)**.** *A* timed automaton *is a tuple* $(L, \ell_0, \mathrm{Act}, C, X, \longrightarrow, I_C, I_X)$ *where* L *is a finite set of* locations *and* $\ell_0 \in L$ *is the* initial location*,* Act *is a finite set of* actions*,* C *is a finite set of* clocks*,* X *is a finite set of* integer variables*,* $\longrightarrow \subseteq L \times \mathcal{G}(C) \times VG(X) \times \mathrm{Act} \times 2^C \times 2^{VA(X)} \times L$ *is a finite set of* edges *s.t. whenever* $(\ell, g, \phi, a, r, \mathcal{A}, \ell') \in \longrightarrow$ *then* $\mathcal{A}$ *is finite and non-conflicting set of variables assignments,* $I_C : L \to \mathcal{G}_{Inv}(C)$ *is a function assigning* clock invariants *to locations, and* $I_X : L \to VG(X)$ *is a function assigning* variable invariants *to locations.*

We write $\ell \xrightarrow{g,\phi,a,r,\mathcal{A}} \ell'$ instead of $(\ell, g, \phi, a, r, \mathcal{A}, \ell') \in \longrightarrow$, where ℓ is a source location, g is a clock guard, ϕ is a variable guard, a is an action, r is a set of clocks to be reset, $\mathcal{A}$ is a finite non-conflicting set of variable assignments and ℓ' is a target location.

We can now define a network (parallel composition) of timed automata communicating via broadcast. Let *Broad* be a finite set of *broadcast channel names* and let τ denote the *internal action* τ performed by a single component. The set of actions is $Act = \{a¡, a¿ \mid a \in \mathit{Broad}\} \cup \{\tau\}$. The intuition is that $a¡$ indicates initiation of broadcasting on a channel a and all automata where the action $a¿$ is enabled must participate in the broadcast communication.

Definition 4.5 (Network of Timed Automata). *Let* $A_1, A_2, \ldots, A_n$ *for some* $n \in \mathbb{N}$ *be timed automata over a fixed set of actions* Act, *clocks* C *and integer variables* X *such that* $A_i = (L_i, Act, C, X, \longrightarrow_i, I^i_C, I^i_X, \ell^i_0)$ *for all* $1 \leq i \leq n$. *A* network of timed automata *(NTA) is a parallel composition* $A_1 \parallel A_2 \parallel \ldots \parallel A_n$.

A *configuration* of an NTA is a tuple $(\ell_1, \ell_2, \ldots, \ell_n, z, \nu)$ where $\ell_i \in L_i$ for all $1 \leq i \leq n$, z is a variable valuation over X and ν is a clock valuation over C such that for every i, $1 \leq i \leq n$, it holds that $z \models I^i_X(\ell_i)$ and $\nu \models I^i_C(\ell_i)$. The set of all configurations of a given NTA A is denoted by *Conf*(A).

We can now define the precise semantics of networks of timed automata as TTSs. Let $A = A_1 \parallel A_2 \parallel \ldots \parallel A_n$, where $A_i = (L_i, Act, C, X, \longrightarrow_i, I^i_C, I^i_X, \ell^i_0)$ be an NTA. The TTS generated by A is $T(A) = (\mathit{Conf}(A), \longrightarrow, \mathcal{AP}, \mu)$ such that the transition relation consists of

- *Ordinary transitions:* $(\ell_1, \ldots, \ell_i, \ldots, \ell_n, z, \nu) \longrightarrow (\ell_1, \ldots, \ell'_i, \ldots, \ell_n, z', \nu')$ if there is an edge $\ell_i \xrightarrow{g,\phi,\tau,r,\mathcal{A}}_i \ell'_i$ in the i'th automaton such that $\nu \models g$, $z \models \phi$, $\nu' = \nu[r]$, $z' = z[\mathcal{A}]$, $\nu' \models I^i_C(\ell'_i) \wedge \bigwedge_{j \neq i} I^j_C(\ell_j)$ and $z' \models I^i_X(\ell'_i) \wedge \bigwedge_{j \neq i} I^j_X(\ell_j)$,
- *Broadcast synchronization transitions:* $(\ell_1, \ldots, \ell_n, z, \nu) \longrightarrow (\ell'_1, \ldots, \ell'_n, z', \nu')$ if there is $a \in \mathit{Broad}$ and
  - there exists an i, $1 \leq i \leq n$, s.t. $\ell_i \xrightarrow{g_i,\phi_i,a¡,r_i,\mathcal{A}_i}_i \ell'_i$ is an edge in the i'th automaton where $\nu \models g_i$ and $z \models \phi_i$,
  - let $\mathcal{J}$ be the set of all j, $1 \leq j \neq i \leq n$, s.t. $\ell_j \xrightarrow{g_j,\phi_j,a¿,r_j,\mathcal{A}_j}_j \ell'_j$ in the j'th automaton where $\nu \models g_j$ and $z \models \phi_j$,
  - for all $j \in \mathcal{J}$ we set ℓ'_j, $\mathcal{A}_j$ and r_j according to the edge $\ell_j \xrightarrow{g_j,\phi_j,a¿,r_j,\mathcal{A}_j}_j \ell'_j$ (note that there may be multiple edges to choose from),

