AVACS – Automatic Verification and Analysis of Complex Systems

REPORTS

of SFB/TR 14 AVACS

Dynamic Communicating Probabilistic Timed Automata Playing Games

by

Rüdiger Ehlers E. Moritz Hahn Martin Mehlmann

Hans-Jörg Peter Jan Rakow Tobe Toben Bernd Westphal

AVACS Technical Report No. 75

July 2011

ISSN: 1860-9821

Publisher: Sonderforschungsbereich/Transregio 14 AVACS
(Automatic Verification and Analysis of Complex Systems)
Editors: Bernd Becker, Werner Damm, Bernd Finkbeiner, Martin Fränzle,
Ernst-Rüdiger Olderog, Andreas Podelski
ATRs (AVACS Technical Reports) are freely downloadable from `www.avacs.org`

Author(s) contact: Bernd Westphal (`westphal@informatik.uni-freiburg.de`).

Dynamic Communicating Probabilistic Timed Automata Playing Games

Rüdiger Ehlers[3], E. Moritz Hahn[3], Martin Mehlmann[4], Hans-Jörg Peter[3], Jan Rakow[1], Tobe Toben[2], and Bernd Westphal[4]

[1] Carl von Ossietzky University Oldenburg, Germany
[2] OFFIS e.V., Oldenburg, Germany
[3] Saarland University Saarbrücken, Germany
[4] Albert-Ludwigs-University Freiburg, Germany

Abstract. Systems of Systems (SoS) comprising a varying number of communicating processes (or agents) are getting ever more important. As of yet, formal modeling languages and specification logics address isolated features of SoS only. We propose the concise modeling language DCS^{++} and the property specification logic DPTATL that address all relevant SoS aspects in a unified game-theoretic framework. Language and logic turn out to be an orthogonal extension of well-known modeling formalisms and logics. Both modeling and specification languages are demonstrated on a non-trivial network routing example.

1 Introduction

The growing interest in System of Systems (SoS), that is, a collection of concurrently interacting sub-systems, raises the need for new integrated modeling and specification frameworks. In particular safety-critical applications of SoS require a rigid understanding of the SoS behavior. As of yet, many isolated characteristics of SoS can be addressed by different formal languages. In particular, Timed Automata (TA) [2] and Markov Decision Processes (MDP) [22] have been proposed for (monolithic) modeling of *real-time* and *probabilistic* aspects, respectively. Also, there are different (modular) frameworks to specify *communication* aspects, e.g. Communicating Finite State Machines (CFSM) [11], which have already been extended to cope with *dynamic process creation* and *dynamically changing communication topologies*, e.g. in Dynamic Communication Systems (DCS) [8]. Similarly, Probabilistic Timed Automata (PTA) [18] pose an extension of TA and MDP. There is, however, no unified framework that addresses all the dimensions of communication-, real-time-, probabilistic- and dynamic behavior in a non-monolithic, but modular style of dynamic communicating automata. A further dimension, games, allows to distinguish between adversarial and collaborative behavior. We propose a framework that is expressive enough to deal with realizability properties such as *"does team T of processes have a strategy to realize a certain objective within time t with a probability of at least p against all other processes"*. The underlying formal language DCS^{++} is based on a new class of automata, namely *dynamic communicating probabilistic timed automata* (DCPTA). DCPTA are instantiated to processes owning a unique *identity* from a (possibly unbounded) set of identities Id. Processes store identities in link-typed variables and may communicate them by multi-cast synchronous message passing, leading to a dynamically changing communication topology.

The semantics of DCS^{++} is defined in terms of a *dynamic probabilistic timed game structure* (DPTGS), a combination of probabilistic timed structures (PTS) [19] and timed game structures (TGS) [4]. To address the dynamics in communication, we use a state $\mathsf{s} \in \mathsf{S}$ in such structures to actually represent a graph that characterizes the current communication topology. This yields a probabilistic timed game model, where $\Gamma : \mathsf{S} \to \{(p, d, a) \in \mathsf{Id} \times \mathbb{R}_{\geq 0} \times \mathsf{Act}\}$ assigns available *moves* to a state, stating which process p after delay d may apply action a. The effect of a move $(p, d, a) \in \mathsf{M}$ is defined in a probabilistic transition relation $\delta : \mathsf{S} \times \mathsf{M} \to \mu(\mathsf{S})$, which determines the probability to take a transition from a state s to state s' under a move m. We obtain an open system semantics, where, roughly speaking, a team of processes with all its descendants plays against all other processes and their descendants.

DCS^{++} is complemented by the specification logic DPTATL that expresses SoS properties regarding time, probability, and strategies, and which has a sound formal interpretation. DPTATL provides logical agents variables, including first-order quantification and the ability to refer to the birth, state and communication topology of dedicated processes. The property above, intuitively, requires the existence of a team strategy that, against all possible strategies for the adverse team, yields a set of *strategy outcomes* (essentially paths in the DPTGS) that still meets the probability and time bound.

Note that the concept of *identities* in DPTGS smoothly integrates into both the property specification and the open system semantics – in DPTATL a team is made up of processes that have a direct counterpart in the formal model description of DCPTA.

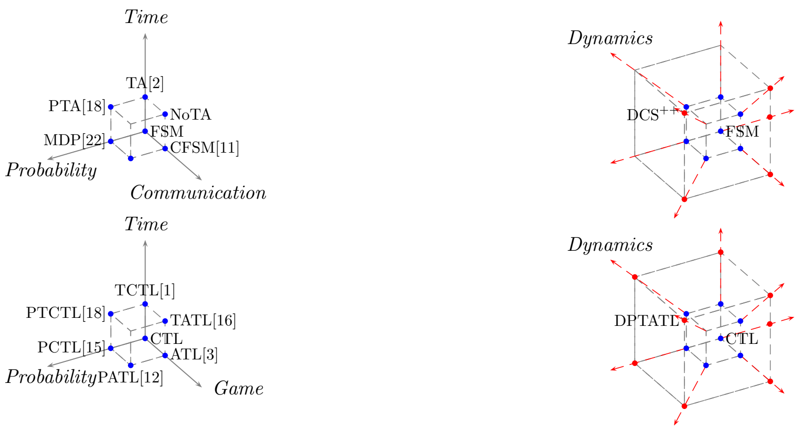

Fig. 1. DCS^{++} and DPTATL situated.

Related Work TA [2] extend classical finite state machines (FSM) by clock variables and thereby allow constraining the times at which transitions occur. Probabilistic aspects are typically expressed by variants of MDP [22], where edges are labeled with actions and the behavior is a non-deterministic choice among probabilistic edges (i.e. the successor state is determined by a probabilistic choice). Run-time creation of processes can be modeled in the π-calculus [21], automata-based frameworks like Dynamic Input/Output Automata [5] or DCSs [8], or in Graph Transformation Systems [23]. These formalisms typically also allow for *communication* among the set of active processes. CFSM [11] focus on FIFO-buffered communication within a statically connected set of FSMs. PTA [18] are an extension of timed automata with discrete probability distributions. Games on PTAs have been proposed in [20]. Networks of Timed Automata (NoTA) are considered by the UPPAAL Model-Checker [9]. The Promela language of SPIN has been extended with dynamics and probabilism [7]. Modest [10] combines time and probabilism in a compositional framework, and also allows for a limited form of dynamic process creation. In concurrent game structures [3] transitions correspond to moves which are controlled by dedicated players. This formalism allows expressing open systems that interact in a (hostile) environment.

