CSL^{TA}: an Expressive Logic for Continuous-Time Markov Chains *

Susanna Donatelli
Dipartimento di Informatica
Università di Torino
Italy
susi@di.unito.it

Serge Haddad
LAMSADE, CNRS &
Université Paris-Dauphine
France
haddad@lamsade.dauphine.fr

Jeremy Sproston
Dipartimento di Informatica
Università di Torino
Italy
sproston@di.unito.it

Abstract

The stochastic temporal logic CSL can be used to describe formally properties of continuous-time Markov chains, and has been extended with expressions over states and actions to obtain the logic asCSL. However, properties referring to the probability of a finite sequence of timed events (such as "with probability at least 0.75, the system will be in state set A at time 5, then in state set B at time 7, then in state set C at time 20") cannot be expressed in either CSL or asCSL. With the aim of increasing the expressive power of temporal logics for continuous-time Markov chains, we introduce the logic CSL^{TA} and its model-checking algorithm. CSL^{TA} extends CSL and asCSL by allowing the specification of timed properties through a deterministic one-clock timed automata.

1 Introduction

The complexity of hardware and software systems has led to interest in automatic methods for increasing our confidence that the functional and performance requirements of such systems are satisfied. Stochastic systems can be modelled as continuous-time Markov chains (CTMCs), or in some higher-level language, such as stochastic Petri nets (SPNs) or stochastic process algebra. Performance and dependability requirements are then defined using state and/or transition rewards or in terms of stochastic extensions of temporal logic, such as CSL [3, 5] or asCSL [4]. Temporal logic based approaches are particularly useful when the measure of interest depends on the execution path. Given a formal description of the system and its requirements, we can then execute a model-checking algorithm which establishes automatically whether the system model meets the requirements expressed in CSL or asCSL.

In this paper, we propose a new stochastic temporal logic which builds on CSL and asCSL, but which enriches the set of properties that can be defined and verified, and present its associated model-checking algorithm. Let us first explain the main motivations for introducing a new logic. Consider a system whose stochastic behavior is described by a CTMC whose states are partitioned into "system is working properly" (*work*-states), "system is working in degraded mode" (*degr*-states), or "system is not working properly" (*fail*-states). The CTMC can move from *work* to *degr* states and to *fail* states (either directly or through *degr* states). A classical dependability property requires the computation of the probability of failing within the time interval I: these probabilities can be easily computed using classical solution methods for CTMCs.

Instead, if we are interested in only those failures in which the system fails within the time interval I, without first entering the degraded mode, we have to compute the probability of reaching a *fail* state within I, while passing only through *work* states. The stochastic temporal logic CSL has temporal operators that allow a simple and semantically-clear description of such a property using the Until operator: $\mathcal{P}_{\leq\lambda}(work\ \mathcal{U}^I fail)$.

The logic asCSL permits the specification of paths in terms of state and *action labels*. For example, $\mathcal{P}_{\leq\lambda}((work, Act)^*; (work, \mathtt{tr}); (fail, \surd)^I)$, is similar to the CSL formula above, with the additional restriction that the change from $work$-states to $fail$-states is due to action $\mathtt{tr}$.

However, properties in which paths are specified by an arbitrary number of time constraints, or by time constraints which can be dependent on the behavior of the CTMC, cannot be expressed in the model-checking analyses which we are aware of. Consider the case in which we are interested in the probability of the system exhibiting the following behavior: the system goes from *work* states to *degr* states within I, and then from *degr* states to *fail* states within I'. With this property, we are therefore characterizing paths not only in terms of states, but also in terms of two time constraints. Furthermore, we can also consider the case in which the time interval I' is not relative to the start of the execution of the CTMC, but relative to the transition from

*Supported in part by ANR-06-SETI-002 project Checkbound and EEC project Crutial.

work states to *degr* states. A similar extension can be considered for paths defined also in terms of actions.

The proposal of this paper is to describe paths of interest using a *timed automaton* [2] with a *single* clock: the paths of the CTMC on which the probability is computed will be those finite paths that "match" the specification of the timed automaton. The timed automaton has state and action labels, and *deterministic* behaviour. The resulting logic is called CSL$^{\text{TA}}$. We also present a model-checking algorithm for CSL$^{\text{TA}}$. Contrary to the previous approaches, which perform *ad hoc* transformations of the CTMC before a transient or steady-state analysis, this algorithm generates a Markov regenerative process and then computes a reachability probability on this process. Furthermore, we prove that CSL$^{\text{TA}}$ subsumes both CSL and asCSL. Finally, we prove that CSL$^{\text{TA}}$ is strictly more expressive than CSL: this proof is completely different from the ones used for similar results in non-stochastic models.

With regard to related work, performance metrics that depend on paths have also been studied in [8, 14] In particular, the work in [14] uses automata for the specification of the set of paths of interest of a CTMC: rewards, which are usually associated with states or transitions of the CTMC, are instead associated with locations and transitions of the automaton, thus providing a wide range of performance measures based on states and/or events of the CTMC. We also note that the logic CSL$^{\text{TA}}$ is similar to the logic $\text{TECTL}^*_\exists$ [7] from the non-probabilistic model-checking literature. One-clock timed automata have been studied in, for example, [13, 15]. Finally, we recall that the original definition of CSL permitted the description of a sequence of timed Until formulae within a single probabilistic operator $\mathcal{P}_{\sim\lambda}$ [3]: however, only decidability issues are considered, while model-checking algorithms for CSL [5] do not permit sequences of timed Until formulae.

The rest of the paper is organized as follows: Section 2 defines the syntax and semantics of CSL$^{\text{TA}}$, illustrated with the help of a number of small examples. Section 3 presents the model-checking algorithm for CSL$^{\text{TA}}$ and gives an example on a simple CTMC, while Section 4 shows that CSL$^{\text{TA}}$ is strictly more expressive than CSL and asCSL. Section 5 summarizes the paper and reports on future work.

2 Syntax and Semantics of CSL$^{\text{TA}}$

2.1 Markov Chains and Timed Automata

We first introduce continuous-time Markov chains labelled both by atomic propositions on states and by actions on transitions. Let $\mathbb{R}_{\geq 0}$ ($\mathbb{R}_{>0}$) be the set of non-negative (positive) reals, and let $\mathbb{N}$ be the set of natural numbers.

Definition 2.1 (Action- and state-labelled Markov chain) *An* action- and state-labelled continuous-time Markov chain *(ASMC) is a tuple* $\mathcal{M} = \langle S, Act, AP, lab, \mathbf{R}\rangle$*, where* S *is a finite set of states,* Act *is a finite set of action labels,* AP *is a finite set of atomic propositions,* $lab : S \to 2^{AP}$ *is a state labeling function, and* $\mathbf{R} : S \times Act \times S \to \mathbb{R}_{\geq 0}$ *is a rate matrix. We require that for any state* s *there exists a pair* $(a, s') \in Act \times S$ *with* $\mathbf{R}(s, a, s') > 0$.

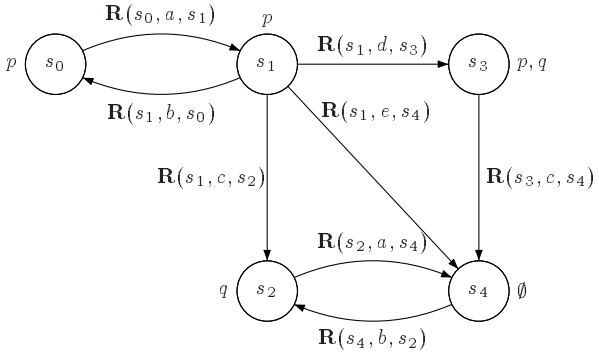

Figure 1. An ASMC

Intuitively, rate matrix $\mathbf{R}$ describes the transitions that can be made between states of the ASMC, on which actions, and with which rate. A transition from state s to state s', performing action a, exists if $\mathbf{R}(s, a, s') > 0$. A transition from s to s' performing a and of sojourn time $\tau \in \mathbb{R}_{>0}$ is denoted by $s \xrightarrow{a,\tau} s'$.

Definition 2.2 (Paths of $\mathcal{M}$) *A* finite path *of an ASMC* $\mathcal{M}$ *is a finite sequence of transitions* $\sigma = s_0 \xrightarrow{a_0,\tau_0} s_1 \xrightarrow{a_1,\tau_1} \ldots s_{n-1} \xrightarrow{a_{n-1},\tau_{n-1}} s_n$ *where* $\mathbf{R}(s_i, a_i, s_{i+1}) > 0$ *for* $i = 0, \ldots, n-1$*. An* infinite path *of* $\mathcal{M}$ *is an infinite sequence of transitions* $\sigma = s_0 \xrightarrow{a_0,\tau_0} s_1 \xrightarrow{a_1,\tau_1} \ldots$ *where* $\mathbf{R}(s_i, a_i, s_{i+1}) > 0$ *for all* $i \geq 0$ *and such that* $\sum_{i\geq 0} \tau_i = \infty$.