– for all $j \notin \mathcal{J}$, $1 \le j \ne i \le n$, we let $\ell'_j = \ell_j$, $\mathcal{A}_j = \emptyset$ and $r_j = \emptyset$,

– $z' = (\ldots((\ldots(((z[\mathcal{A}_i])[\mathcal{A}_1])[\mathcal{A}_2])\ldots[\mathcal{A}_{i-1}])[\mathcal{A}_{i+1}])\ldots[\mathcal{A}_{n-1}])[\mathcal{A}_n]$ such that $z' \models \bigwedge_{k=1}^{n} I_X^k(\ell'_k)$,

– $\nu' = \nu[R]$ where $R = \bigcup_{k=1}^{n} r_k$ such that $\nu' \models \bigwedge_{k=1}^{n} I_C^k(\ell'_k)$, and

- *Delay transitions:* $(\ell_1, \ldots, \ell_n, z, \nu) \xrightarrow{d} (\ell_1, \ldots, \ell_n, z, \nu + d)$ if $d \in \mathbb{R}_{\ge 0}$ s.t. $\nu + d \models \bigwedge_{i=1}^{n} I_i(\ell_i)$.

We let $\mathcal{AP} \overset{\text{def}}{=} \{(\#\ell \bowtie m) \mid \ell \in \cup_{i=1}^{n} L_i, m \in \mathbb{N}_0 \text{ and } \bowtie \in \{<, \le, =, \ge, >\}\}$, and $\mu : \mathit{Conf}(A) \longrightarrow 2^{\mathcal{AP}}$ is defined such that a proposition $(\#\ell \bowtie m)$ is true in a given configuration if and only if the number of parallel components that are currently in the location ℓ satisfies the constraint with respect to m.

The initial configuration is $(\ell_0^1, \ell_0^2, \ldots, \ell_0^n, z_0, \nu_0)$ where $\nu_0(c) = 0$ for all $c \in C$. We require that z_0 satisfies the variable invariants of all initial locations.

Remark 4.6. *Note that during a broadcast, the assignments on the edge of the sender are evaluated first, followed by the assignments of the receivers that are evaluated in the order from* A_1 *to* A_n.

4.3 The Translation

We will now present the translation from k-bounded TAPN (where the maximum number of tokens in every reachable marking is at most k) to NTA. For each token in the net, we create a parallel component in the network of timed automata. Since we cannot dynamically instantiate new timed automata, we need to have a constant number of tokens in the net at all times. As we assume that the net is k-bounded, it is enough to construct k automata to simulate each token. In each of these automaton there is a location corresponding to each place in the net. Whenever a TA is in one of these locations, it simulates a token in the corresponding place. Moreover, each automaton has a local clock which represents the age of the token. All automata simulating the tokens have the same structure, the only difference being their initial location, which corresponds to the tokens' initial position in the net. Because there may not always be exactly k tokens present during the execution of the net, we add a new location $\ell_{capacity}$ where the automata representing currently unused tokens are waiting.

In addition to these "token" automata we create a single control automaton. The purpose is to simulate the firing of transitions and to move tokens around via broadcasts

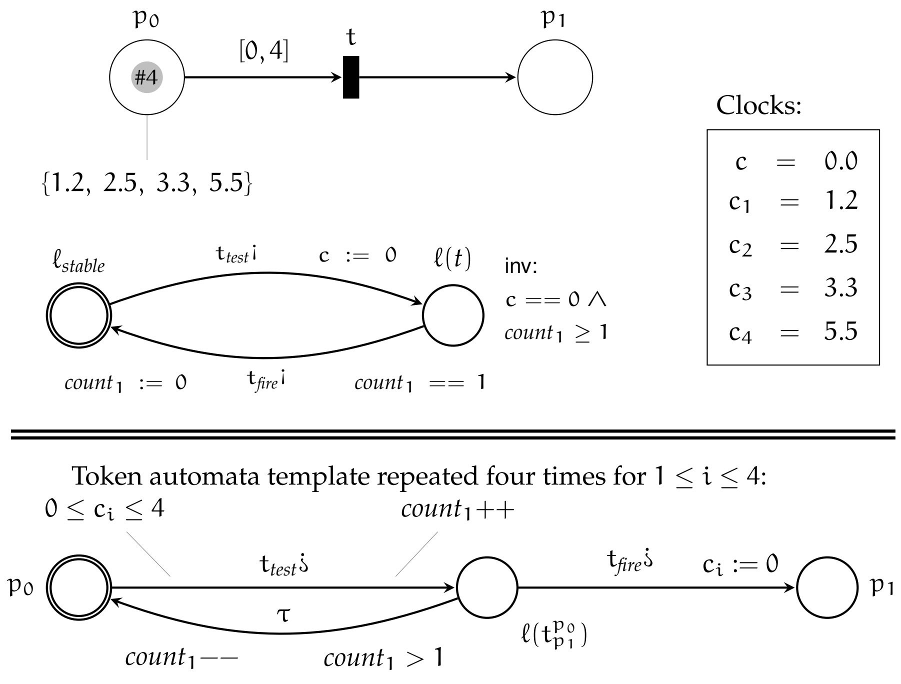

Figure 6: A simple TAPN and the translated NTA.