Corresponding extensions related to time, probabilism, games and dynamics have also been proposed for specification logics, typically as variants of the temporal logics CTL and LTL. In particular, there are Timed CTL [1] and Probabilistic Timed CTL [19], Alternating-time Temporal Logic (ATL) [3] as a generalization of CTL with path quantification based on the game semantics, timed and probabilistic variants of ATL (TATL [16], PATL [12]), a stochastic game logic [6], and first-order extensions of temporal logics like Evolution Logic [26], BOTL [14] and METT [8] for addressing dynamics in the set of processes.

To the best of our knowledge, there is no modeling language and no specification logic that addresses all considered aspects in a unified framework. Fig. 1 illustrates the orthogonality in DCS^{++} and DPTATL.

Preliminaries. By $\mu(Q) \subset [Q \to [0,1]]$ we denote the set of all finite discrete *probability distributions* over a set Q, i.e. for each $P \in \mu(Q)$, $\sum_{q \in Q} P(q) = 1$ and $\{q \in Q \mid P(q) > 0\}$ is finite. A *probability space* is a tuple $(\Omega, \mathcal{F}, \mathbf{P})$, where Ω is the *sample space*, $\mathcal{F} \subseteq 2^{\Omega}$ is a σ-*algebra* on Ω (i.e., a set containing Ω and closed under complement and denumerable union), and $\mathbf{P}$ is a *probability measure* on $\mathcal{F}$ (i.e. a function $\mathbf{P} : \mathcal{F} \to [0,1]$ such that $\mathbf{P}(\Omega) = 1$ and $\mathbf{P}(\biguplus_{i \geq 0} B_i) = \sum_{i \geq 0} \mathbf{P}(B_i)$, where $\{B_i\}_{i \geq 0}$ is a disjoint denumerable family of sets in $\mathcal{F}$). The pair $(\Omega, \mathcal{F})$ is called *measurable space*. A *Borel measurable space* is the smallest measurable space containing all open sets of a topology. We denote by $\mathcal{B}(\Omega)$ the Borel σ-algebra on a sample space Ω. Let $\mathbf{Prob}(\Omega)$ denote the set of all probability measures on $\mathcal{B}(\Omega)$, and $\mathbf{supp}(\mathbf{P}) = \{X \in dom(\mathbf{P}) \mid \mathbf{P}(X) > 0\}$.

2 Dynamic Communicating Probabilistic Timed Automata

In this section, we introduce DCPTA, an orthogonal composition of TA and MDP extended by communication over links and dynamic process creation. As semantic model we will introduce DPTGS which combines PTS and TGS with the concept of dynamic topologies.

2.1 DCPTA Syntax

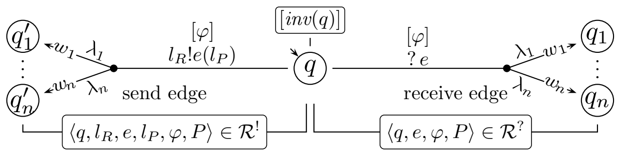

Fig. 2. DCPTA Syntax

A DCPTA is basically given as an automaton description, hence comprising a set of locations and (labelled) hyper edges. Additionally, a DCPTA has two sets of variables, namely *clock-* and *link-typed* ones, denoted by $\mathcal{X}$ and $\mathcal{L}$, respectively. While a clock is real-valued, a link has the domain 2^{Id} ranging over a set of identities Id. Sending a message over a link means sending the message to each identity stored in the link variable. Also, a link can be attached to a message as parameter such that the content of a link can be passed to other processes. By this, dynamic communication topologies can be established. We introduce the basic syntactic concepts first and then define DCPTA below.

Constraints We inductively define the set $\mathsf{Constr}(\mathcal{V})$ of *constraints* over variables $\mathcal{V} = \mathcal{X} \,\dot\cup\, \mathcal{L}$ by

$$\psi ::= \neg\psi_1 \mid \psi_1 \wedge \psi_2 \mid \zeta_1 < \zeta_2 \mid \zeta_1 = \zeta_2 \mid l \subset l' \mid l = l',$$

where $l, l' \in \mathcal{L}$ and ζ_1, ζ_2 is either a variable $c \in \mathcal{X}$ or a constant (rational number).

Messages The set of all *messages* is denoted by *Msgs*, and each message $e \in Msgs$ has a set of predefined formal parameters $Prms = \{\mathsf{snd}, \mathsf{param}\} \subseteq \mathcal{L}$.

Update Assignments We denote the set of type-consistent *basic update assignments* over a set of variables $\mathcal{V}$ by $\mathsf{Assign}_0(\mathcal{V})$. Each basic update assignment has a notion of a *left-hand side variable.* $\mathsf{Assign}_0(\mathcal{V})$ is inductively defined by

$$\tilde{\lambda} ::= c := 0 \mid l := l' \cup l'' \mid l := l' \cap l'' \mid l := l' \setminus l''$$

where $c \in \mathcal{X}$ is a clock, and $l, l', l'' \in \mathcal{L}$ are links. The set of *update assignments* $\mathsf{Assign}(\mathcal{V})$ induced by $\mathsf{Assign}_0(\mathcal{V})$ comprises the consistent terms of the form

$$\lambda ::= \mathsf{skip} \mid \tilde{\lambda} \mid l := \mathsf{create(B)} \mid \lambda_1; \lambda_2,$$

where $\tilde{\lambda} \in \mathsf{Assign}_0(\mathcal{V})$, $l \in \mathcal{L}$ is a link, and B is a DCPTA. Such a term is called *consistent* if and only if each variable from $\mathcal{V}$ occurs at most once as left-hand side variable.

DCPTA Definition Figure 2 gives an overview of DCPTA syntax by means of two (symbolic) edges. Formally, a DCPTA is a tuple

$$\mathsf{B} = (Q, q_0, \mathcal{V}, inv, \mathcal{M}, \mathcal{R}^!, \mathcal{R}^?), \text{ where}$$

- Q is a finite set of locations with *initial* location $q_0 \in Q$,
- $\mathcal{V} \subseteq \mathcal{X} \dot{\cup} \mathcal{L}$ is a finite set of variables with $\mathsf{self} \in \mathcal{L}$ and $\{\mathsf{snd}, \mathsf{param}\} \cap \mathcal{V} = \emptyset$,
- $inv : Q \to \mathsf{Constr}(\mathcal{X})$ assigns each location a constraint over clock variables,
- $\mathcal{M} \subseteq Msgs$ is a finite set of messages.
- $\mathcal{R}^! \subseteq Q \times \mathcal{L} \times \mathcal{M} \times \mathcal{L} \times \mathsf{Constr}(\mathcal{V}) \times \mu(\mathsf{Assign}(\mathcal{V}) \times Q)$ is a set of *send* edges,
- $\mathcal{R}^? \subseteq Q \times \mathcal{M} \times \mathsf{Constr}(\mathcal{V} \cup Prms) \times \mu(\mathsf{Assign}(\mathcal{V} \cup Prms) \times Q)$ is a set of *receive* edges.

For edges $\langle q, l_R, e, l_P, \varphi, P\rangle \in \mathcal{R}^!$ and $\langle q, e, \varphi, P\rangle \in \mathcal{R}^?$, we have that in both cases, q is the *source location*, e is the *synchronization message*, φ is the *guard*, and P is a discrete probability distribution on a finite set of pairs $\langle \lambda, q\rangle$. A pair $\langle \lambda, q\rangle$ comprising an update assignment λ and *target location* q is also called *update*. In a send edge $\langle q, l_R, e, l_P, \varphi, P\rangle \in \mathcal{R}^!$, additionally, l_R symbolically denotes the *receivers* and l_P is the link *parameter*, storing identities to be communicated. Link self allows accessing the own identity. The formal parameters snd and param allow accessing the identity of the sender and the messages parameter (c.f. Sect.4), respectively. An *internal* transition is modeled by a send edge with self as receiver.