Notation. Given $s \in S$, let $Path^{\mathcal{M}}(s)$ be the set of infinite paths $s_0 \xrightarrow{a_0,\tau_0} s_1 \xrightarrow{a_1,\tau_1} \ldots$ such that $s_0 = s$. Let $\Pr_s^{\mathcal{M}}$ be the probability measure on $Path^{\mathcal{M}}(s)$ defined in the standard manner (for example, see [5, 4]). Let $\sigma = s_0 \xrightarrow{a_0,\tau_0} s_1 \xrightarrow{a_1,\tau_1} \ldots s_{n-1} \xrightarrow{a_{n-1},\tau_{n-1}} s_n$ be a finite path. Then $|\sigma| = n$ denotes the length of σ, and $\tau(\sigma) = \sum_{i=0}^{n-1} \tau_i$ is the total duration of σ. By convention, $\tau_{-1} = 0$. For an infinite path σ, we let $|\sigma| = \infty$ and $\tau(\sigma) = \infty$.

As usual, we can describe an ASMC by a graph (see Figure 1). The vertices of this graph are its states whereas the edges represent its transitions. The atomic propositions (here p and q) that are satisfied in a state are indicated near the corresponding node. Finally, the rate of a transition labels the corresponding edge.

We now present a restricted variant of timed automata [2], which are used in CSL$^{\text{TA}}$ to describe properties

of ASMC paths. More precisely, in our context, timed automata are used as acceptors of finite ASMC paths. The class of timed automata that we consider are deterministic (i.e., given a path of an ASMC, there is at most one path of the timed automaton which reads σ), and have a single clock. In the same manner as in classical analysis techniques for timed automata [2], we present our timed automata using natural-numbered constants (rational-numbered constants can also be considered through re-scaling) We proceed to define deterministic (one-clock) timed automata. As in asCSL, the symbol $\surd$ denotes a pseudo-action that is not included in the action set Act of any ASMC ($\surd \notin Act$). We consider the *clock* variable x. A *valuation* $\bar{x} \in \mathbb{R}_{\geq 0}$ is interpreted as assigning a real-valued value to x. A *constraint* is of the form $\alpha \prec x \prec \beta$ or $\alpha \prec x$ where $\alpha, \beta \in \mathbb{N}, \alpha \leq \beta$ and $\prec$ stands for either $<$ or $\leq$. An *inner constraint* is a constraint $\alpha \prec x \prec \beta$ such that $\alpha < \beta$. The set of inner constraints is denoted Inner. A *boundary constraint* is a constraint $\alpha \leq x \leq \beta$ such that $\alpha = \beta$; we generally write boundary constraints as $x = \alpha$. The set of boundary constraints is denoted $\mathsf{Boundary}$. Let χ be a constraint and $\bar{x}$ be a clock valuation. Then we write $\bar{x} \vDash \chi$ if χ is satisfied when $\bar{x}$ is substituted for x in χ.

A *state proposition* is a proposition which either holds, or does not hold, in an ASMC state. For a set Σ of state propositions, let $\models_\Sigma$ be its associated satisfaction relation: hence we write $\mathcal{M}, s \models_\Sigma \gamma$ to denote that the state s of the ASMC $\mathcal{M}$ satisfies γ. We omit $\mathcal{M}$ and write $s \models_\Sigma \gamma$ when clear from the context. We also consider boolean expressions of state propositions: for example $s \models_\Sigma \gamma_1 \wedge \gamma_2$ denotes that s satisfies γ_1 and γ_2. Let $\mathcal{B}(\Sigma)$ be the set of Boolean expressions over state propositions of Σ. We will make precise later in the paper the set of state propositions Σ used in CSL$^{\text{TA}}$. For the purposes of the current explanation, the reader can consider the case in which $\Sigma = AP$ with $s \models_{AP} a$ if and only if $a \in lab(s)$, for a state s and $a \in AP$.

Definition 2.3 (Deterministic Timed Automaton) *A* deterministic timed automaton *(DTA)* $\mathcal{A} = \langle \Sigma, Act, L, \Lambda, Init, Final, \rightarrow \rangle$ *comprises:*

- Σ*, a finite alphabet of state propositions;*
- Act*, a finite alphabet of actions;*
- L*, a finite set of locations;*
- $\Lambda : L \rightarrow \mathcal{B}(\Sigma)$*, a location labelling function;*
- $Init$*, a subset of* L *called the initial locations;*
- $Final$*, a subset of* L *called the final locations;*
- $\rightarrow \subseteq L \times ((\mathsf{Inner} \times 2^{Act}) \cup (\mathsf{Boundary} \times \{\surd\})) \times \{\emptyset, x\} \times L$*, a set of edges, where* $l \xrightarrow{\gamma, A, r} l'$ *denotes that* $(l, \gamma, A, r, l') \in \rightarrow$.

Furthermore $\mathcal{A}$ *fulfills the following conditions.*

Initial determinism: $\forall l, l' \in Init, \Lambda(l) \wedge \Lambda(l') \Leftrightarrow \mathtt{false}$.

Determinism on actions: $\forall A, A' \subseteq Act\, s.t.\, A \cap A' \neq \emptyset, \forall l, l', l'' \in L$, *if* $l'' \xrightarrow{\gamma, A, r} l \wedge l'' \xrightarrow{\gamma', A', r'} l'$ *then either* $\Lambda(l) \wedge \Lambda(l') \Leftrightarrow \mathtt{false}$ *or* $\gamma \wedge \gamma' \Leftrightarrow \mathtt{false}$.

Determinism on $\surd$: $\forall l, l', l'' \in L$, *if* $l'' \xrightarrow{\gamma, \surd, r} l \wedge l'' \xrightarrow{\gamma', \surd, r'} l'$ *then either* $\Lambda(l) \wedge \Lambda(l') \Leftrightarrow \mathtt{false}$ *or* $\gamma \wedge \gamma' \Leftrightarrow \mathtt{false}$.

No $\surd$-labelled loops: *For all sequences* $l_0 \xrightarrow{\gamma_0, A_0, r_0} l_1 \xrightarrow{\gamma_1, A_1, r_1} \cdots l_{n-1} \xrightarrow{\gamma_{n-1}, A_{n-1}, r_{n-1}} l_n$ *such that* $l_0 = l_n$, *there exists* $i \leq n$ *such that* $A_i \neq \surd$.

Notation. Edges labelled by $\surd$ are called *boundary edges* while the other edges are called *inner edges*. Given an edge $e = (l, \gamma, A, r, l') \in \rightarrow$, let $\mathsf{source}(e) = l$, $\mathsf{guard}(e) = \gamma$, $\mathsf{action}(e) = A$, $\mathsf{reset}(e) = r$, and $\mathsf{target}(e) = l'$. We let the valuation $\bar{x}[x := 0]$ be equal to 0 and let the valuation $\bar{x}[\emptyset := 0]$ be equal to $\bar{x}$.

The semantics of DTA, expressed in terms of paths, is standard [2], apart from the case of boundary edges, which are *urgent* and have *priority* over other edges. Urgency specifies that time cannot elapse if a boundary edge is enabled, and is a feature of variant of timed automata used in the tools UPPAAL [6] and KRONOS [16].

Definition 2.4 (Configurations and control switches of $\mathcal{A}$) *A* configuration *of a DTA* $\mathcal{A}$ *is a pair* $(l, \bar{x})$*, where* $l \in L$ *and* $\bar{x}$ *is a valuation. There is a* control switch $(l, \bar{x}) \xrightarrow{\delta, e} (l', \bar{x}')$ *of* $\mathcal{A}$ *corresponding to the passage of* $\delta \in \mathbb{R}_{\geq 0}$ *time units elapse from configuration* $(l, \bar{x})$*, after which the edge* $e \in \rightarrow$ *is taken to configuration* $(l', \bar{x}')$*, if the following conditions are satisfied:*

Standard requirements: $\mathsf{source}(e) = l$, $\bar{x} + \delta \vDash \mathsf{guard}(e)$, $\mathsf{target}(e) = l'$, *and* $\bar{x}' = \bar{x}[\mathsf{reset}(e) := 0]$;

Boundary edges are urgent: *for all* $0 \leq \delta' < \delta$, *there does not exist an edge* $e' \in \rightarrow$ *such that* $\mathsf{source}(e') = l$, $\bar{x} + \delta' \vDash \mathsf{guard}(e')$, *and* $\mathsf{action}(e') = \surd$;

Boundary edges have priority: *for all edges* $e' \in \rightarrow$ *such that* $e \neq e'$, *if* $\mathsf{source}(e') = l$ *and* $\bar{x} + \delta \vDash \mathsf{guard}(e')$, *then* $\mathsf{action}(e') \neq \surd$.

Definition 2.5 (Paths of $\mathcal{A}$) *A* finite path *of a DTA* $\mathcal{A}$ *is a finite sequence of control switches* $(l_0, \bar{x}_0) \xrightarrow{\delta_0, e_0} (l_1, \bar{x}_1) \xrightarrow{\delta_1, e_1} \ldots (l_{n-1}, \bar{x}_{n-1}) \xrightarrow{\delta_{n-1}, e_{n-1}} (l_n, \bar{x}_n)$.

An infinite path *of a DTA* $\mathcal{A}$ *is an infinite sequence of control switches* $(l_0, \bar{x}_0) \xrightarrow{\delta_0, e_0} (l_1, \bar{x}_1) \xrightarrow{\delta_1, e_1} \ldots$.