initiated by the control automaton. This automaton has a location ℓ_{stable} which acts as a mutex in the sense that the control automaton moves out of this location once the simulation of a transition begins and returns back once the simulation of the transition ends. Moreover, each time the automaton is in ℓ_{stable}, the token automata in the composed NTA correspond to a marking in the TAPN. We will first show how the translation works on two examples.

Example 4.7. *Figure 6 shows a simple TAPN with a single transition and four tokens of different ages. The translated NTA consists of five automata, one control automaton (topmost automaton) and four token automata, one for each token. Notice that in this example we have refrained from drawing the $\ell_{capacity}$ location as it is not used.*

The translated NTA works as follows. First, the control automaton broadcasts on the channel t_{test}. Any token automaton with its clock in the interval $[0,4]$ is forced to participate in the broadcast; in our case three token automata will participate. We use integer variables to count the number of token automata that took part in the broadcast. Because the preset of t has size one, we only need one counter variable $count_1$. Once the token automata synchronized in the broadcast, they move to the intermediate locations $\ell(t^{p_0}_{p_1})$ and during the update each increments $count_1$ by one; in our case the value of $count_1$ will become three. This means that the invariant on $\ell(t)$ in the control automaton is satisfied. In other words, we know that there are enough tokens with appropriate ages in the input places for t to fire. Notice that if there were not enough

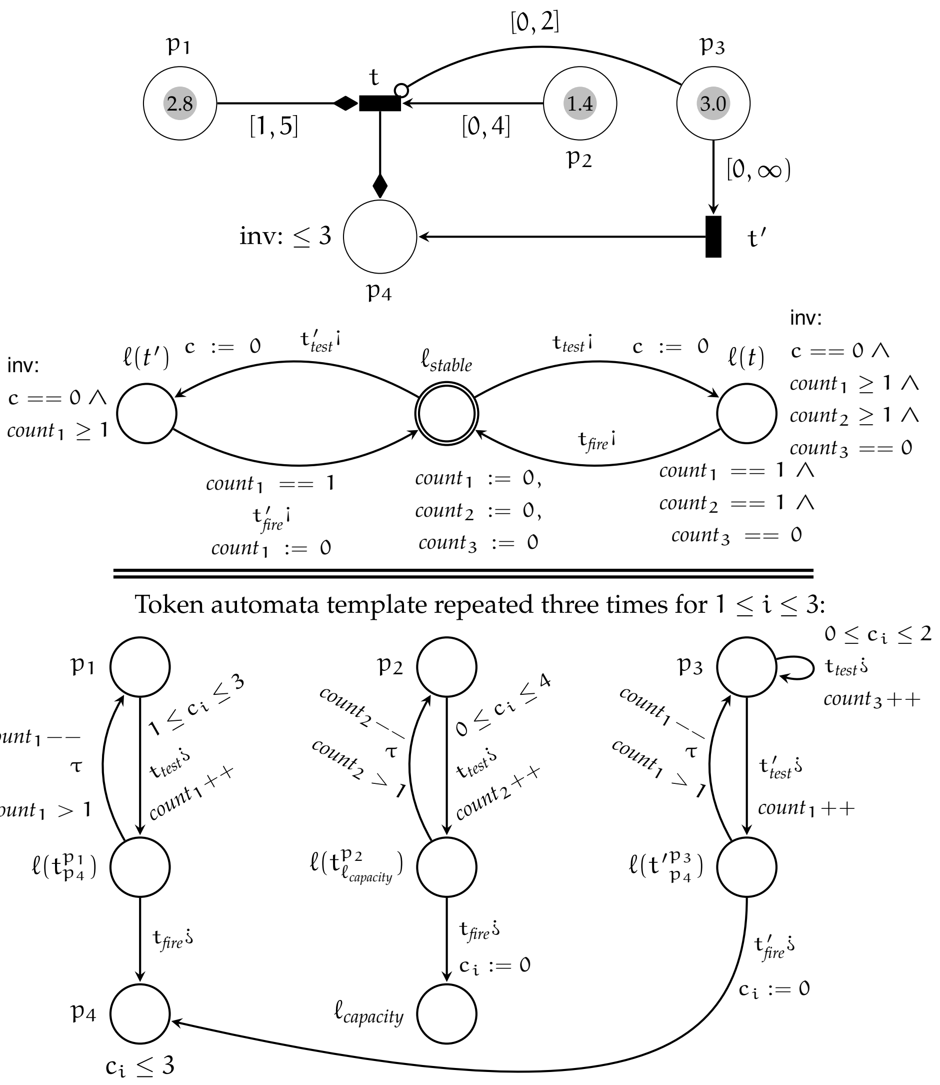

Figure 7: A TAPN and the translated NTA.

tokens in some of the input places, then the invariant on $\ell(t)$ was not satisfied and the broadcast could not take place at all. This is one of the crucial aspects to realize in order to see why this translation preserves liveness properties.