Well-formedness Updates $\langle \lambda, q\rangle$ may use variables in $\mathcal{V} \setminus \{\mathsf{self}\}$ as left-hand side variables only. Thus assignments to parameters in *Prms* are excluded. Message reception in a DCPTA has to be *input deterministic*, formally $\forall\, t_1 = \langle q_1, e_1, \varphi_1, P_1\rangle, t_2 = \langle q_2, e_2, \varphi_2, P_2\rangle \in \mathcal{R}^? : t_1 \neq t_2 \;\wedge\; q_1 = q_2 \;\wedge\; e_1 = e_2 \implies \varphi_1 \wedge \varphi_2 \equiv \mathit{false}$. Non-determinism is only allowed among send edges. By this, we will be able to determine a unique probability distribution on the state space by considering the distribution of the send edge fired and the involved receive edges distributions (if any). Moreover, given that P is induced by the probabilities $w_1, \ldots, w_n \in \mathbb{R}_{\geq 0}$, we require $w_{1 \leq i \leq n} \geq 0 \wedge \sum_{i=1}^{n} w_i = 1$.

For all updates $\langle \lambda, q\rangle$ not explicitly denoted (or drawn), we implicitly have $P(\langle \lambda, q\rangle) = 0$. A send edge is blocked unless all named receivers are ready. Hence, we assume that DCPTAs are *input enabled*, i.e. each location has at least one receive edge per message. The guard of an edge can be omitted if φ is *true*.

	l_R	φ	$\mathbf{supp}(P)$	inv	λ
FSM	$\equiv \mathsf{self}$	$\equiv true$	$= 1$	$dom(inv) = \emptyset$	$\emptyset$
TA	$\equiv \mathsf{self}$	$\in \mathsf{Constr}(\mathcal{X})$	$= 1$	$\in \mathsf{Constr}(\mathcal{X})$	$\mathsf{Assign}(\mathcal{X})$
MDP	$\equiv \mathsf{self}$	$\equiv true$	≥ 1	$dom(inv) = \emptyset$	$\emptyset$
PTA	$\equiv \mathsf{self}$	$\in \mathsf{Constr}(\mathcal{X})$	≥ 1	$\in \mathsf{Constr}(\mathcal{X})$	$\mathsf{Assign}(\mathcal{X})$
DCS	$\in \mathcal{L}$	$\in \mathsf{Constr}(\mathcal{L})$	$= 1$	$dom(inv) = \emptyset$	$\mathsf{Assign}(\mathcal{L})$
CFSM	$\in \mathcal{L}$	$\in \mathsf{Constr}(\mathcal{L})$	$= 1$	$dom(inv) = \emptyset$	$\mathsf{Assign}(\mathcal{L})$

Fig. 3. DCPTA Subclasses

Orthogonality DCPTA encompass FSM, TA, MDP and PTA as illustrated in Fig. 3. The monolithic formalisms of FSM, TA, MDP and PTA are obtained by restricting the receivers l_R to self only. While timed variants arise by restricting all guards φ to expressions over clock variables and the assignments to clock-resets only, untimed ones are derived if guard φ, invariants $inv(q)$ and assignments are entirely absent. In contrast to non-probabilistic formalisms TA and FSM, probabilistic branching is still present in MDP and PTA, so that edges may define non trivial distributions P with $|\mathbf{supp}(P)| > 1$. The modular formalisms CFSMs and DCSs, are derived by exclusion of clocks and non-trivial distributions. Actually, CFSMs describe a *static* setting of anonymous communicating automata, where create statements only occur in an initialization phase.

DCS^{++} A DCPTA serves as a behavioral template that is dynamically instantiated to processes. A set of DCPTAs $\mathsf{D} = (\{\mathsf{B}_1, \ldots, \mathsf{B}_n\}, \mathsf{B}^0)$ is called a DCS^{++} model.

While the initial DCPTA $\mathsf{B}^0 \in \{\mathsf{B}_1, \ldots, \mathsf{B}_n\}$ is implicitly instantiated from the start, all other processes are created dynamically by some process executing a create statement.

2.2 Dynamic Probabilistic Timed Game Structure

Dynamic Probabilistic Timed Game Structures (DPTGS) will serve as the *semantic domain* for DCS^{++} models and thereby unify the semantic models of TA, MDP, PTA and TGA with the concept of identities from DCSs.

DPTGS Definition A *DPTGS* over identities Id is given as a tuple $(\mathsf{S}, \Gamma, \mathsf{Act}, \delta)$, where

- S is a set of states on which a notion of domain is defined in form of a function $dom : \mathsf{S} \to 2^{\mathsf{Id}}$ indicating a set of processes *active* in a state.
- $\mathsf{M} \subseteq \mathsf{Id} \times \mathbb{R}_{\geq 0} \times \mathsf{Act}$ is a set of *moves*. We assume Act to comprise an idle action $\ominus \in \mathsf{Act}$. A *move* is a tuple $(p, d, a) \in \mathsf{M}$ of an identity p, delay d and an action a.
- $\Gamma : \mathsf{S} \to 2^{\mathsf{M}}$ assigns each state $\mathsf{s} \in \mathsf{S}$ a set of available moves $\Gamma(\mathsf{s})$.
- $\delta : \mathsf{S} \times \mathsf{M} \to \mu(\mathsf{S})$ is a probabilistic transition relation

such that the following well-formedness constraints hold:

1. $|\mathbf{supp}(\delta(\mathsf{s}, (p, d, \ominus)))| = 1$ for $p \in \mathsf{Id}, \mathsf{s} \in \mathsf{S}, d \in \mathbb{R}_{\geq 0}$
2. For all $d, d' \in \mathbb{R}_{\geq 0}$ with $d' \leq d$ and all $a \in \mathsf{Act}$
   (a) $(p, d, a) \in \Gamma(\mathsf{s})$ iff $(p, d', \ominus) \in \Gamma(\mathsf{s})$ and $(p, d - d', a) \in \Gamma(\delta(\mathsf{s}, (p, d', \ominus)))$ (time additivity), and
   (b) if $\delta(\mathsf{s}, (p, d', \ominus)) = \mathsf{s}'$ and $\delta(\mathsf{s}', (p, d{-}d', a)) = P$, then $\delta(\mathsf{s}, (p, d, a)) = P$ (time determinism).
3. $(p, 0, \ominus) \in \Gamma(\mathsf{s})$ for all $p \in \mathsf{Id}$ and $\mathsf{s} \in \mathsf{S}$ (stutter move).
4. $\delta(\mathsf{s}, m)(\mathsf{s}') > 0$ implies $dom(\mathsf{s}) \subseteq dom(\mathsf{s}')$ (monotone frame).

The first three constraints are well-known from timed and probabilistic timed structures [27, 19, 4]. The monotone frame property requires active processes not to disappear. Disappearance can be modeled by restricting the set of available moves (i.e. by entering a sink location).

As in TGS, the transition relation is split into two functions assigning a set of available moves to a state (Γ) and the effect of a move (δ). In this semantic (game) model, intuitively, in a state s all processes $p \in dom(\mathsf{s})$ propose *simultaneously* and *independently* a move $m = (p, d, a) \in \Gamma(\mathsf{s})$. The move with the shortest delay prevails and is applied to s: After delay d action a is fired and with probability $\delta(\mathsf{s}, m)(\mathsf{s}')$ state s' is entered. In case of several moves of equal delay one move is selected non-deterministically. Formally, a *finite* set of moves $\kappa \subseteq \Gamma(\mathsf{s})$ is combined by the *joint destination function* $\delta_j : 2^{\mathsf{M}} \to 2^{\mathsf{M}}$:

$$\delta_j(\kappa) = \big\{(p, d, a) \in \kappa \mid d = \min\{d' \mid (p', d', a') \in \kappa\}\big\}.$$

Our logic DPTATL (see Sect. 3) will allow to require the existence of a *strategy* for a team $T \subseteq dom(\mathsf{s})$ of processes in a DPTGS that with probability p enforces an *outcome* achieving a certain objective against the rest of processes. For this, we need some technical definitions of paths, strategies, and outcomes.