We now give an intuitive explanation of how a path $\sigma = s_0 \xrightarrow{a_0, \tau_0} s_1 \xrightarrow{a_1, \tau_1} \ldots$ of an ASMC $\mathcal{M}$ can be used as input to a DTA $\mathcal{A}$. The key idea is that $\mathcal{A}$ performs control switches according to the states and actions that it "reads" along σ. Recalling that the value of clocks in timed automata increase at the same rate as real-time, as time elapses in $\mathcal{M}$, then the value of the clock x of $\mathcal{A}$ also changes accordingly. Control switches corresponding to inner edges of $\mathcal{A}$ *are triggered by transitions of* $\mathcal{M}$, whereas control

switches of boundary edges of $\mathcal{A}$ *are taken autonomously, without a corresponding transition of* $\mathcal{M}$.

The DTA $\mathcal{A}$ begins in a configuration $(l_0, 0)$ with location $l_0 \in Init$ such that the initial state s_0 of σ satisfies the expression $\Lambda(l_0)$ over state propositions (formally, $s_0 \models_\Sigma \Lambda(l_0)$). Note that, by initial determinism, there is at most one $l \in Init$ such that s_0 satisfies $\Lambda(l)$. If s_0 does not satisfy $\Lambda(l)$ for all $l \in Init$, then $\mathcal{A}$ rejects σ.

Given the existence of $l_0 \in Init$ such that s_0 satisfies $\Lambda(l_0)$, the DTA $\mathcal{A}$ then moves from $(l_0, 0)$ to another configuration depending on the first transition $s_0 \xrightarrow{a_0, \tau_0} s_1$ of σ. First we consider the case in which there are no outgoing boundary edges from l_0. If there exists a control switch $(l_0, 0) \xrightarrow{\tau_0, e_0} (l_1, \bar{x}_1)$ such that $a_0 \in \mathsf{action}(e_0)$ and s_1 satisfies $\Lambda(l_1)$, then this control switch is taken. Note that, by determinism on actions, there exists at most one control switch satisfying these conditions. If no such control switch exists, then $\mathcal{A}$ rejects σ.

Now we consider the case in which there exists at least one boundary edge from l_0. Consider the control switch $(l_0, 0) \xrightarrow{\delta_0', e_0'} (l_1', \bar{x}_1')$, where $\mathsf{action}(e_0') = \surd$, which (by urgency of boundary edges) corresponds to the earliest boundary edge available by letting time elapse from $(l_0, 0)$. If $\delta_0' > \tau_0$, then this control switch is available only *after* $\mathcal{M}$ has performed the transition $s_0 \xrightarrow{a_0, \tau_0} s_1$; hence, the DTA "reads" the ASMC transition *before* the boundary edge is available, and this case is similar to the case in which there are no boundary edges from l_0 in the previous paragraph. If, however, $\delta_0' \leq \tau_0$, the DTA takes the control switch before "reading" the ASMC transition. Note that, in this case, the remaining time before the transition of $\mathcal{M}$ must be "read" by $\mathcal{A}$ is $\tau_0 - \delta_0'$, rather than τ. This has implications for deciding whether a boundary edge can be taken from $(l_1', \bar{x}_1')$, or whether the transition of $\mathcal{M}$ must be "read" before any boundary edge is enabled for choice.

Unless $\mathcal{A}$ has already rejected σ, the path of $\mathcal{A}$ generated by σ then continues from $(l_1, \bar{x}_1)$ or $(l_1', \bar{x}_1')$. Finally, if the path of $\mathcal{A}$ generated by σ reaches a configuration with a location in $Final$, then the run of $\mathcal{A}$ generated by σ is accepted. If, however, the path of $\mathcal{A}$ generated by σ does not reach such a configuration, then σ is rejected. Hence, there are two ways in which $\mathcal{A}$ can reject σ: if there does not exist an control switch corresponding to the "reading" of a transition of σ, or if a final location is never reached.

Examples of DTA: Next and Until. Let Φ_1 and Φ_2 be state propositions in the alphabet Σ of the DTA that we consider. In Figure 2 we illustrate a DTA, using the usual conventions (i.e., nodes represent locations, and edges represent transitions labelled with their guards, actions sets, and the set of clocks to be reset to 0, respectively). Initial locations are denoted by an incoming arrow with no source, and final locations are denoted by a double border. The DTA $\mathcal{A}_{\mathcal{X}^{[\alpha,\beta]}\Phi_1}$

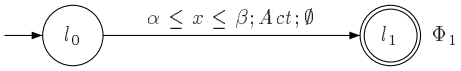

Figure 2. The DTA $\mathcal{A}_{\mathcal{X}^{[\alpha,\beta]}\Phi_1}$

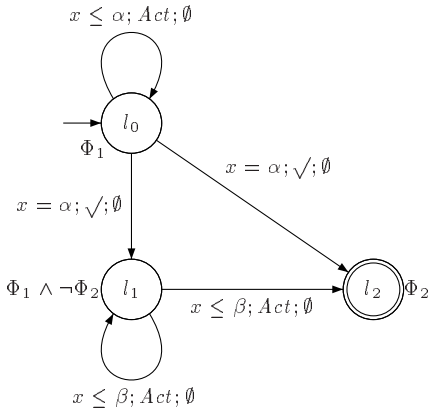

Figure 3. The DTA $\mathcal{A}_{\Phi_1 \mathcal{U}^{[\alpha,\beta]}\Phi_2}$

in Figure 2 specifies behaviours in which the first transition must be taken to a state satisfying Φ_1 after at least α time units, but not after β time units, and corresponds to the Next path formula $\mathcal{X}^{[\alpha,\beta]}\Phi_1$ of CSL [5]. The action of the transition is not important; this fact is represented by the action set Act on the edge of the DTA.

We can use the DTA $\mathcal{A}_{\Phi_1 \mathcal{U}^{[\alpha,\beta]}\Phi_2}$ of Figure 3 to represent the property of eventually reaching a state satisfying Φ_2 at some instant between α and β time units, remaining within states satisfying Φ_1 before that point (the timed Until path property $\Phi_1 \mathcal{U}^{[\alpha,\beta]}\Phi_2$ of CSL [5]). In contrast to the previous example, this DTA uses boundary edges which witness that the time interval $[\alpha, \beta]$ has been entered. In this way, we distinguish between the time interval $[0, \alpha)$, where the truth value of Φ_2 is irrelevant, and the time interval $[\alpha, \beta]$, where the truth value of Φ_2 becomes relevant.

Path acceptance. We now describe formally the conditions for the acceptance of an ASMC path by a DTA.

Definition 2.6 *Let $\mathcal{M}$ be an ASMC, and let $\mathcal{A}$ be a DTA. The infinite path $\sigma_\mathcal{M} = s_0 \xrightarrow{a_0, \tau_0} s_1 \xrightarrow{a_1, \tau_1} \ldots$ of $\mathcal{M}$ is* accepted *by $\mathcal{A}$ if there exists:*

1. *a finite path $\sigma_\mathcal{A} = (l_0, \bar{x}_0) \xrightarrow{\delta_0, e_0} (l_1, \bar{x}_1) \xrightarrow{\delta_1, e_1} \ldots (l_{m-1}, \bar{x}_{m-1}) \xrightarrow{\delta_{m-1}, e_{m-1}} (l_m, \bar{x}_m)$ of $\mathcal{A}$,*
2. *an index $n \in \mathbb{N}$,*
3. *a time $\tau \leq \tau_n$, and*
4. *a function $\kappa : \{0, \ldots, m\} \to \{0, \ldots, n\}$ which maps indices of $\sigma_\mathcal{A}$ to indices of $\sigma_\mathcal{M}$,*

such that the following conditions are satisfied:

- $l_0 \in Init$, $\bar{x}_0 = 0$, $\kappa(0) = 0$ *and* $\forall 0 \leq i \leq m, l_i \in Final \Leftrightarrow i = m$;
- $\forall 0 \leq i \leq m$, $s_{\kappa(i)} \models_\Sigma \Lambda(l_i)$;

- $\forall 0 \leq i \leq m$, *if* e_i *is an inner edge then* $\kappa(i+1) = \kappa(i) + 1 \wedge a_{\kappa(i)} \in \mathsf{action}(e_i)$ *else* $\kappa(i+1) = \kappa(i)$;
- $\forall 0 \leq i < n$, $\sum_{j|\kappa(j)=i} \delta_j = \tau_{\kappa(i)}$;
- $\sum_{j|\kappa(j)=n} \delta_j = \tau$.

The first condition specifies that $\sigma_{\mathcal{A}}$ must start from an initial location and end in a final location. The second condition requires that the state propositions must satisfy the corresponding locations in the sequence. The third condition specifies that κ can map different (but consecutive) indices of $\sigma_{\mathcal{A}}$ to the same index of $\sigma_{\mathcal{M}}$, provided that the DTA edges corresponding to these indices are boundary edges. It also requires that a transition of the ASMC in σ can be matched by a traversal of an inner edge provided that the action of the transition is included in the action set of the edge. The fourth condition requires that the sum of durations in $\sigma_{\mathcal{A}}$ corresponding to a particular index i of σ is τ_i. The fifth point applies the reasoning of the fourth point to the case in which the path $\sigma_{\mathcal{A}}$ features boundary edges directly before reaching a final state.

It should be clear, due to our requirements for DTA, that given an ASMC $\mathcal{M}$ and a DTA $\mathcal{A}$, there is at most one path of $\mathcal{A}$ that accepts a given path of $\mathcal{M}$. Accordingly, if s is a state of $\mathcal{M}$ we let $AccPath^{\mathcal{M}}(s, \mathcal{A})$ be the set of infinite paths of $\mathcal{M}$ starting from s that are accepted by $\mathcal{A}$.