Now the value of $count_1$ is three and the control automaton may not broadcast on the t_{fire} channel yet since the guard ensures that this is only possible when exactly one token automaton remains in its intermediate place. Therefore, we are forced to move two of the token automata back to p_0 via the τ-transitions. This is possible only as long as $count_1$ is strictly greater than one. Hence exactly one token automaton has to remain in its intermediate place before the control automaton can broadcast on the t_{fire} channel and finalize the simulation of firing t. Note that due to the invariant $c == 0$ in the control automaton, no time delay is possible during the simulation of the transition.

After demonstrating the basic idea of the broadcast translation, let us discuss a slightly more elaborate example using all of the features of the TAPN model.

Example 4.8. *Consider the TAPN model in Figure 7 that uses transport arcs (the pair of arcs with filled tips from* p_1 *to* p_4*) for moving tokens while preserving their ages, an inhibitor arc (the arc with the circle tip) and an invariant in place* p_4*. The NTA created by our algorithm is below the net. As before, the template is repeated three times, once for each token, the only difference being the initial location (*p_1, p_2 *and* p_3*, respectively) and the name of the clock (*c_1, c_2 *and* c_3*, respectively).*

We see that the control automaton has a test-fire loop for every transition in the TAPN model. There are some special constructions worth mentioning. First of all, consider the inhibitor arc from p_3 *to* t*. This arc is encoded using a self-loop participating in the* t_{test} *broadcast transition. We use a counter variable to count the number of automata that take this edge. We simply encode the requirement that there is no token in the interval* $[0,2]$ *by adding the invariant* $count_3 == 0$ *on the location* $\ell(t)$*.*

A second observation is the guard on the edge from p_1 *to* $\ell(t^{p_1}_{p_4})$*. It is evident that this does not match the interval* $[1,5]$ *located on the arc from* p_1 *to* t *in the TAPN model. The guard* $1 \leq c_i \leq 3$ *is in fact the intersection of the interval* $[1,5]$ *and the invariant* ≤ 3 *on the place* p_4*. This is because the age of the token consumed in* p_1 *will be preserved once moved to* p_4 *and by intersecting the intervals we avoid possible deadlocks. One may think that it is enough to add the invariant* ≤ 3 *on the intermediate place, however, this may result in incorrect behavior. If there were two tokens in* p_1 *with ages 4 and 2, the broadcast on* t_{test} *would be blocked. This is because invariants block the entire broadcast transition even if only a single automaton with a satisfied guard cannot participate due to the violation of the invariant in its target location.*

For our specific example, we need at least one token of age $[1,3]$ *in* p_1*, at least one token of age* $[0,4]$ *in* p_2 *and zero tokens of age* $[0,2]$ *in* p_3 *in order for* t *to be enabled, which is precisely encoded in the invariant on* $\ell(t)$*. The reader may also notice that different transitions share counter variables. The variable* $count_1$ *is used in the simulation of both* t *and* t' *but they are used in a non-conflicting way, in the sense that we are never simulating* t *and* t' *at the same time. We also see that during the simulation of* t' *we do not take the invariant of the target location into account since the arc from* t' *to* p_4 *is a normal arc and produces a token of age zero which always satisfies any invariant.*

We shall now proceed to present the translation algorithm. For every transition t we assume an a priori fixed set *Pairing*(t), motivated by [9], where