Paths, Strategies and Outcomes. A *path* ω is a sequence $\omega = \mathsf{s}_0 \xrightarrow{m_0} \mathsf{s}_1 \xrightarrow{m_1} \ldots$ where $\mathsf{s}_i \in \mathsf{S}$, $m_i = (p_i, d_i, a_i) \in \Gamma(\mathsf{s}_i)$ and $\delta(\mathsf{s}_i, m_i)(\mathsf{s}_{i+1}) > 0$ for all $0 \leq i \leq |\omega|$. For a finite path ω, $|\omega| \in \mathbb{N}$ is the length of the path. $move(\omega^i)$ is the i-th move, and ω^i the i-th state of ω. By $\omega\uparrow^i$ we denote the finite prefix of ω up to ω^i. FinPth (InfPth) is the set of all finite (infinite) paths of a given structure and $\mathsf{FinPth}(\mathsf{s})$ ($\mathsf{InfPth}(\mathsf{s})$) is the set of all finite (infinite) paths starting at s. The *physical time* of a path $\omega \in \mathsf{InfPth}$ at position $k \in \mathbb{N}_0$ is defined by $\mathsf{time}(\omega, 0) = 0$ and $\mathsf{time}(\omega, k) = \mathsf{time}(\omega, k-1) + d$ iff $m_k = (\cdot, d, \cdot)$. The set of *time-divergent* paths $\mathsf{Timediv} \subseteq \mathsf{InfPth}$ is defined as $\mathsf{Timediv} = \{\omega \in \mathsf{InfPth} \mid \lim_{k\to\infty} \mathsf{time}(\omega, k) = \infty\}$.

By $\mathsf{Desc}(T, \omega) \subseteq \mathsf{Id}$ we denote the set of all processes created along a finite path ω by processes T or their descendants, i.e. $\mathsf{Desc}(T, \omega) = \mathsf{Desc}(T, \omega\uparrow^{|\omega|-1}) \cup \{p \in dom(\omega^{|\omega|}) \setminus dom(\omega^{|\omega|-1}) \mid move(\omega^{|\omega|}) \in \mathsf{Desc}(T, \omega\uparrow^{|\omega|-1}) \times \mathbb{R}_{\geq 0} \times \mathsf{Act}\}$ if $|\omega| > 1$ and $\mathsf{Desc}(T, \omega) = T$ if $|\omega| = 1$. Intuitively, by this definition those identities that emerge in any transition controlled by a team process (or a descendant) are cumulated.

A *joint strategy* for a team of processes $T \subseteq dom(\mathsf{s})$ at state s is a function

$$\pi_T : \{(p, \omega) \in \mathsf{Id} \times \mathsf{FinPth}(\mathsf{s}) \mid p \in \mathsf{Desc}(T, \omega)\} \to \mathsf{M},$$

such that $\pi_T(p, \omega) \in \{(p', d', a') \in \Gamma(last(\omega)) \mid p = p'\}$ for all $p \in \mathsf{Desc}(T, \omega)$ and $\omega \in \mathsf{FinPth}(\mathsf{s})$. That is, given a finite path ω, a joint strategy for team T processes assigns each team T process and any descendant along path ω a move m available in $last(\omega)$.

Let π_T be a joint strategy for team T starting at state s and $\pi_{\overline{T}}$ be a joint counter strategy to team T starting at state s, that is, let $\pi_{\overline{T}}$ be a joint strategy for team $\overline{T} = dom(\mathsf{s}) \setminus T$. Intuitively, $\mathsf{Outcomes}(\mathsf{s}, \pi_T, \pi_{\overline{T}}, \pi_s)$ comprises all the paths starting in s that may arise when team T processes (and their descendants) stick to π_T, team $\overline{T}$ processes (and their descendants) stick to $\pi_{\overline{T}}$, and remaining non-determinism due to move proposals with equal delay is resolved by a scheduler $\pi_s : \mathsf{FinPth}(\mathsf{s}) \times 2^{\mathsf{M}} \to \mathsf{M}$.

Formally, $\mathsf{Outcomes}(\mathsf{s}, \pi_T, \pi_{\overline{T}}, \pi_s)$ is the set of infinite paths $\mathsf{s}_0 \xrightarrow{m_1} \mathsf{s}_1 \xrightarrow{m_2} \ldots$ such that $\mathsf{s} = \mathsf{s}_0$ and for all $i \geq 1$ there is a set of moves $\kappa_i = \{\pi_T(p, \mathsf{s}_0 \xrightarrow{m_1} \ldots \mathsf{s}_{i-1}) \mid p \in T\} \cup \{\pi_{\overline{T}}(p, \mathsf{s}_0 \xrightarrow{m_1} \ldots \mathsf{s}_{i-1}) \mid p \in dom(\mathsf{s}_{i-1}) \setminus T\}$ such that

i) $m_i = \pi_s(\mathsf{s}_0 \xrightarrow{m_1} \ldots \mathsf{s}_{i-1}, \delta_j(\kappa_i))$ and
ii) $\delta(\mathsf{s}_{i-1}, m_i)(\mathsf{s}_i) > 0$.

Zeno Behaviour We will rule out non-meaningful strategies that prevent divergence of time (zeno behavior, cf. [4, 16]). For example, preventing a bad state by blocking time is not considered a reasonable strategy. We define the set of *blameless outcomes* for team T as follows. Let again π_T be a joint strategy for team T and $\pi_{\overline{T}}$ be a joint counter strategy to team T, both starting at state s. $\mathsf{BlamelessOutcomes}(\mathsf{s}, \pi_T, \pi_{\overline{T}})$ is the set of all non-divergent paths $\mathsf{s}_0 \xrightarrow{m_1} \mathsf{s}_1 \xrightarrow{m_2} \ldots$ such that $\mathsf{s} = \mathsf{s}_0$ and for all $i \geq 1$ there is a set of moves $\kappa_i = \{\pi_T(p, \mathsf{s}_0 \xrightarrow{m_1} \ldots \mathsf{s}_{i-1}) \mid p \in T\} \cup \{\pi_{\overline{T}}(p, \mathsf{s}_0 \xrightarrow{m_1} \ldots \mathsf{s}_{i-1}) \mid p \in dom(\mathsf{s}_{i-1}) \setminus T\}$ such that

i) $m_i \in \delta_j(\kappa_i)$ and
ii) $\delta(\mathsf{s}_{i-1}, m_i)(\mathsf{s}_i) > 0$ and
iii) there is a $k \in \mathbb{N}$ such that for all $l \geq k$: $\{(p, d, a) \in \delta_j(\kappa_l) \mid p \in T\} = \emptyset$.

Intuitively, $\mathsf{BlamelessOutcomes}(\mathsf{s}, \pi_T, \pi_{\overline{T}})$ comprises all the paths resulting from π_T and $\pi_{\overline{T}}$ where moves of team T may occur in a finite prefix only.