Examples of path acceptance. In Figure 4, we present two examples of the way in which a path of the ASMC $\mathcal{M}$ of Figure 1 can be accepted by the DTA $\mathcal{A}_{p\mathcal{U}^{[\alpha,\beta]}q}$. We write e_{ij} to refer to the edge of $\mathcal{A}_{p\mathcal{U}^{[\alpha,\beta]}q}$ from location l_i to location l_j, and we use dotted lines to represent the κ function. Note that if we have more than one dotted line from a state s_i, then the DTA performs a $\sqrt{}$. Example 1 of Figure 4 has $\alpha = 2$ and $\beta = 6$ and depicts a case in which q does not hold at time α, but becomes true at time 5, which belongs to $[\alpha, \beta]$; therefore the DTA reaches l_2 through l_1. Example 2 of Figure 4 has $\alpha = 6$ and $\beta > 6$ and depicts a case in which q already holds before α; therefore the DTA reaches l_2 directly from l_0.

We now describe briefly some examples of paths of $\mathcal{M}$ which are rejected by $\mathcal{A}_{p\mathcal{U}^{[2,6]}q}$. If the first transition of $\mathcal{M}$ is $s_0 \xrightarrow{a,7} s_1$, then the associated path of $\mathcal{A}_{p\mathcal{U}^{[2,6]}q}$ will consist of the single control switch $(l_0, 0) \xrightarrow{e_{01},2} (l_1, 2)$: after the value of the clock x exceeds 6, it will not be possible to take further control switches. If on the other hand the path of $\mathcal{M}$ is $s_0 \xrightarrow{a,1} s_1 \xrightarrow{c,0.5} s_2$, then the associated path of $\mathcal{A}_{p\mathcal{U}^{[2,6]}q}$ will consist of the single control switch $(l_0, 0) \xrightarrow{e_{00},1} (l_0, 1)$, after which it will not be possible to take any further control switches: the state s_2 is not labelled by p, boundary edges are available only at time 2, and yet the transition $s_1 \xrightarrow{c,0.5} s_2$ occurs before time 2.

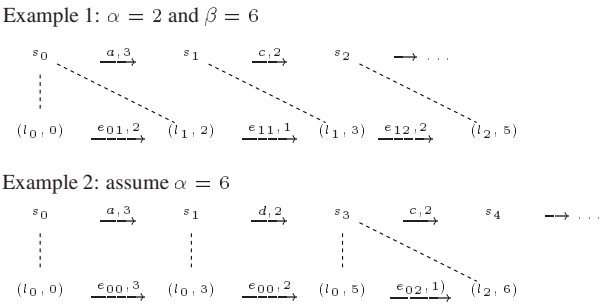

Figure 4. Examples of paths

2.2 CSL$^{\text{TA}}$

Given the definition of DTA, we can now present formally the syntax of CSL$^{\text{TA}}$. Note that the syntax of CSL$^{\text{TA}}$ is essentially identical to that of CSL or asCSL [3, 5, 4], apart from the fact that properties of paths are specified using DTA (instead of being specified by timed temporal logic operators as, for example, in CSL).

Definition 2.7 (Syntax of CSL$^{\text{TA}}$) *Let* $\lambda \in [0, 1]$ *be a real number, and let* $\sim \in \{\leq, <, >, \geq\}$ *be a comparison operator. The syntax of CSL^{TA} is defined by:*

$$\Phi ::= p \mid \neg\Phi \mid \Phi \wedge \Phi \mid \mathcal{S}_{\sim\lambda}(\Phi) \mid \mathcal{P}_{\sim\lambda}(\mathcal{A}(\Phi_1, \ldots, \Phi_n))$$

where $p \in AP$, $\lambda \in [0, 1]$ *is a real number,* $\sim \in \{\leq, <, >, \geq\}$ *is a comparison operator, and* $\mathcal{A}(\Phi_1, \ldots, \Phi_n)$ *is a DTA with a finite alphabet* Σ *of state propositions such that* $\Sigma = \{\Phi_1, \ldots, \Phi_n\}$ *and* $\Phi_1, \ldots, \Phi_n$ *are CSL^{TA} formulas.*

Note that CSL$^{\text{TA}}$ is a CTL*-like language with nesting of path and state formulae [9]; in particular, the state propositions of a DTA are state formulae of CSL$^{\text{TA}}$. For example, we can write a CSL$^{\text{TA}}$ formula such as $\mathcal{P}_{\geq 0.99}(\mathcal{A}_{p\mathcal{U}^{[\alpha,\beta]}\mathcal{P}_{\geq 0.1}(\mathcal{A}_{\mathcal{X}^{[\alpha,\beta]}q})})$ (which corresponds to the CSL formula $\mathcal{P}_{\geq 0.99}(p\mathcal{U}^{[\alpha,\beta]}\mathcal{P}_{\geq 0.1}(\mathcal{X}^{[\alpha,\beta]}q))$).

Intuitively, the state s satisfies the formula $\mathcal{S}_{\sim\lambda}(\Phi)$ if and only if, starting the execution of the ASMC from s, in the steady-state the probability val that Φ is true fulfills $val \sim \lambda$. The state s satisfies the formula $\mathcal{P}_{\sim\alpha}(\mathcal{A})$ if and only if the probability val that the execution of $\mathcal{A}$ triggered by a random path of the ASMC from s is successful (i.e., stops in a final location) fulfills $val \sim \lambda$.

We proceed to define the semantics of CSL$^{\text{TA}}$ in terms of the satisfaction relation $\models$. Intuitively, for a given CSL$^{\text{TA}}$ formula Φ and state s of $\mathcal{M}$, we write $\mathcal{M}, s \models \Phi$ to denote that Φ is satisfied in state s. We write $\pi(s, \cdot)$ for the steady-state distribution of $\mathcal{M}$ starting from state s.

Definition 2.8 (Semantics of CSL$^{\text{TA}}$) *The satisfaction relation* $\models$ *for CSL^{TA} is defined as follows:*

$$
\begin{array}{lcl}
\mathcal{M}, s \models p & \Leftrightarrow & p \in lab(s) \\
\mathcal{M}, s \models \neg\Phi & \Leftrightarrow & \mathcal{M}, s \not\models \Phi \\
\mathcal{M}, s \models \Phi_1 \wedge \Phi_2 & \Leftrightarrow & \mathcal{M}, s \models \Phi_1 \text{ and } \mathcal{M}, s \models \Phi_2 \\
\mathcal{M}, s \models \mathcal{S}_{\sim\lambda}(\Phi) & \Leftrightarrow & \sum_{s' \models \Phi} \pi(s, s') \sim \lambda \\
\mathcal{M}, s \models \mathcal{P}_{\sim\lambda}(\mathcal{A}(\Phi_1, \ldots, \Phi_n)) & & \\
\quad \Leftrightarrow \Pr_s^{\mathcal{M}}(AccPath^{\mathcal{M}}(s, \mathcal{A}(\Phi_1, \ldots, \Phi_n))) \sim \lambda\ . & &
\end{array}
$$

3 Model checking for CSL$^{\text{TA}}$

As usual with CTL*-like languages [9], in order to evaluate the satisfaction of a formula Φ over an ASMC $\mathcal{M}$, we proceed by a bottom-up evaluation of the subformulas occurring in Φ over all the states of $\mathcal{M}$, labelling accordingly the states with the subformulas that they satisfy. Let Φ' be such a subformula. If Φ' is either an atomic proposition p, $\neg\Psi$ or $\Psi \wedge \Psi'$, then the evaluation is performed by a straightforward application of Definition 2.8. If $\Phi' = \mathcal{S}_{\sim\lambda}(\Psi)$, then first one computes the steady-state of $\mathcal{M}$ w.r.t. to every state s. We then compute the steady-state probability of the subset of states that fulfill Ψ and compare it with λ in order to check whether s satisfies Φ'. Finally, if $\Phi' = \mathcal{P}_{\sim\lambda}(\mathcal{A})$, for each state s, we compute the probability of $AccPath^{\mathcal{M}}(s, \mathcal{A})$ (the set of paths of $\mathcal{M}$ accepted by $\mathcal{A}$), and we compare it to λ in order to check whether s satisfies Φ'. The computation of $\Pr_s^{\mathcal{M}}(AccPath^{\mathcal{M}}(s, \mathcal{A}))$ is the topic of the remainder of this section. We use s_0 to denote the state for which we compute $\Pr_{s_0}^{\mathcal{M}}(AccPath^{\mathcal{M}}(s_0, \mathcal{A}))$, and let $l_0 \in Init$ be the location of $\mathcal{A}$ for which $s_0 \models_\Sigma \Lambda(l_0)$ (if no such location exists then $\Pr_{s_0}^{\mathcal{M}}(AccPath^{\mathcal{M}}(s_0, \mathcal{A})) = 0$.

The "synchronized" stochastic process $\mathcal{M} \times \mathcal{A}$. The computation of $\Pr_{s_0}^{\mathcal{M}}(AccPath^{\mathcal{M}}(s_0, \mathcal{A}))$ requires the definition of a stochastic process $\mathcal{M} \times \mathcal{A}$ that describes the joint evolution of $\mathcal{M}$ and $\mathcal{A}$. At some instant of its execution, this process may be in one of three situations. (1) At some previous instant, $\mathcal{A}$ has not be able to mimic the execution of $\mathcal{M}$ and thus this process is in the absorbing state $\perp$ whatever are the subsequent transitions of $\mathcal{M}$. (2) At some previous instant, $\mathcal{A}$ has reached a final location by following the execution of $\mathcal{M}$ and thus this process is in the absorbing state $\top$ whatever are the subsequent transitions of $\mathcal{M}$. (3) Otherwise the process is in some state of $\mathcal{M}$ associated with a finite timed execution of $\mathcal{A}$ not ending in a final location.