Algorithm 1: Translation from k-bounded TAPN to NTA.

```
Input: A k-bounded TAPN $N = (P, T, F, c, F_{tarc}, c_{tarc}, F_{inhib}, c_{inhib}, \iota)$ with a marking $M_0$
Output: NTA $P_{TA} = A \| A_1 \| A_2 \| \dots \| A_k$ s.t. $A = (L, \mathcal{A}ct, C, X, \longrightarrow, I_C, I_X, \ell_0)$ and
        $A_i = (L_i, \mathcal{A}ct, C, X, \longrightarrow_i, I^i_C, I^i_X, \ell^i_0)$
begin
    for $i := 1$ to $k$ do $L_i := P \cup \{\ell_{capacity}\}$
    $L := \{\ell_{stable}\}$; $\mathcal{A}ct := \{t_{test}¡, t_{test}¿, t_{fire}¡, t_{fire}¿ \mid t \in T\} \cup \{\tau\}$
    $C := \{c, c_1, c_2, \dots, c_k\}$; $X := \{count_i \mid 1 \leq i \leq NumVars(N)\}$
    forall $t \in T$ do
        $j := 0$; $varInv_t := \mathtt{true}$; $varGuard_t := \mathtt{true}$
        while $|Pairing(t)| > 0$ do
            $j := j + 1$; Remove some $(p, I, p', type)$ from $Pairing(t)$
            for $i := 1$ to $k$ do
                $L_i := L_i \cup \{\ell(t^p_{p'})\}$
                Add $p \xrightarrow{g,\, true,\, t_{test}¿,\, \emptyset,\, count_j++}_i \ell(t^p_{p'})$ s.t. $g := c \in I$ if $type = normal$ else $g := c \in I \cap \iota(p')$
                Add $\ell(t^p_{p'}) \xrightarrow{true,\, true,\, t_{fire}¿,\, R,\, \emptyset}_i p'$ s.t. $R = \{c_i\}$ if $type = normal$ else $R = \emptyset$
                Add $\ell(t^p_{p'}) \xrightarrow{true,\, count_j > 1,\, \tau,\, \emptyset,\, count_j--}_i p$
            $varInv_t := varInv_t \wedge count_j \geq 1$; $varGuard_t := varGuard_t \wedge count_j == 1$
        forall $p \in P$ where $(p, t) \in F_{inhib}$ do
            $j := j + 1$; for $i := 1$ to $k$ do Add $p \xrightarrow{c_i \in c_{inhib}(p,t),\, true,\, t_{test}¿,\, \emptyset,\, count_j++}_i p$
            $varInv_t := varInv_t \wedge count_j == 0$; $varGuard_t := varGuard_t \wedge count_j == 0$
        $L := L \cup \{\ell(t)\}$; Add $\ell_{stable} \xrightarrow{true,\, true,\, t_{test}¡,\, \{c\},\, \emptyset} \ell(t)$ and $\ell(t) \xrightarrow{true,\, varGuard_t,\, t_{fire}¡,\, \emptyset,\, \{count_i := 0 \mid 1 \leq i \leq j\}} \ell_{stable}$
    for $i := 1$ to $k$ do
        $I^i_C(p) := \begin{cases} c_i \leq a & \text{if } p \in P \text{ and } \iota(p) = [0, a] \\ c_i < b & \text{if } p \in P \text{ and } \iota(p) = [0, b) \\ \mathtt{true} & \text{if } p \in L_i \setminus P \end{cases}$    $I^i_X(p) := true$ for $p \in L_i$
    $I_C(p) := \begin{cases} \mathtt{true} & \text{if } p = \ell_{stable} \\ c \leq 0 & \text{if } p \in L \setminus \{\ell_{stable}\} \end{cases}$    $I_X(p) := \begin{cases} varInv_t & \text{if } p = \ell(t) \text{ for } t \in T \\ true & \text{if } L \setminus \{\ell(t) \mid t \in T\} \end{cases}$
    $i := 0$; forall $p \in P$ do forall $Token \in M_0(p)$ do $\ell^i_0 := p$; $i := i + 1$
    for $i := |M_0| + 1$ to $k$ do $\ell^i_0 := \ell_{capacity}$
    $\ell_0 := \ell_{stable}$
end
```

$$
\begin{aligned}
Pairing(t) = \{&(p, I, p', tarc) \mid (p, t, p') \in F_{tarc} \wedge I = c_{tarc}(p, t, p')\} \cup \\
&\{(p_1, I_1, p'_1, normal), \dots, (p_m, I_m, p'_m, normal) \mid \\
&\;\{p_1, \dots, p_\ell\} = \{p \mid (p, t) \in F\}, \{p'_1, \dots, p'_{\ell'}\} = \{p \mid (t, p) \in F\}, \\
&\;m = \max(\ell, \ell'), I_i = c(p_i, t) \text{ if } 1 \leq i \leq \ell \text{ else } I_i = [0, \infty), \\
&\;p_i = \ell_{capacity} \text{ if } \ell < i \leq m, p'_i = \ell_{capacity} \text{ if } \ell' < i \leq m\}.
\end{aligned}
$$

The set *Pairing*(t) simply pairs input and output places of t in order to fix the paths on which tokens will travel when firing t. It also records the time interval on the input arc and the type of the arc (*normal* for normal arcs and *tarc* for transport arcs). As an example, a possible pairing for the transition t in Figure 7 is $\mathit{Pairing}(t) = \{(p_1, [1,5], p_4, \mathit{tarc}), (p_2, [0,4], \ell_{\mathit{capacity}}, \mathit{normal})\}$. We also let $\mathit{NumVars}(N) \overset{\text{def}}{=} \max_{t \in T}(|\mathit{Pairing}(t)| + |\{(p,t) \mid (p,t) \in F_{\mathit{inhib}}\}|)$ denote the maximum number of integer variables needed in the translation. The translation is given in Algorithm 1. Note that it works in polynomial time.

4.4 Translation Correctness

For notational convenience, we shall in this section sometimes label discrete transitions with the names of the transitions by which they were performed.To prove the correctness of this translation, we will follow the methodology described in Section 3.2. We let (N, M_0) be a marked k-bounded TAPN and let P_{TA} be the NTA constructed by Algorithm 1 with initial configuration s_0.