Path Measure We define the probability measure P on sets of paths as follows. For any DPTGS, joint strategy π_T, joint counter strategy $\pi_{\overline{T}}$, and scheduler π_s, all starting at state s, let $\mathsf{InfPth}^{\pi_T, \pi_{\overline{T}}, \pi_s}(\mathsf{s}) = \mathsf{Outcomes}(\mathsf{s}, \pi_T, \pi_{\overline{T}}, \pi_s)$ and $\mathsf{FinPth}^{\pi_T, \pi_{\overline{T}}, \pi_s}(\mathsf{s})$ the set of all finite path prefixes thereof. Let $\mathcal{F}_{\mathsf{s}}^{\pi_T, \pi_{\overline{T}}, \pi_s}$ be the smallest $\sigma-algebra$ on $\mathsf{InfPth}^{\pi_T, \pi_{\overline{T}}, \pi_s}(\mathsf{s})$ which contains the sets $\{\omega \mid \omega \in \mathsf{InfPth}^{\pi_T, \pi_{\overline{T}}, \pi_s}(\mathsf{s}) \wedge \omega\uparrow^{|\omega'|} = \omega'\}$ for all $\omega' \in \mathsf{FinPth}^{\pi_T, \pi_{\overline{T}}, \pi_s}(\mathsf{s})$.

For finite paths $\omega \in \mathsf{FinPth}^{\pi_T, \pi_{\overline{T}}, \pi_s}(\mathsf{s})$ with $\mathsf{s} = \omega^0$, $\mathsf{Prob} : \mathsf{FinPth}^{\pi_T, \pi_{\overline{T}}, \pi_s}(\mathsf{s}) \to [0, 1]$ is inductive defined by $\mathsf{Prob}(\omega) = 1$, iff $|\omega| = 1$ and $\mathsf{Prob}(\omega) = \mathsf{Prob}(\omega') \cdot \delta(last(\omega'), m)(\mathsf{s}')$, iff $\omega = \omega' \xrightarrow{m} \mathsf{s}'$. The measure P on $\mathcal{F}_{\mathsf{s}}^{\pi_T, \pi_{\overline{T}}, \pi_s}$ is now the unique measure such that $\mathsf{P}(\{\omega \mid \omega \in \mathsf{InfPth}^{\pi_T, \pi_{\overline{T}}, \pi_s}(\mathsf{s}) \wedge \omega\uparrow^{|\omega'|} = \omega'\}) = \mathsf{Prob}(\omega')$ (cylinder set construction, see e.g. [17]).

2.3 DCS^{++} Semantics

The semantics of a DCS^{++} model $\mathsf{D} = (\{\mathsf{B}_1, \ldots, \mathsf{B}_n\}, \mathsf{B}^0)$ will be given by a translation to a DPTGS. To this end, we define (1.) the set of *states* induced by D, (2.) the induced *actions* of D, (3.) the *available moves* of D per state, and (4.) the probabilistic *transition relation* when playing a move in a state.

(1.) States A state of the resulting DPTGS comprises the configurations of all active processes. A process configuration in turn is given by its location and the valuation of its variables. This will be defined in the following. A link (clock) valuation is a function $\sigma_{\mathcal{L}} : \mathcal{L} \to 2^{\mathsf{Id}}$ ($\sigma_{\mathcal{X}} : \mathcal{X} \to \mathbb{R}_{\geq 0}$). For convenience, we will also refer to (type consistent) combined valuations $\sigma : \mathcal{V} \to \mathbb{R}_{\geq 0} \,\dot{\cup}\, 2^{\mathsf{Id}}$. By $\Sigma(\mathcal{V})$ we denote the set of all combined valuations. We assume that expressions and boolean constraints ζ have a type-consistent *interpretation* $[\![\zeta]\!](\sigma)$, given a valuation $\sigma \in \Sigma(\mathcal{V})$, and write $\sigma \models \zeta$ when $[\![\zeta]\!](\sigma) =$ *true*. We denote the effect of an assignment λ on σ by $[\![\lambda]\!](\sigma)$. By σ_0 we refer to the initial valuation that assigns each variable $x \in \mathit{dom}(\sigma_0)$ its initial value (i.e. 0 or $\emptyset$). A *configuration* of a process described by DCPTA $\mathsf{B}_i = (Q_i, q_0^i, \mathcal{V}_i, \mathit{inv}, \mathcal{M}, \mathcal{R}_i^!, \mathcal{R}_i^?)$ is a tuple $(q, \sigma) \in Q_i \times \Sigma(\mathcal{V}_i)$ of the current location q and valuation σ of the variables. For DCS^{++} model $\mathsf{D} = (\{\mathsf{B}_1, \ldots, \mathsf{B}_n\}, \mathsf{B}^0)$, we set $Q_{\mathsf{D}} = \biguplus_{i=0}^n Q_i$, $\mathcal{V}_{\mathsf{D}} = \biguplus_{i=0}^n \mathcal{V}_i$, $\mathcal{R}_{\mathsf{D}}^! = \biguplus_{i=0}^n \mathcal{R}_i^!$ and $\mathcal{R}_{\mathsf{D}}^? = \biguplus_{i=0}^n \mathcal{R}_i^?$. A *state* of D is a total function $\mathfrak{s} : \mathsf{Id} \to \{\bot\} \,\dot{\cup}\, (Q_{\mathsf{D}} \times \Sigma(\mathcal{V}_{\mathsf{D}}))$ mapping each identity $p \in \mathsf{Id}$ either to $\bot$ or to a configuration. The set of all states of D is denoted by $\mathfrak{S}(\mathsf{D})$. An agent a is active in $\mathfrak{s}$ if $\mathfrak{s}(a) \neq \bot$, and by $\mathit{dom}(\mathfrak{s})$ we refer to the set of all active agents in $\mathfrak{s}$. By $\mathfrak{s}_{\mathsf{loc}}(p)$ and $\mathfrak{s}_{\mathsf{val}}(p)$ we refer to location q and valuation σ of process p in state $\mathfrak{s}$, respectively.

(2). Actions The actions are send edges and the idle action: $\mathsf{A}(\mathsf{D}) = \mathcal{R}_{\mathsf{D}}^! \,\dot{\cup}\, \{\ominus\}$.

(3). Available Moves Passage of time is represented by a simultaneous increase of the clock variables among all processes, hence clock increments are defined as $\mathfrak{s}+d$ with $(\mathfrak{s}+d)(p) = \bot$ iff $\mathfrak{s}(p) = \bot$ and $(\mathfrak{s}+d)(p) = (q, \sigma[c \mapsto (\sigma(c) + d) \mid c \in \mathcal{X}]$ iff $\mathfrak{s}(p) = (q, \sigma)$. With this, in state $\mathfrak{s}$ process p is able to play a send action $a = \langle q, l_R, e, l_P, \varphi, P \rangle$ after delay d if all receivers l_R are able to synchronize on message e after d. We define the receivable predicate as

$$\begin{aligned}\mathsf{rcvbl}(\mathfrak{s}, (p, d, a)) = \forall p' \in \mathfrak{s}_{\mathsf{val}}(p)(l_R) \setminus \{p\}\ \exists \langle (\mathfrak{s}+d)_{\mathsf{loc}}(p'), e, \varphi', \cdot \rangle \in \mathcal{R}_{\mathsf{D}}^? :&\\ (\mathfrak{s}+d)_{\mathsf{val}}(p') \cup \{\mathsf{snd} \mapsto \{p\}, \mathsf{param} \mapsto \mathfrak{s}_{\mathsf{val}}(p')(l_P)\} \models \varphi'&\end{aligned}$$

Note that this implements value matching and testing snd (e.g. against an expected sender identity) as a precondition for synchronization. With this, a move (p, d, a) is called *available* in $\mathfrak{s}$ iff it is receivable and enabled in $\mathfrak{s}$ after delay d, formally

$$\begin{aligned}&\mathsf{avail}(\mathfrak{s}, (p, d, a = \langle q, l_R, e, l_P, \varphi, \cdot \rangle)) =\\ &\quad \mathsf{rcvbl}(\mathfrak{s}, (p, d, a)) \quad \wedge\\ &\quad (\mathfrak{s}+d)_{\mathsf{val}}(p) \models \varphi \,\wedge\, (\mathfrak{s}+d)_{\mathsf{loc}}(p) = q\end{aligned}$$

Time Progress Condition Time progress in state $\mathfrak{s}$ is limited by a conjunction over the location invariants of all active processes in $\mathfrak{s}$. Formally, we define the set of *urgent delays* in state $\mathfrak{s}$ by

$$\mathsf{urgent}(\mathfrak{s}) = \Big\{ d \in \mathbb{R}_{\geq 0} \mid \mathfrak{s} + d \models \neg \bigwedge_{p \in \mathit{dom}(\mathfrak{s})} \mathit{inv}(\mathfrak{s}_{\mathsf{loc}}(p)) \Big\}.$$

With these ingredients, we are able to define the set of available moves $\Gamma(\mathfrak{s})$ below.