States of $\mathcal{M} \times \mathcal{A}$. If at some instant the execution of $\mathcal{M}$ is neither rejected nor accepted, we observe that, for the future behaviour of the process $\mathcal{M} \times \mathcal{A}$, only the current location of the path in the TA and the value of clock x are relevant. This yields the following state description: $N(t) = (s(t), l(t), \bar{x}(t))$, where $s(t)$ is the state of $\mathcal{M}$ at time $t \in \mathbb{R}_{\geq 0}$, $l(t)$ is a location of $\mathcal{A}$ at time t, and $\bar{x}(t)$ is the value of the clock at time t. However, in $\mathcal{M} \times \mathcal{A}$ we consider only *tangible* states, i.e., states which do not trigger a boundary edge. Therefore we introduce the following definition (which is sound because there are no loops of $\surd$ transitions in $\mathcal{A}$).

Definition 3.1 *Let $(s, l, v) \in S \times L \times \mathbb{R}_{\geq 0}$. Then $closure(s, l, \bar{x})$ is defined as follows: if $l \in Final$ then $closure(s, l, \bar{x}) = \top$; if $l \notin Final$ and there is a boundary edge $l \xrightarrow{\gamma, \surd, r} l'$ with $\bar{x} \models \gamma$ and $s \models_\Sigma \Lambda(l')$ then $closure(s, l, \bar{x}) = closure(s, l', \bar{x}[r := 0])$; otherwise $closure(s, l, \bar{x}) = (s, l, \bar{x})$.*

The set of states of the process $\mathcal{M} \times \mathcal{A}$ is a subset of $\{\perp, \top\} \cup \{(s, l, \bar{x}) \mid closure(s, l, \bar{x}) = (s, l, \bar{x}),\ s \models_\Sigma \Lambda(l)\}$.

Behaviour of $\mathcal{M} \times \mathcal{A}$. Let $C = \{c_0, \ldots, c_m\}$ be the set of constants used in the clock constraints of $\mathcal{A}$ enlarged with 0, ordered as follows: $0 = c_0 < c_1 < \cdots < c_m$. We define $\mathsf{next}(c_i) = c_{i+1}$ for all $i < m$ and $\mathsf{next}(c_m) = \infty$.

Let $(s, l, \bar{x})$ be a state such that $\bar{x} \in [c_i, \mathsf{next}(c_i))$ for some $i \leq m$. Then the process $\mathcal{M} \times \mathcal{A}$ can evolve from $(s, l, \bar{x})$ due to the ASMC $\mathcal{M}$ changing its state by a transition $s \xrightarrow{a, \tau} s'$ before the next timing constant $\mathsf{next}(c_i)$ is reached, i.e., $\bar{x} + \tau < \mathsf{next}(c_i)$. If this transition of $\mathcal{M}$ cannot be mimicked by $\mathcal{A}$ then the stochastic process makes a transition to $\perp$. Otherwise, there is an edge (l, γ, A, r, l') in $\mathcal{A}$ with $a \in A$, which mimics this transition. In this case, $\mathcal{M} \times \mathcal{A}$ reaches the new state $closure(s', l', (v + \tau)[r := 0])$ ($closure$ is needed since boundary transitions may be then triggered in zero time and/or $\mathcal{A}$ may reach a final state).

We also note that, if $\bar{x} \in [c_i, \mathsf{next}(c_i))$ for some $i < m$, the process $\mathcal{M} \times \mathcal{A}$ can evolve from $(s, l, \bar{x})$ due to time $\mathsf{next}(c_i) - \bar{x}$ elapsing (i.e., the next timing constant is reached) before $\mathcal{M}$ changes state. Then the new state is $closure(s, l, \mathsf{next}(c_i))$.

It is straightforward to show that each path of $\mathcal{M} \times \mathcal{A}$ leading to $\top$ corresponds to a single path in $\mathcal{M}$ accepted by $\mathcal{A}$, and vice versa. Furthermore, $\Pr_{s_0}^{\mathcal{M}}(AccPath^{\mathcal{M}}(s_0, \mathcal{A}))$ can be computed as the probability of reaching $\top$ in process $\mathcal{M} \times \mathcal{A}$ from $(s_0, l_0, 0)$. In the remainder of this section we explain how this probability can be computed.

$\mathcal{M} \times \mathcal{A}$ is a Markov Renewal Process. We can rewrite a state of $\mathcal{M} \times \mathcal{A}$ in terms of the last clock constant reached, as follows: $N(t) = (s(t), l(t), c(t), \bar{x}(t) - c(t))$ where $c(t)$ is the largest $c \in C$ such that $c \leq \bar{x}(t)$

We now show that $\mathcal{M} \times \mathcal{A}$ is a Markov renewal process (MRP). For the definition of MRP and Markov renewal sequences, see, for example, [11]. Consider a sequence $\{T_k, k = 0, 1, 2, \ldots\}$ of strictly increasing timing instants in the evolution of $\mathcal{M} \times \mathcal{A}$, with $N(t_k) = (s_k, l_k, c(T_k), \bar{x}_k - c(T_k))$. The timing instants are defined as follows: (1) $T_0 = 0$, (2) if $\bar{x}_k \geq c_m$ then T_{k+1} is the first time after T_k that an event occurs, (3) if $\bar{x}_k < c_m$ then T_{k+1} is the next time at which the next constant in C is reached or the clock x is reset to 0. Let $Y_k = N(T_k^+)$ be the state right after all the events as time T_k.

Theorem 3.2 *$(Y,T) = \{(Y_k,T_k), k = 0,1,2,\ldots\}$ is a Markov renewal sequence and $N(t)$ is an MRP.*

The proof of Theorem 3.2 is straightforward given the definition of MRP (see [11]), because, due to the definition of T_k, we have that $Y_k = (s(T_k^+), l(T_k^+), c(T_k))$ where $c(T_k) \in C$ is the value of the clock at T_k, and the joint distribution of Y_{k+1} and $T_{k+1} - T_k$ only depends on Y_k. Therefore (Y,T) is a Markov renewal sequence. Moreover $N(t)$ is a MRP because $N(T_k + \delta)$, $0 \le \delta \le T_{k+1} - T_k)$ only depends on Y_k.

It is well-known that $Y = \{Y_k, k = 0,1,2,\ldots\}$ is a DTMC (the embedded DTMC of the MRP). In general the solution of an MRP requires the definition of the global and local kernel matrices (see [11]). The computation of the probability of reaching the absorbing state $\top$ from the initial state can be performed on the DTMC $P_{i,j}$ which expresses the probability that, if i is the state at regeneration instant 0, then j is the state at the next regeneration instant T_1 (that is, $P_{ij} = Pr\{Y_1 = j | Y_0 = i\}$).

Tangible Reachability Graph of $\mathcal{M} \times \mathcal{A}$. We next define a data structure that supports the definition of the DTMC and the computation of its transition probabilities. This data structure is called Tangible Reachability Graph (TRG), and is inspired by the identically-named graph of Deterministic SPNs [1], in which the elapsing of time between two consecutive timing constants c and $\mathsf{next}(c)$ is interpreted as a deterministic "transition" of duration $\mathsf{next}(c) - c$. Note that in our case a deterministic "transition" can only be preempted by a transition of $\mathcal{M} \times \mathcal{A}$ that includes a clock reset.

The nodes of the TRG take the form of elements of $(S \times L \times C) \cup \{\bot, \top\}$. The arcs between nodes of the TRG are defined by the following four rules:

[M]: a simple Markovian move, in which the ASMC $\mathcal{M}$ moves "according to" the DTA $\mathcal{A}$ and there is no clock reset. Formally, there exists the arc $(s,l,c) \xrightarrow{\mathbf{M}(a,e)} (s',l',c)$ if (1) $\mathbf{R}(s,a,s') > 0$, (2) $e = (l,\gamma,A,\emptyset,l')$ is an inner edge of $\mathcal{A}$ such that $(c, \mathsf{next}(c)) \vDash \gamma$, $a \in A$ and $s' \models_\Sigma \Lambda(l')$, and (3) $l' \notin Final$. Furthermore, there exists the arc $(s,l,c) \xrightarrow{\mathbf{M}(a,e)} \top$ if the conditions (1) and (2) above are satisfied, and $l' \in Final$.

[M_res]: as for a simple Markovian move, but with a clock reset that can start an evolution of $\mathcal{A}$ over boundary transitions. Formally, there exists the arc $(s,l,c) \xrightarrow{\mathbf{M\_res}(a,e)} closure(s',l',0)$ if (1) $\mathbf{R}(s,a,s') > 0$ and (2) $e = (l,\gamma,A,x,l')$ is an inner edge of $\mathcal{A}$ such that $(c, \mathsf{next}(c)) \vDash \gamma$, $a \in A$ and $s' \models_\Sigma \Lambda(l')$.