We define the *stable* proposition as $(\#\ell_{\mathit{stable}} = 1)$. Recall that $(\#\ell_{\mathit{stable}} = 1)$ is true whenever there is one TA in the ℓ_{stable} location. Thus $(\ell, \ell_1, \ell_2, \dots, \ell_k, z, \nu) \models (\#\ell_{\mathit{stable}} = 1)$ if and only if $\ell = \ell_{\mathit{stable}}$.

We will first show that P_{TA} possesses the three properties required by a complete one-by-many correspondence. Recall that $T(P_{TA})$ is the TTS generated by P_{TA}.

Lemma 4.9. $T(P_{TA})$ *is a* delay-implies-stable *TTS.*

Proof. Since all locations in the control automaton except ℓ_{stable} have the invariant $c \leq 0$, it follows that only 0 delays are possible in intermediate states. □

Lemma 4.10. $T(P_{TA})$ *is a* delay-preserves-stable *TTS.*

Proof. Since time delays do not change the current location of any automaton, it follows that any time delay from a stable configuration will result in another stable configuration. Thus, $T(P_{TA})$ is a *delay-preserves-stable* TTS.

□

Lemma 4.11. $T(P_{TA})$ *is an* eventually-stable *TTS.*

Proof. We must show that for any (finite or infinite) maximal discrete sequence of length at least 1

$$\rho = s_0 \longrightarrow s_1 \longrightarrow s_2 \longrightarrow s_3 \longrightarrow \cdots$$

where $s_0 \models (\#\ell_{\mathit{stable}} = 1)$, there exists an $i \geq 1$ such that $s_i \models (\#\ell_{\mathit{stable}} = 1)$.

Because s_0 is stable, it follows that the control automaton is in location ℓ_{stable} in s_0. We will show that by construction any maximal discrete sequence starting in s_0 contains a prefix of the form

$$s_0 \xrightarrow{t_{test}} s_1 \longrightarrow s_2 \longrightarrow \cdots \longrightarrow s_{n-1} \xrightarrow{t_{fire}} s_n$$

such that $n \leq k + 1$ (we assume that the input net is k-bounded) and $s_n \models (\#\ell_{stable} = 1)$.

If any discrete transition is enabled in s_0 then by construction, it is a broadcast transition on some channel t_{test}. Assume that $s_0 \xrightarrow{t_{test}} s_1$. In s_1 there are only two possibilities for discrete transitions, which are mutually exclusive and by construction one of them is always possible (due to the use of integer guards and invariants). Either there is a τ-transition enabled in some token TA in case there are more than one token TA currently in the same intermediate location $\ell(t^p_{p'})$ or a broadcast on the t_{fire} channel is enabled when there is exactly one token TA in each of the required $\ell(t^p_{p'})$ locations.

In the first case, the only possibility is to keep taking τ-transitions, which move the extra token TA back to their original locations. This must continue until some configuration s_{n-1} is reached where there are no more τ-transitions enabled. By construction, this will eventually happen. It follows that t_{fire} will be the only enabled discrete transition in s_{n-1}. In the worst case, all k TA participated in the t_{test} broadcast synchronization and only a single TA is required to synchronize on the t_{fire} broadcast channel. Thus, we must move $k - 1$ TA back to their original locations.

In the second case, only the t_{fire} broadcast synchronization is enabled and we have by construction of P_{TA} that the invariants on the target location of all t_{fire} broadcast receivers will be satisfied. Since synchronizing on the t_{fire} channel brings the control automaton back to ℓ_{stable}, it follows that $s_n \models (\#\ell_{stable} = 1)$. Hence, $T(P_{TA})$ is an *eventually-stable* TTS. $\square$

We now define the proposition translation function tr_p. A TAPN proposition $(p \bowtie n)$ is translated into $(\#p \bowtie n)$.

Let us now define a correspondence relation $\mathcal{R}$ between markings and configurations. Let $M = \{(p_1, r_1), (p_2, r_2), \ldots, (p_n, r_n)\}$ be a marking of N such that $n \leq k$, where (p_i, r_i) is a token located in the place p_i with age $r_i \in \mathbb{R}_{\geq 0}$. Further, let $s = (\ell, \ell_1, \ell_2, \ldots, \ell_k, z, v)$ be a configuration of P_{TA}. We define $\mathcal{R}$ such that $(M, s) \in \mathcal{R}$ iff there exists an injection $h : \{1, 2, \ldots, n\} \longrightarrow \{1, 2, \ldots, k\}$ such that $\ell = \ell_{stable}$, $\ell_{h(i)} = p_i$ and $v(c_{h(i)}) = r_i$ for all i where $1 \leq i \leq n$, $\ell_j = \ell_{capacity}$ for all $j \in \{1, 2, \ldots, k\} \setminus \mathit{range}(h)$ and $\mathit{count}_i = 0$ for all $1 \leq i \leq \mathit{NumVars}(N)$. Intuitively, if $(M, s) \in \mathcal{R}$ then for every token in M there is a TA where its location and clock valuation matches the token data and vice versa.