(4.) Transition Relation Moves can be divided in three classes, describing transitions where time passes only, a discrete step or inseparable first time passes by and then a discrete step takes place. In general, the set of successor states after playing a move is determined by (i) the effect of the update assignments and (ii) the probabilistic branches in the different DCPTAs.

(4.1) Update Assignments Locally, the simultaneous effect of assignments is well-defined because each local variable may occur at most once as a left-hand side variable. Special care however has to be taken for *create assignments*. In order to obtain a well-defined probability distribution below, we require a deterministic semantics for the process creation, that is, we have to determine which of the currently unused process identities is mapped to the new process configuration. Let $\Lambda_{cr} \subseteq \mathsf{Assign}(\mathcal{V}_{\mathsf{D}} \cup Prms)$ be the set of all create assignments in a DCS^{++} model D. Without loss of generality, we require both Λ_{cr} and Id to be totally ordered sets. This allows us to define the *create binding* depending on state $\mathfrak{s} \in \mathfrak{S}(\mathsf{D})$ as an injective function $\beta_{\mathfrak{s}} : \Lambda_{cr} \times \mathsf{Id} \to \mathsf{Id} \setminus dom(\mathfrak{s})$ that determines a unique identity for a pair (λ_{cr}, p), i.e. when executing a create assignment λ_{cr} in process p. With this, the effect on a state $\mathfrak{s} \in \mathfrak{S}(\mathsf{D})$ of create assignment $\lambda_{cr} = \mathsf{create}(\mathsf{B})$ with left-hand side l_c executed by process p is

$$\begin{aligned}
&[\![\langle \lambda_{cr}, q' \rangle]\!]_p(\mathfrak{s}, \beta_{\mathfrak{s}}) = \\
&\qquad\qquad \mathfrak{s}[\beta_{\mathfrak{s}}(\lambda_{cr}, p) \mapsto (q_0, \sigma_0[\mathsf{self} \mapsto \{\beta_{\mathfrak{s}}(\lambda_{cr}, p)\}])] \\
&\qquad\qquad [p \mapsto (q', \mathfrak{s}_{\mathsf{val}}(p)[l_c \mapsto \{\beta_{\mathfrak{s}}(\lambda_{cr}, p)\}]],
\end{aligned}$$

where (q_0, σ_0) is the initial configuration for DCPTA B.

The effect of a *sequence of update assignments* $\lambda = \tilde{\lambda}_1; \ldots; \tilde{\lambda}_n$ of process p is then the simultaneous effect of all basic and create assignments occurring in it, i.e.,

$$[\![\langle \lambda, q' \rangle]\!]_p(\mathfrak{s}, \beta_{\mathfrak{s}}) = [\![\langle \tilde{\lambda}_1, q'_1 \rangle]\!]_p(\ldots [\![\langle \tilde{\lambda}_n, q'_n \rangle]\!]_p(\mathfrak{s}, \beta_{\mathfrak{s}}) \ldots, \beta_{\mathfrak{s}}),$$

where $[\![\langle \tilde{\lambda}, q' \rangle]\!]_p(\mathfrak{s}, \beta_{\mathfrak{s}}) = \mathfrak{s}[p \mapsto (q', [\![\tilde{\lambda}]\!](\mathfrak{s}_{\mathsf{val}}(p)))]$ if $\tilde{\lambda} \in \mathsf{Assign}_0$ is a basic update assignment. For assignments λ occurring in receive edges, we by $[\![\langle \lambda, q' \rangle]\!]_p(\mathfrak{s}, \beta_{\mathfrak{s}}, m = (p', \cdot, a))$ denote the effect of λ on $\mathfrak{s}$ under move m by process p', where $a = \langle \cdot, \cdot, e, l_P, \cdot, \cdot \rangle$. Here, update assignments are evaluated considering also the values of the formal parameters induced by action a and the senders configuration $\mathfrak{s}_{\mathsf{val}}(p')$, that is, on the *enhanced* valuation $\mathfrak{s}_{\mathsf{val}}(p) \cup \{\mathsf{snd} \mapsto \{p'\}, \mathsf{param} \mapsto \mathfrak{s}_{\mathsf{val}}(p')(l_P)\}$. Note that the sender's identity is explicitly assigned to snd.

(4.2) Probability Distribution A move $m = (p, d, a = \langle q, l_R, e, l_P, \varphi, P \rangle)$ applied to state $\mathfrak{s}$ uniquely determines a discrete probability distribution $P_{\mathfrak{s},m} \in \mu(\mathfrak{S})$. To see this, recall that there is no non-deterministic choice among several receive edges. Thus in any process $p' \in \mathfrak{s}_{\mathsf{val}}(p)(l_R)$ edge $t_{p'} = \langle q', e, \varphi', P' \rangle \in \mathcal{R}^?_{\mathsf{D}}$ is triggered. We abuse notation and by $P^{p'}_{\mathfrak{s},m}$ refer to distribution P' triggered in state $\mathfrak{s}$ by m in p'. The effect of an update $[\![\langle \lambda, q' \rangle]\!]$ where the distributions $\{P, P^{p'}_{\mathfrak{s},m}\}_{p' \in \mathfrak{s}_{\mathsf{val}}(p)(l_R)}$ are ranging over, are deterministic and interference free by definition. Each $\langle \lambda, q' \rangle$ affects only the process that is executing this update and the disjoint set of processes created by it. Thus there is a bijection from the set of updates $\langle \lambda, q \rangle$ to the set of possible successor configurations (q', σ) that extends the distributions to range over $\mathfrak{S}$. The set of successor states is (unbounded but) finite as only finitely many creations may occur.

Formally, probability distribution $P_{\mathfrak{s},m}(\mathfrak{s}')$ is defined as shown in Fig. 4, where $R = \mathfrak{s}_{\mathsf{val}}(p)(l_R)$

$$P_{\mathfrak{s},m}(\mathfrak{s}') = \begin{cases} \sum P(\langle \lambda, q' \rangle) \cdot \prod\limits_{p' \in R} P^{p'}_{\mathfrak{s},m}(\langle \lambda_{p'}, q'_{p'} \rangle) \text{ iff } \mathsf{avail}(\mathfrak{s}, m) \wedge \\ \qquad\qquad\qquad \mathfrak{s}' \in [\![\langle \lambda, q' \rangle]\!]_p(\ldots [\![\langle \lambda_{p'}, q'_{p'} \rangle]\!]_{p' \in R}(\mathfrak{s}, \beta_{\mathfrak{s}}, m) \ldots, \beta_{\mathfrak{s}}), \\ \qquad\qquad\qquad \text{where } \langle \lambda, q' \rangle \in dom(P) \wedge \langle \lambda_{p'}, q'_{p'} \rangle \in dom(P^{p'}_{\mathfrak{s},m}) \\ \qquad 0, \qquad\qquad \text{otherwise} \end{cases}$$

Fig. 4. Probability distribution induced by state $\mathfrak{s}$ and a move m.

is the set of receivers triggered by m. If m is available in $\mathfrak{s}$, intuitively, $P_{\mathfrak{s},m}(\mathfrak{s}')$ amounts to the sum of the products of the weights determined by all updates available to sender p and (possible) receivers $p' \in R$, so that the effects yield state $\mathfrak{s}'$ under create binding $\beta_{\mathfrak{s}}$. The update assignments need not be permuted because they are by definition interference free.