[M_KO]: a Markovian move that is not accepted by $\mathcal{A}$. Formally, there exists the arc $(s,l,c) \xrightarrow{\mathbf{M\_KO}(a)} \bot$ if there exists $s' \in S$ such that $\mathbf{R}(s,a,s') > 0$ and there does not exist an inner edge $e = (l,\gamma,A,r,l')$ of $\mathcal{A}$ such that $(c, \mathsf{next}(c)) \vDash \gamma$, $a \in A$ and $s' \models_\Sigma \Lambda(l')$.

[D]: let time elapse. Formally, there exists the arc $(s,l,c) \xrightarrow{\mathbf{D}} closure(s,l,\mathsf{next}(c))$ if $c < c_m$.

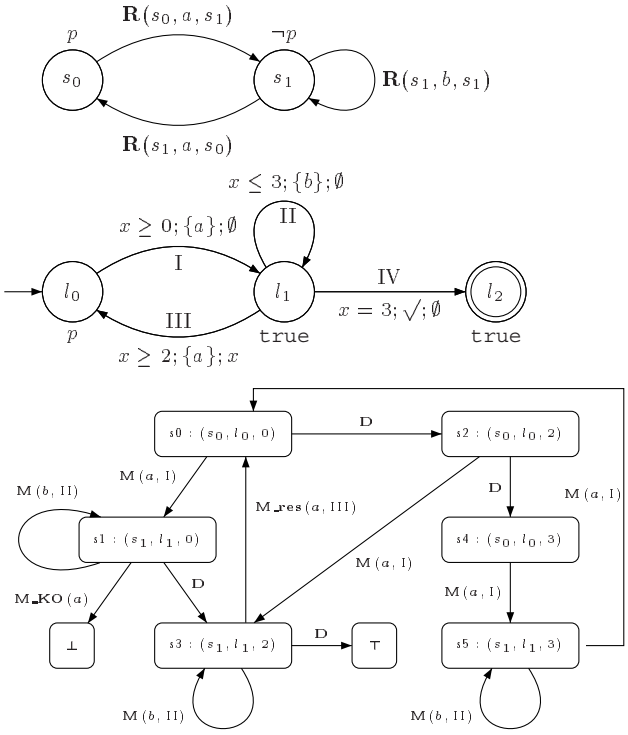

Figure 5. An ASMC $\mathcal{M}$, a DTA $\mathcal{A}$, and their TRG

Note that there is a single arc from a node (s,l,c) due to a transition (s,a,s') in the ASMC, because of the assumption of determinism of $\mathcal{A}$, and that there is at most one **D** arc from a node.

We now define TRS as the set of nodes reachable from the set of states $(s,l,0)$, for all $s \in S$ and $l \in Init$, with $s \models_\Sigma \Lambda(l)$, applying the reachability expressed by the four rules above (note that we all considering all states $s \in S$ since satisfaction need to be checked on all states of $\mathcal{M}$). Then the TRG of $\mathcal{M} \times \mathcal{A}$ is defined as the graph over TRS where the arcs are described as above.

Observe that, if (s,l,c) is a node of the TRG, then any $(s,l,c+\delta)$ with $0 \le \delta \le \mathsf{next}(c) - c$ is a state of the MRP $N(t)$, and that a **D**-arc to a node (s,l,c) means that upon event **D** the state of $\mathcal{M} \times \mathcal{A}$ is exactly (s,l,c), while if the same state is entered through an **M**-arc, the state of the process can be $(s,l,c+\delta)$ for any $0 < \delta < \mathsf{next}(c)$.

The upper part of Figure 5 shows an ASMC $\mathcal{M}$ and a DTA $\mathcal{A}$. DTA edges have been tagged with roman numerals to cross-reference them in the TRG of $\mathcal{M} \times \mathcal{A}$ shown in the lower part.

To compute the probability of reaching $\top$, we need to identify in the TRG the states of the DTMC and the associated transition probabilities.

Definition 3.3 *Let* $\mathsf{s} \in TRS$. *Then* s *is a state of the DTMC embedded into the MRP* (Y,T) *if either: (1)* s *can be entered by an arc labelled* **D** *or* **M_res**, *(2)* $\mathsf{s} = (s,l,c)$ *with* $l \in Init$ *and* $c = 0$ *(initial states), (3)* $\mathsf{s} = \top$, *or (4)* $\mathsf{s} = \bot$.

Note that not all states of the TRG are states of the DTMC: indeed in the example of Figure 5 state s5 is not a state of the DTMC (there is no renewal point T_k such that $Y(T_k) = \mathsf{s5}$).

To compute the probabilities of the DTMC, we need to define, for each state $(s,l,c) \in TRS \setminus \{\bot,\top\}$ of the DTMC, how the process $\mathcal{M} \times \mathcal{A}$ can evolve before reaching the next regeneration point. This (transient) behaviour is driven by the subordinated CTMC $\mathcal{C}_{(s,l,c)}$ that describes the evolution of the process from (s,l,c) until a successive state of $\mathcal{M} \times \mathcal{A}$ is reached, either due to a state change in $\mathcal{M}$, due to the clock having reached $\mathtt{next}(c)$, or due to the clock being reset.

The states of the subordinated CTMC $\mathcal{C}_{(s,l,c)}$ can be computed from (s,l,c) by taking in the TRG the transitive closure over arcs of type **M**, possibly followed by a **M_res**-arc or a **M_KO**-arc. For simplicity we separate the cases $c \neq 0$ and $c = 0$. If $c \neq 0$, then:

- $(s,l,c) \in \mathcal{C}_{(s,l,c)}$;
- $(s',l',c) \in \mathcal{C}_{(s,l,c)}$ if there exists a path in the TRG from (s,l,c) to (s',l',c) of arcs all of type **M**;
- $\top \in \mathcal{C}_{(s,l,c)}$ if there exists a path in the TRG from (s,l,c) to $\top$ in which all the arcs are of type **M**, or which ends in an **M_res**-arc and for which all other arcs (if any) are of type **M**;
- $(s',l',0) \in \mathcal{C}_{(s,l,c)}$ if there exists a (possible empty) path in the TRG from (s,l,c) to (s'',l'',c) of arcs all of type **M**, and an arc from (s'',l'',c) to $(s',l',0)$ of type **M_res**.
- $\bot \in \mathcal{C}_{(s,l,c)}$ if there exists a (possible empty) path in the TRG from (s,l,c) to (s'',l'',c) of arcs all of type **M**, and a **M_KO** arc from (s'',l'',c) to $(s',l',0)$.

When $c = 0$ we need to distinguish in the CTMC $\mathcal{C}_{(s,l,0)}$ the state $(s,l,0)^{Reset}$, entered upon a clock reset, from the state $(s,l,0)$ entered through a Markovian transition: indeed we need to compute for $\mathcal{C}_{(s,l,0)}$ the probability of being in the various states of $\mathcal{C}_{(s,l,0)}$ at the next regeneration point, given that $\mathcal{C}_{(s,l,0)}$ starts in $(s,l,0)$, and therefore we need to distinguish the two cases. Observe that when $c > 0$ it is never the case that (s,l,c) can be entered through a non-Markovian transition. Note that in the TRG a **M_res** transition corresponds in DSPNs to the case of an exponential transition that preempts a deterministic transition and then immediately re-enables it, which, as explained in [11], requires a duplication of the states of the subordinated CTMC.

The rates of the CTMC $\mathcal{C}_{(s,l,c)}$ can be computed directly from the rates of the transitions of $\mathcal{M}$ that cause the change of state in the TRG. The states of the subordinated CTMC $\mathcal{C}_{(s,l,c)}$ are of the form (s',l',c') for $s' \in S$, $l' \in L$ and $c' \in \{0,c\}$, or $\bot$ or $\top$. The transition probabilities of the DTMC $P_{(s,l,c)(s',l',c')}$ are computed in $\mathcal{C}_{(s,l,c)}$ (with one subordinated CTMC per each row of P) as follows. If $c < c_m$ then $P_{(s,l,c),(s',l',\mathtt{next}(c))}$ is the probability of being at time $\mathtt{next}(c) - c$ in state (s',l',c); furthermore $P_{(s',l',c),j}$, with $j \in \{(s',l',0),\top,\bot\}$, is the probability of being in j at time $\mathtt{next}(c) - c$. However, if $c = c_m$, then we compute on $\mathcal{C}_{(s,l,c_m)}$ the probability to reach each absorbing state, i.e., $\top$, $\bot$ or some $(s',l',0)$ (which has been reached by an inner edge with a clock reset).

Note that there are two peculiarities of the embedded DTMC. First, we can re-enter the same state due to a clock reset. This has no effect on the computation. Second, the transition matrix can be substochastic, because for some DTMC states there is a non-null probability to never reach another state of the MRP. Again, this is not problematic, because the reachability probability computation with a substochastic matrix is identical as with a stochastic transition matrix.

4 Expressiveness of CSL$^{\mathrm{TA}}$

In this section we study the relationship between CSL$^{\mathrm{TA}}$, CSL [5], and asCSL [4]. Formulae interpreted on ASMCs are described as being equivalent if, for any ASMC, the same states of the ASMC satisfy the formulae. Formally, we say that Φ_1 (of the logic Log_1, with the satisfaction relation $\models_{Log_1}$) *is equivalent to* Φ_2 (of the logic Log_2, with the satisfaction relation $\models_{Log_2}$), if, for any ASMC $\mathcal{M}$, and for any state s of $\mathcal{M}$, we have: $\mathcal{M}, s \models_{Log_1} \Phi_1 \Leftrightarrow \mathcal{M}, s \models_{Log_2} \Phi_2$.