We shall now prove that $\mathcal{R}$ is a complete one-by-many correspondence.

Theorem 4.12. *The relation $\mathcal{R}$ is a complete one-by-many correspondence.*

Proof. Let (N, M_0) be a marked k-bounded TAPN and P_{TA} be the NTA generated by Algorithm 1. We will show that $\mathcal{R}$ satisfies all requirements of Definition 3.2 and thus is a complete one-by-many correspondence. From Lemma 4.9, Lemma 4.10 and Lemma 4.11 it follows that $T(P_{TA})$ is a *delay-implies-stable*, *delay-preserves-stable* and *eventually-stable* TTS.

Let M be a marking and let s be a configuration, such that $M \mathcal{R} s$. We shall now prove that $\mathcal{R}$ satisfies conditions 1-6 of Definition 3.2 on page 10.

1. $s \models (\#\ell_{stable} = 1)$ follows from the definition of $\mathcal{R}$.

2. From the definition of $\mathcal{R}$ it follows that $M \models \wp$ iff $s \models tr_p(\wp)$.

3. Assume $M \longrightarrow M'$. This means that there exists a transition t such that $M \xrightarrow{t} M'$. We must show that $s \rightsquigarrow s'$ such that $M' \mathcal{R} s'$. We will start by showing that t_{test} is enabled in s. Since t_{test} is a broadcast channel, the first requirement is that the t_{test} sender has to be enabled. There are no guards on the sender transition, so only the invariant on $\ell(t)$ may block the t_{test} sender. The second requirement is that every possible receiver (i.e. any receiver whose guard is satisfied) must participate in the broadcast synchronization. Due to the construction of P_{TA}, there are no invariants on the intermediate places $\ell(t^p_{p'})$. Thus, any receiver with a satisfied guard can participate in the broadcast synchronization. By the fact that $M \mathcal{R} s$, it follows that enough TA will participate in the broadcast synchronization to satisfy the invariant on $\ell(t)$. Thus, $s \xrightarrow{t_{test}}$.

   Since $s \models (\#\ell_{stable} = 1)$ and $s \xrightarrow{t_{test}}$ we have by Lemma 4.11 that

   $$s \xrightarrow{t_{test}} s_1 \longrightarrow s_2 \longrightarrow \cdots \longrightarrow s_n = s'$$

   such that $s_i \not\models (\#\ell_{stable} = 1)$ for $1 \leq i < n$ and $s_n \models (\#\ell_{stable} = 1)$. By construction, this sequence has the form

   $$s \xrightarrow{t_{test}} s_1 \longrightarrow s_2 \longrightarrow \cdots \longrightarrow s_{n-1} \xrightarrow{t_{fire}} s_n$$

where $n \leq k+1$. Thus, since $M \xrightarrow{t} M'$ we can use the sequence above and Lemma 4.9 to get the sequence $s \rightsquigarrow s'$. Observe that if $|{}^\bullet t| > |t^\bullet|$ then $|{}^\bullet t| - |t^\bullet|$ token TA will be moved to $\ell_{capacity}$ after simulating t from s and if $|{}^\bullet t| < |t^\bullet|$ then $|t^\bullet| - |{}^\bullet t|$ token TA will be moved out of $\ell_{capacity}$ after simulating t from s. Since $M \mathcal{R} s$, we get $M' \mathcal{R} s'$ by matching the changes in M when firing t to the changes in s when executing the sequence above.

4.,6. Time delays can only be restricted by invariants. By Lemma 4.9 and the fact that the invariant on place p in N is carried over to location p in P_{TA}, we have that it is always possible to do the same time delays in M and s. By Lemma 4.10, it follows clearly that if $M \xrightarrow{d} M'$ then $s \xrightarrow{d} s'$ and $M' \mathcal{R} s'$ and vice versa.

5. Assume $s \rightsquigarrow s'$. This implies that

$$s \longrightarrow s_1 \xrightarrow{0} s_1 \longrightarrow s_2 \xrightarrow{0} s_2 \longrightarrow \cdots \longrightarrow s_{n-1} \xrightarrow{0} s_{n-1} \longrightarrow s_n = s'$$

such that $s_i \not\models (\#\ell_{stable} = 1)$ for $1 \leq i < n$ and $s_n \models (\#\ell_{stable} = 1)$. By construction of P_{TA}, we have that $n \leq k+1$ and that this sequence must be of the form (leaving out 0 delays)

$$s \xrightarrow{t_{test}} s_1 \longrightarrow s_2 \longrightarrow \cdots \longrightarrow s_{n-1} \xrightarrow{t_{fire}} s_n.$$

Since $s \xrightarrow{t_{test}} s_1$ we know that the invariant on $\ell(t)$ is satisfiable. By the construction of P_{TA}, this in turn means that there are enough token automata that can synchronize on t_{test} in such a way that for each input place $p \in {}^\bullet t$ there is at least one automaton whose current location is p and the clocks of these automata satisfy the guards on the t_{test} transitions. Further, for each place p' such that there is an inhibitor arc from p' to t, there is no automaton in location p' with a clock satisfying the guard on t_{test}. This is exactly what is needed to fire t from M, and because $M \mathcal{R} s$, then $M \xrightarrow{t} M'$.