DPTGS for DCS^{++} Using these definitions, a DCS^{++} model D describes the DPTGS $[\![\mathsf{D}]\!] = (\mathsf{S}, \Gamma, \mathsf{Act}, \delta)$ with $\mathsf{S} = \mathfrak{S}(\mathsf{D})$, $\mathsf{Act} = \mathsf{A}(\mathsf{D})$, $\Gamma(\mathsf{s}) =$

$$\big\{(p, d, a) \mid p \in dom(\mathsf{s}) \wedge \nexists t \in \mathsf{urgent}(\mathsf{s}) : t < d \wedge \mathsf{avail}(\mathsf{s}, (p, d, a))\big\} \,\dot\cup$$
$$\big\{(p, d, \ominus) \mid p \in dom(\mathsf{s}) \wedge \forall 0 \leq d' \leq d \nexists p' \in dom(\mathsf{s}) : (\mathsf{s} + d')_{\mathsf{val}}(p') \not\models inv(\mathfrak{s}_{\mathsf{loc}}(p'))\big\}$$

is the set of available moves in state s, and $\delta(\mathsf{s}, m) = \mathbf{P}_{\mathsf{s},m}$ is the probabilistic transition relation that depends on all the probability distributions of the edges triggered by move m in state s.

3 Specification Logic

In this section, we will describe our logic DPTATL which allows specifying properties of DCS^{++} models. The logic extends the alternating real-time aspects of TATL* [16] with probabilistic and topological reasoning, as introduced in PCTL [19] and METT [8], respectively.

Syntax. Similar to TATL*, DPTATL is divided into *state formulae* and *path formulae*, whose satisfaction is related to a state and a path, respectively. In the following, let A and C be finite sets of logical agent and clock names, respectively. DPTATL *state formulae* are inductively defined by:

(S1) $\neg\phi$ or $\phi \vee \phi'$, for state formulae ϕ and ϕ';
(S2) $q(a)$ or $l(a, a')$, for a location $q \in Q$, link $l \in \mathcal{L}$, and agents $a, a' \in A$;
(S3) $x + d_1 \prec y + d_2$, for clocks $x, y \in C$, constants $d_1, d_2 \in \mathbb{N}_0$, and $\prec \in \{<, \leq\}$;
(S4) $\forall a : \phi$ for agent $a \in A$ and state formula ϕ;
(S5) $a_1 = a_2$, for agents $a_1, a_2 \in A$;
(S6) $\langle\langle a_1, \ldots, a_n\rangle\rangle_{\succ p}\, \psi$, for agents $a_i \in A$, a comparator $\succ \in \{\geq, >\}$, a probability $p \in [0, 1]$, and a path formula ψ;
(S7) $\langle\langle a \mid \phi_a\rangle\rangle_{\succ p}\, \psi$, for a comparator $\succ \in \{\geq, >\}$, a probability $p \in [0, 1]$, a path formula ψ, and a propositional state formula ϕ_a (i.e., ϕ_a is defined using rules (S1) and (S2) only) in which only a is a free variable.

Rule (S1) corresponds to standard propositional logic. By rule (S2), we may query the location of some agent and check whether two agents are connected via a link. Rule (S3) formalizes logical clock constraints. Rules (S4) and (S5) add first-order quantification and test for equality of agents, respectively. By rule (S6), we introduce probabilistic strategy quantification. Finally, rule (S7) provides a second-order style strategy quantifier where the team of agents is characterized by a state formula ϕ_a. DPTATL *path formulae* are defined by:

(P1) any state formula;
(P2) $\neg\psi$ or $\psi \vee \psi'$, for path formulae ψ and ψ';
(P3) $x \cdot \psi$, for clock $x \in C$ and path formula ψ;
(P4) $Ǝ\, a : \psi$ for agent $a \in A$ and path formula ψ;
(P5) $\psi_1 \,\mathbf{U}\, \psi_2$, for path formulae ψ_1 and ψ_2.

As usual, rules (P1) and (P2) integrate state formulae and standard propositional logic. Rule (P3) adds a freeze quantifier that sets a clock variable for a given sub-formula, and rule (P4) introduces the birth quantifier Ǝ that bounds an agent to a fresh identity for a given sub-formula. Rule (P5) is the standard temporal until operator. We assume the usual abbreviations for *true*, *false*, $\wedge$ (conjunction), $\exists\, a : \phi$ (existential quantification), $\mathbf{F}$ (finally), and $\mathbf{G}$ (globally). Moreover, we set $Q(a) = \bigvee_{q \in Q} q(a)$ for a set of locations Q. If Q denotes the set of all locations of some DCPTA B, we can use this expression to state the agent's type, written as $a \,\mathsf{is}\, \mathsf{B}$. Finally, we use the freeze quantifiers from (P3) to obtain time-bounded temporal operators by writing $\mathbf{F}_{\leq d}\, \psi$ for $x \cdot \big(\mathbf{F}\, y \cdot (\psi \wedge y \leq x + d)\big)$.

Semantics. The semantic foundation of DPTATL is based on TATL* [16]. To this end, we import the notion of *game time* for paths of a DPTGS $\mathbb{S} = (\mathsf{S}, \Gamma, \mathsf{Act}, \delta)$. Intuitively, the game time is used to refer to the rounds of a path. As we allow moves of zero delay, several rounds of a path may have the same physical time. A *game time* [16] τ of a path ω, written as $\tau \in \mathsf{GameTimes}(\omega)$, is represented by a tuple $\langle t, k\rangle$, where $t \in \mathbb{R}_{\geq 0}$ is a physical time and $k \in \mathbb{N}_0$ refers to the k^{th} round at t. For a path ω, the set of game times is defined as

$$\mathsf{GameTimes}(\omega) = \left\{ \langle t, k\rangle \in \mathbb{R}_{\geq 0} \times \mathbb{N}_0 \mid \; 0 \leq k < |\{j \in \mathbb{N}_0 \mid \mathsf{time}(\omega, j) = t\}| \right\}.$$

We assume a lexicographical order over game times. Furthermore, for a path $\omega \in \mathsf{InfPth}$ and a game time $\langle t, k\rangle \in \mathsf{GameTimes}(\omega)$, we define $\omega^{\langle t,k\rangle} = \omega^{j+k}$, for the smallest $j \in \mathbb{N}_0$ with $\mathsf{time}(\omega, j) = t$ to refer to ω's game state at $\langle t, k\rangle$.