The following two propositions show that CSL$^{\mathrm{TA}}$ is strictly more expressive than CSL.

Proposition 4.1 *For any formula* Φ *of CSL there is a formula* Φ' *of CSL^{TA} equivalent to* Φ. *Furthermore the size of* Φ' *is linear with respect to the size of* Φ.

Proof sketch. The semantics of constructors for state formulas are identical for CSL and CSL$^{\mathrm{TA}}$; therefore it suffices to prove that any path formula of CSL is equivalent to some path formula of CSL$^{\mathrm{TA}}$. The two DTA $\mathcal{A}_{\mathcal{X}^{[\alpha,\beta]}\Phi}$ and $\mathcal{A}_{\Phi_1\mathcal{U}^{[\alpha,\beta]}\Phi_2}$ (see Figure 2 and Figure 3, respectively) have already been presented in Section 2, and encode the Next formula $(\mathcal{X}^{[\alpha,\beta]}\Phi)$ and the Until formula $(\Phi_1\mathcal{U}^{[\alpha,\beta]}\Phi_2)$ of CSL, respectively. The assertion on formula sizes is straightforward. $\diamondsuit$

Proposition 4.2 *There is a formula of CSL^{TA} for which there is no equivalent CSL formula.*

The proof of Proposition 4.2 follows a scheme that is different from proofs of similar results on expressiveness of temporal logics for transition systems. We first define the left-hand delimiter $\langle$ for intervals, where $\langle$ denotes either $[$ or $($. Similarly, the right-hand delimiter $\rangle$ denotes either $]$ or $)$. Consider the family of ASMCs $\mathcal{M}[\mu,\mu']$ of Figure 6 (left), for $0 < \mu, \mu' < 1$. Let Φ be a formula of CSL or CSL$^{\mathrm{TA}}$ (for

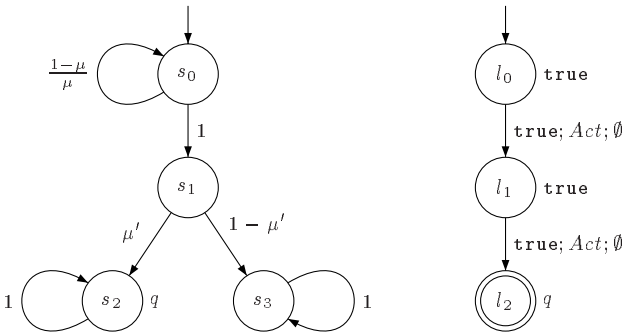

Figure 6. A family of Markov chains $\mathcal{M}[\mu,\mu']$ and a DTA $\mathcal{A}$ of CSL$^{\text{TA}}$

simplicity, we write $\models$ to denote the satisfaction relation of both CSL and CSL$^{\text{TA}}$). Then $[\Phi](s) = \{(\mu,\mu') \in (0,1)^2 \mid \mathcal{M}[\mu,\mu'], s \models \Phi\}$. For any $0 < \zeta < 1$, let $\Phi^{\zeta} = \mathcal{P}_{\geq\zeta}(\mathcal{A})$, where $\mathcal{A}$ is the DTA depicted in Figure 6 (right). It follows that $[\Phi^{\zeta}](s_0) = \{(\mu,\mu') \in (0,1)^2 \mid \mu \cdot \mu' \geq \zeta\}$.

Lemma 4.3 *Let Φ be a formula of CSL. Then:*

1. *$\forall i \in \{2,3\}, [\Phi](s_i)$ is either $(0,1)^2$ or $\emptyset$;*
2. *$[\Phi](s_1)$ is a finite union of rectangles of the form $(0,1) \times \langle \mathsf{a}, \mathsf{b} \rangle$;*
3. *$[\Phi](s_0)$ is a finite union of (open, closed, or mixed) rectangles of $(0,1)^2$.*

Proof. *Assertion (1).* When starting from s_2 or s_3, the satisfaction of Φ does not depend on p or μ'. Therefore assertion (1) is satisfied trivially.

Assertion (2). We prove assertion (2) by induction on the size of the formula. Let Φ be a formula of CSL. If Φ is an atomic proposition, then $[\Phi](s_1)$ is either $(0,1)^2$ or $\emptyset$. If $\Phi = \neg\Phi'$, then $[\Phi](s_1) = (0,1)^2 \setminus [\Phi'](s_1)$, and thus $[\Phi](s_1)$ is a finite union of rectangles of the form $(0,1) \times \langle \mathsf{a}, \mathsf{b} \rangle$. If $\Phi = \Phi' \wedge \Phi''$, then $[\Phi](s_1) = [\Phi'](s_1) \cap [\Phi''](s_1)$, and thus $[\Phi](s_1)$ is a finite union of rectangles of the form $(0,1) \times \langle \mathsf{a}, \mathsf{b} \rangle$.

Consider the case in which $\Phi = \mathcal{S}_{\geq\lambda}(\Phi')$. The steady-state distribution π of $\mathcal{M}[\mu,\mu']$ starting from s_1 is such that $\pi(s_1,s_2) = \mu'$ and $\pi(s_1,s_3) = 1-\mu'$. Now we distinguish different cases depending on whether $s_2 \models \Phi'$ and $s_3 \models \Phi'$. If both states satisfy Φ', then $[\Phi](s_1) = (0,1)^2$; if neither satisfies Φ' then $[\Phi](s_1) = \emptyset$; if $s_2 \models \Phi'$ and $s_3 \not\models \Phi'$ then $[\Phi](s_1) = (0,1) \times [\lambda,1)$; if $s_2 \not\models \Phi'$ and $s_3 \models \Phi'$ then $[\Phi](s_1) = (0,1) \times (0,1-\lambda]$. The cases of $\Phi = \mathcal{S}_{\sim\lambda}(\Phi')$, with $\sim \in \{\leq, <, >\}$, follow similarly.

If $\Phi = \mathcal{P}_{\geq\lambda}(\mathcal{X}^{[\alpha,\beta]}\Phi')$ we distinguish different cases depending on whether $s_2 \models \Phi'$ and $s_3 \models \Phi'$. All the cases are handled similarly, and we only consider that in which $s_2 \models \Phi'$ and $s_3 \not\models \Phi'$: Then $[\Phi_1](s_1) = \{(\mu,\mu') \in (0,1)^2 \mid (e^{-\alpha} - e^{-\beta}) \cdot \mu' \geq \lambda\} = (0,1) \times [\frac{\lambda}{e^{-\alpha}-e^{-\beta}}, 1)$, which is of the required form. The cases of $\Phi = \mathcal{P}_{\sim\lambda}(\mathcal{X}^{[\alpha,\beta]}\Phi')$, with $\sim \in \{\leq, <, >\}$, follow similarly.

If $\Phi = \mathcal{P}_{\geq\lambda}(\Phi' \mathcal{U}^{[\alpha,\beta]} \Phi'')$, we make a case analysis w.r.t. to the rectangles where the satisfaction of Φ' and Φ'' by s_1 is invariant (that is, we consider rectangles of the partition of $(0,1)^2$ induced by the rectangles of $[\Phi'](s_1)$ and $[\Phi''](s_1)$). Our aim is to obtain $[\Phi](s_1)$ by replacing each such rectangle with a set of rectangles in which Φ is satisfied.

Given such a rectangle $\mathcal{R} \subseteq (0,1)^2$ for which $\mathcal{M}[\mu,\mu'], s_1 \models \Phi''$ for all $(\mu,\mu') \in \mathcal{R}$, then $\mathcal{M}[\mu,\mu'], s_1 \models \Phi$ for all $(\mu,\mu') \in \mathcal{R}$. Hence $\mathcal{R}$ is included in $[\Phi](s_1)$. Conversely, if $\mathcal{M}[\mu,\mu'], s_1 \models \neg\Phi' \wedge \neg\Phi''$ for all $(\mu,\mu') \in \mathcal{R}$, then $\mathcal{M}[\mu,\mu'], s_1 \not\models \Phi$ for all $(\mu,\mu') \in \mathcal{R}$. Hence no rectangle contained in $\mathcal{R}$ is included in $[\Phi](s_1)$.

Now consider a rectangle $\mathcal{R}$ for which, for all $(\mu,\mu') \in \mathcal{R}$, we have $\mathcal{M}[\mu,\mu'], s_1 \models \Phi' \wedge \neg\Phi''$. Assume that $\mathcal{M}[\mu,\mu'], s_2 \models \Phi''$ and $\mathcal{M}[\mu,\mu'], s_3 \not\models \Phi''$ (the other cases are handled similarly). Then we obtain $\{(\mu,\mu') \in \mathcal{R} \mid \mathcal{M}[\mu,\mu'], s_1 \models \Phi\} = \{(\mu,\mu') \in \mathcal{R} \mid (1-e^{-\beta}) \cdot \mu' + e^{-\beta} \geq \lambda\} = \{(\mu,\mu') \in \mathcal{R} \mid \mu' \geq \frac{\lambda - e^{-\beta}}{1-e^{-\beta}}\}$. Thus we include in $[\Phi](s_1)$ the rectangle $\mathcal{R} \cap ((0,1) \times [\frac{\lambda-e^{-\beta}}{1-e^{-\beta}}, 1))$.

The cases of $\Phi = \mathcal{P}_{\sim\lambda}(\Phi' \mathcal{U}^{[\alpha,\beta]} \Phi'')$, $\sim \in \{\leq, <, >\}$, follow similarly.