If any token TA moves out of $\ell_{capacity}$ when $s \rightsquigarrow s'$ then additional tokens will be produced when firing transition t from M and if any token TA moves into $\ell_{capacity}$ when $s \rightsquigarrow s'$ then t will consume more tokens than it produces when fired from M. Since $M \mathcal{R} s$, we clearly get $M' \mathcal{R} s'$.

□

5 Conclusion

We have introduced a general framework for arguing when a translation between two timed transition systems preserves TCTL model checking. The framework generalizes earlier translations like [9] and [10] that dealt with concrete models. Apart from [9, 10], the framework is applicable also to other translations like [8, 11, 13, 17]. We have further described a novel reduction from bounded timed-arc Petri nets with transport/inhibitor arcs and invariants on places to networks of timed automata in the UPPAAL style to which the framework is applicable. Compared to earlier translations, we considered a more general class of nets and showed that also liveness TCTL properties are preserved. The translation works in polynomial time and it has been implemented in the verification tool TAPAAL [9].

Acknowledgements. We would like to thank Alexandre David and Kenneth Y. Jørgensen for their comments and help with the implementation issues.

References

[1] R. Alur and D. L. Dill. A Theory of Timed Automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[2] M. Archer, L. HongPing, N. Lynch, S. Mitra, and S. Umeno. Specifying and proving properties of timed I/O automata in the TIOA toolkit. In *Proc. of MEMOCODE'06*, pages 129 –138, 2006.

[3] G. Behrmann, A. David, and K. G. Larsen. A tutorial on UPPAAL. In *Proc. of SFM-RT'04*, volume 3185 of *LNCS*, pages 200–236. Springer, 2004.

[4] B. Bérard, F. Cassez, S. Haddad, D. Lime, and O.H. Roux. Comparison of the expressiveness of timed automata and time Petri nets. In *Proc. of FORMATS'05*, volume 3829 of *LNCS*, pages 211–225. Springer, 2005.

[5] B. Berthomieu, F. Peres, and F. Vernadat. Bridging the gap between timed automata and bounded time Petri nets. In *Proc. of FORMATS'06*, volume 4202 of *LNCS*, pages 82–97. Springer, 2006.

[6] T. Bolognesi, F. Lucidi, and S. Trigila. From timed Petri nets to timed LOTOS. In *Proc. of PSTV'90*, pages 395–408, 1990.

[7] H. Boucheneb, G. Gardey, and O.H. Roux. TCTL model checking of time Petri nets. *Journal of Logic and Computation*, 19(6):1509–1540, 2009.

[8] P. Bouyer, S. Haddad, and P.A. Reynier. Timed Petri nets and timed automata: On the discriminating power of Zeno sequences. *Information and Computation*, 206(1): 73–107, 2008.

[9] J. Byg, K.Y. Joergensen, and J. Srba. An efficient translation of timed-arc Petri nets to networks of timed automata. In *Proc. of ICFEM'09*, volume 5885 of *LNCS*, pages 698–716. Springer, 2009.

[10] F. Cassez and O.H. Roux. Structural translation from time Petri nets to timed automata. *ENTCS*, 128(6):145 – 160, 2005. Proc. of AVoCS'04.

[11] J.S. Dong, P. Hao, S. Qin, J. Sun, and W. Yi. Timed Automata Patterns. *IEEE Transactions on Software Engingeering*, 34(6):844–859, 2008.

[12] G. Gardey, D. Lime, M. Magnin, and O.H. Roux. Romeo: A tool for analyzing time Petri nets. In *Proc. of CAV'05*, volume 3576 of *LNCS*, pages 418–423. Springer, 2005.

[13] A. Janowska, P. Janowski, and D. Wróblewski. Translation of Intermediate Language to Timed Automata with Discrete Data. *Fundamenta Informaticae*, 85(1-4): 235–248, 2008.

[14] P. M. Merlin. *A Study of the Recoverability of Computing Systems*. PhD thesis, University of California, Irvine, 1974.

[15] W. Penczek and A. Pólrola. *Advances in Verification of Time Petri Nets and Timed Automata: A Temporal Logic Approach*, volume 20 of *Studies in Computational Intelligence*. Springer, 2006.

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

[17] J. Srba. Timed-arc Petri nets vs. networks of timed automata. In *Proc. of ICATPN'05*, volume 3536 of *LNCS*, pages 385–402. Springer, 2005.

[18] J. Srba. Comparing the expressiveness of timed automata and timed extensions of Petri nets. In *Proc. of FORMATS'08*, volume 5215 of *LNCS*, pages 15–32. Springer, 2008.