Assertion (3). We now prove assertion (3) by induction on the size of the formulas. Let Φ be a formula of CSL. The cases in which Φ is an atomic proposition, $\Phi = \neg\Phi'$, $\Phi = \Phi' \wedge \Phi''$ and $\Phi = \mathcal{S}_{\sim\lambda}(\Phi')$ are proved exactly as for assertion (2) (the steady-state distribution of $\mathcal{M}[\mu,\mu']$ starting from s_0 is the same as that starting from s_1).

If $\Phi = \mathcal{P}_{\geq\lambda}(\mathcal{X}^{[\alpha,\beta]}\Phi')$, we make a case analysis w.r.t. to the rectangles in which the satisfaction of Φ' by s_0 and s_1 is invariant (that is, we consider rectangles of the partition of $(0,1)^2$ induced by the rectangles of $[\Phi'](s_0)$ and $[\Phi'](s_1)$). As above, we obtain $[\Phi](s_0)$ by replacing each such rectangle with a set of rectangles in which Φ is satisfied. We only consider one such case (the other cases are handled similarly). Consider a rectangle $\mathcal{R} \subseteq (0,1)^2$ for which, for all $(\mu,\mu') \in \mathcal{R}$, we have $\mathcal{M}[\mu,\mu'], s_0 \models \Phi'$ and $\mathcal{M}[\mu,\mu'], s_1 \not\models \Phi'$ Then $\{(\mu,\mu') \in \mathcal{R} \mid \mathcal{M}[\mu,\mu'], s_0 \models \Phi\} = \{(\mu,\mu') \in \mathcal{R} \mid (e^{-\alpha/\mu} - e^{-\beta/\mu}) \cdot \mu \geq \lambda\}$. Let $f(\mu) = (e^{-\alpha/\mu} - e^{-\beta/\mu}) \cdot \mu$. Note that the derivative of f changes its sign only a finite number of times inside $(0,1)$ (in fact in $\mathbb{R}$). Therefore $(0,1)$ may be decomposed into a finite number of consecutive intervals where inside an interval f is monotonic. As a consequence $(0,1)$ may be partitioned into a finite number of consecutive intervals (different from the previous ones) where alternatively f is greater or equal than λ or strictly smaller than λ. The intervals for which f is greater than or equal to λ induce a finite number of rectangles of the form $\langle \mathsf{a}, \mathsf{b} \rangle \times (0,1)$, which are included $[\Phi](s_0)$. The cases of $\Phi = \mathcal{P}_{\sim\lambda}(\mathcal{X}^{[\alpha,\beta]}\Phi')$, with $\sim \in \{\leq, <, >\}$, follow similarly.

If $\Phi = \mathcal{P}_{\geq\lambda}(\Phi' \mathcal{U}^{[\alpha,\beta]} \Phi'')$, we make a case analysis w.r.t. to the rectangles where the satisfaction of Φ' and Φ'' by s_0 and by s_1 is invariant (that is, we consider rectangles of the partition of $(0,1)^2$ induced by the rectangles of $[\Phi'](s_0)$, $[\Phi'](s_1)$, $[\Phi''](s_0)$ and $[\Phi''](s_1)$). Again, we obtain $[\Phi](s_0)$ by replacing each such rectangle with a set of rectangles in which Φ is satisfied.

We handle only one case, noting that the other cases are handled similarly. Consider the rectangle $\mathcal{R} \subseteq (0,1)^2$ such that, for $i \in \{0,1\}$, we have $\mathcal{M}[\mu,\mu'], s_i \models \Phi' \wedge \neg\Phi''$ $\mathcal{M}[\mu,\mu'], s_2 \models \neg\Phi' \wedge \Phi''$ and $\mathcal{M}[\mu,\mu'], s_3 \models \neg\Phi' \wedge \neg\Phi''$. The key observation here is that, inside any such rectangle, the loop around s_0 is irrelevant due to the nature of the Until operator $\mathcal{U}$. Then we have $\{(\mu,\mu') \in \mathcal{R} \mid \mathcal{M}[\mu,\mu'], s_0 \models \Phi\} = \{(\mu,\mu') \in \mathcal{R} \mid (e^{-2\alpha}(1+2\alpha) - e^{-2\beta}(1+2\beta)) \cdot \mu' \geq \lambda\} = \{(\mu,\mu') \in \mathcal{R} \mid \mu' \geq \frac{\lambda}{e^{-2\alpha}(1+2\alpha) - e^{-2\beta}(1+2\beta)}\}$ (the first formula has been obtained by applying an Erlang distribution). Then we include the rectangle $\mathcal{R} \cap ((0,1) \times [\frac{\lambda}{e^{-2\alpha}(1+2\alpha) - e^{-2\beta}(1+2\beta)}, 1))$ in $[\Phi](s_0)$.

The cases of $\Phi = \mathcal{P}_{\sim\lambda}(\Phi' \mathcal{U}^{[\alpha,\beta]} \Phi'')$, $\sim \in \{\leq, <, >\}$, follow similarly. $\diamondsuit$

Because $[\Phi^{\zeta}](s_0) = \{(\mu,\mu') \mid p \cdot \mu' \geq \zeta\}$ cannot be expressed as a finite union of rectangles, Lemma 4.3 establishes that Φ^{ζ} is not equivalent to any formula of CSL. Lemma 4.4 then gives a direct proof of Proposition 4.2. Observe also that the proof can be adapted easily to build a formula of asCSL not equivalent to any formula of CSL.

Lemma 4.4 *For each $0 < \zeta < 1$, the CSL^{TA} formula Φ^{ζ} is not equivalent to any formula of CSL.*

Proposition 4.5 states that CSL$^{\text{TA}}$ is as least as expressive as asCSL. The proof of the proposition is omitted for reasons of space. We also conjecture that there exists a CSL$^{\text{TA}}$ formula for which no equivalent asCSL formula exists.

Proposition 4.5 *For any formula Φ of asCSL there is a formula Φ' of CSL^{TA} equivalent to Φ.*

5 Conclusion

In this paper we have defined a new stochastic temporal logic CSL$^{\text{TA}}$, based on timed automata, which we propose as a good trade-off between adding flexibility to property specification and limiting the explosion of complexity in analysis. With regard to the specification of properties, the most significant extension is the possibility of specifying an arbitrary number of timing constraints along an execution path which may also depend on the history of the process. More precisely, CSL$^{\text{TA}}$ subsumes both CSL and asCSL. Furthermore, the evaluation process is handled in an uniform way *via* Markov regenerative processes rather than by *ad hoc* transformations as previously.

Further work can consider an implementation of the proposed method (possibly exploiting existing DSPN tools), and the extension of CSL$^{\text{TA}}$ to allow for rewards [12]. We also plan to compare the DTA approach with the automata-based approach of Obal and Sanders [14].

References

[1] M. Ajmone Marsan and G. Chiola. On Petri nets with deterministic and exponentially distributed firing times. In *Proc. ICATPN'86*, volume 266 of *LNCS*, pages 132–145. Springer, 1986.

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

[3] A. Aziz, K. Sanwal, V. Singhal, and R. Brayton. Model-checking continuous time Markov chains. *ACM Transactions on Computational Logic*, 1(1):162–170, 2000.

[4] C. Baier, L. Cloth, B. Haverkort, M. Kuntz, and M. Siegle. Model checking action- and state-labelled Markov chains. In *Proc. DSN'04*, pages 701–710. IEEE, 2004.

[5] C. Baier, B. Haverkort, H. Hermanns, and J.-P. Katoen. Model-checking algorithms for continuous-time Markov chains. *IEEE Transactions on Software Engineering*, 29(6):524–541, 2003.

[6] G. Behrmann, A. David, K. G. Larsen, J. Håkansson, P. Pettersson, W. Yi, and M. Hendriks. UPPAAL 4.0. In *Proc. QEST'06*, pages 125–126. IEEE, 2006.

[7] A. Bouajjani, Y. Lakhnech, and S. Yovine. Model-checking for extended timed temporal logics. In *Proc. FTRTFT'96*, volume 1134 of *LNCS*, pages 306–325. Springer, 1996.

[8] G. Clark and J. Hillston. Towards automatic derivation of performance measures from PEPA models. In *Proceedings of the UK Performance Engineering Workshop*, 1996.

[9] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, 1999.

[10] E. A. Emerson and J. Y. Halpern. "Sometimes" and "not never" revisited: On branching versus linear time temporal logic. *Journal of the ACM*, 33(1):151–178, 1986.

[11] R. German. *Performance Analysis of Communication Systems with Non-Markovian Stochastic Petri Nets*. Wiley, 2000.

[12] B. Haverkort, L. Cloth, H. Hermanns, J.-P. Katoen, and C. Baier. Model checking performability properties. In *Proc. DSN'02*, pages 103–112. IEEE, 2002.

[13] F. Laroussinie, N. Markey, and P. Schnoebelen. Model checking timed automata with one or two clocks. In *Proc. CONCUR'04*, volume 3170 of *LNCS*, pages 387–401. Springer, 2004.

[14] W. D. Obal II and W. H. Sanders. State-space support for path-based reward variables. *Performance Evaluation*, 35(3-4):233–251, 1999.

[15] J. Ouaknine and J. Worrell. On the language inclusion problem for timed automata: Closing a decidability gap. In *Proc. LICS'04*, pages 54–63. IEEE, 2004.

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