We define the semantics of DPTATL *state formulae* over tuples of the form $(\mathsf{s}, t, \mathcal{C}, \mathcal{A})$, where $\mathsf{s} \in \mathsf{S}$ is a state of $\mathbb{S}$, $t \in \mathbb{R}_{\geq 0}$ is a physical time, $\mathcal{C} : C \to \mathbb{R}_{\geq 0}$ is a valuation of logical clock variables, and $\mathcal{A} : A \to \mathsf{Id}$ is a valuation of logical agent variables, as follows:

$[\![\text{S2}]\!]$ $(\mathsf{s}, t, \mathcal{C}, \mathcal{A}) \models q(a)$ iff $\mathsf{s}_{\mathsf{loc}}(\mathcal{A}(a)) = q$;
$[\![\text{S2}]\!]$ $(\mathsf{s}, t, \mathcal{C}, \mathcal{A}) \models l(a, a')$ iff $\mathcal{A}(a') \in \mathsf{s}_{\mathsf{val}}(\mathcal{A}(a))(l)$;
$[\![\text{S3}]\!]$ $(\mathsf{s}, t, \mathcal{C}, \mathcal{A}) \models x + d_1 \leq y + d_2$ iff $\mathcal{C}(x) + d_1 \leq \mathcal{C}(y) + d_2$;
$[\![\text{S4}]\!]$ $(\mathsf{s}, t, \mathcal{C}, \mathcal{A}) \models \forall a : \phi$ iff $\forall p \in \mathsf{Id} : \mathsf{s}(p) \neq \bot \;\Rightarrow\; (\mathsf{s}, t, \mathcal{C}, \mathcal{A}[a \mapsto p]) \models \phi$;
$[\![\text{S5}]\!]$ $(\mathsf{s}, t, \mathcal{C}, \mathcal{A}) \models a_1 = a_2$ iff $\mathcal{A}(a_1) = \mathcal{A}(a_2)$;
$[\![\text{S6}]\!]$ $(\mathsf{s}, t, \mathcal{C}, \mathcal{A}) \models \langle\langle a_1, \ldots, a_n\rangle\rangle_{\succ p}\, \psi$, iff the team $T = \{\mathcal{A}(a_1), \ldots, \mathcal{A}(a_n)\}$ has a strategy π_T starting at state s such that for any counter strategy $\pi_{\overline{T}}$ and scheduler π_s starting at s: $\mathsf{P}(\mathsf{Outcomes}(\mathsf{s}, \pi_T, \pi_{\overline{T}}, \pi_s) \cap (\{\omega \in \mathsf{Timediv} \mid (\omega, \langle 0, 0\rangle, t, \mathcal{C}, \mathcal{A}) \models \psi\} \cup \mathsf{BlamelessOutcomes}(\mathsf{s}, \pi_T, \pi_{\overline{T}}))) \succ p$;
$[\![\text{S7}]\!]$ $(\mathsf{s}, t, \mathcal{C}, \mathcal{A}) \models \langle\langle a \mid \phi_a\rangle\rangle_{\succ p}\, \psi$, analogously to $[\![\text{S6}]\!]$ with team T defined as the set $\{p \in dom(\mathsf{s}) \mid (\mathsf{s}, t, \mathcal{C}, \mathcal{A}[a \mapsto p]) \models \phi_a\}$.

Strategy quantification requires that a team of agents can enforce the satisfaction of the path formula ψ with a probability $\succ p$. The two rules only differ in the determination of the teams: (S6) explicitly gives a set of agent variables, and (S7) denotes the team by an open state formula ϕ_a. Note that we measure the probability of those *time-divergent* outcomes which are *winning* for ψ, and those *time-convergent* ones for which no team T agent is to blame for.

We define the semantics of DPTATL *path formulae* over tuples of the form $(\omega, \tau, t, \mathcal{C}, \mathcal{A})$, where $\omega \in \mathsf{InfPth}$ is a suffix of a path of $\mathbb{S}$, $\tau \in \mathsf{GameTimes}(\omega)$ is a game time referring to a round in ω, $\mathcal{C} : C \to \mathbb{R}_{\geq 0}$ and $\mathcal{A} : A \to \mathsf{Id}$ are valuations of logical clock and agent variables as follows:

$[\![\text{P1}]\!]$ $(\omega, \langle u, k\rangle, t, \mathcal{C}, \mathcal{A}) \models \phi$ iff $(\omega^{\langle u,k\rangle}, t + u, \mathcal{C}, \mathcal{A}) \models \phi$;
$[\![\text{P3}]\!]$ $(\omega, \langle u, k\rangle, t, \mathcal{C}, \mathcal{A}) \models x \cdot \psi$ iff $(\omega, \langle u, k\rangle, t, \mathcal{C}[x \mapsto t + u], \mathcal{A}) \models \psi$;
$[\![\text{P4}]\!]$ $(\omega, \tau, t, \mathcal{C}, \mathcal{A}) \models \text{Ɐ}\, a : \psi$ iff $\exists p \in \mathsf{Id} : \big(\forall \tau' < \tau : \omega^{\tau'}(p) = \bot\big)$
$\qquad\wedge\; \omega^{\tau}(p) \neq \bot \;\wedge\; (\omega, \tau, \mathcal{C}, \mathcal{A}[a \mapsto p]) \models \psi$;
$[\![\text{P5}]\!]$ $(\omega, \tau, t, \mathcal{C}, \mathcal{A}) \models \psi_1 \,\mathbf{U}\, \psi_2$ iff $\exists \tau' : \tau \leq \tau' \wedge (\omega, \tau', t, \mathcal{C}, \mathcal{A}) \models \psi_2$
$\qquad\wedge\; \forall \tau'' : \tau \leq \tau'' < \tau' \Rightarrow (\omega, \tau'', t, \mathcal{C}, \mathcal{A}) \models \psi_1$.

The definitions for propositional part of the logic are canonical. Note that the birth query (P4) becomes true iff a *fresh* identity is bound to the agent $a \in A$ such that the corresponding path formula ψ is satisfied.

Satisfaction Relation Let $\mathsf{D} = (\{\mathsf{B}_1, \ldots, \mathsf{B}_n\}, \mathsf{B}^0)$ be a DCS^{++} model. Its initial state s_0 in the corresponding DPTGS $[\![\mathsf{D}]\!]$ is defined by $dom(\mathsf{s}_0) = \{p\}$ with p being the minimal element of Id and $\mathsf{s}_0(p) = (q_0, \sigma_0[\mathsf{self} \mapsto p])$ where q_0 is the initial location of B^0 and σ_0 its initial valuation. Then D satisfies a closed DPTATL formula ϕ, denoted as $\mathsf{D} \models \phi$, if $(\mathsf{s}_0, 0, \mathcal{C}_0, \mathcal{A}_0) \models \phi$, where $\mathcal{C}_0$ initializes all clock variables of ϕ to zero and $\mathcal{A}_0$ is the empty agent valuation.

4 Example

Routing in dynamic networks occurs in a multitude of modern applications, including intelligent transportation systems, telephone networks and data network streams. In the following we present a DCS^{++} model of adaptive packet routing in a dynamic network, where all the features combined in DCPTA (DPTGS) and DPTATL become essential. In dynamic networks, a varying number of nodes form *dynamically changing communication topologies* over time. Throughput and reliability of the network are crucial and require to factor in *real-time* as well as *probabilistic* aspects. At design time, given a still incomplete design, *realizability* of (quality of service) properties such as *"the team of all nodes in the network is able to ensure with probability p to eventually route each packet within time t"* have to be observed. The monolithic formalism of PTAs aims at real-time-probabilistic system aspects, but it lacks of explicit treatment of dynamic changing communication topologies and modular modeling of the processes making up the system. The logic PTCTL, typically applied to PTA, is no alternating logic and does not consider dynamic aspects.

In our model, over time, a dedicated *environment* process populates the dynamic network by creation of new *node* and *link* processes. The latter allows us to explicitly model error-prone (wireless) connections between nodes and thus to incorporate quantitative phenomena such as different probabilities of data packet loss. A link may be forced into a 'sabotaged mode' by the environment which results in a temporary decrease of its reliability and throughput. Also triggered by the environment, addressed data packets are transmitted from a source node to a target node, passing a sequence of intermediate links and nodes.

Note that most of the DCPTA comprise time and action non-determinism. The nodes, for example, are free to route packets to any of their outgoing links and the environment is free to arbitrarily schedule the tasks of creation, link sabotage and packet transmission (up to timing constraints ensuring minimal delays between sequent tasks). As we will see below, the strategy quantor of DPTATL determines a team of controllable processes, whose non-deterministic choices may (or may not) suffice to establish an objective.

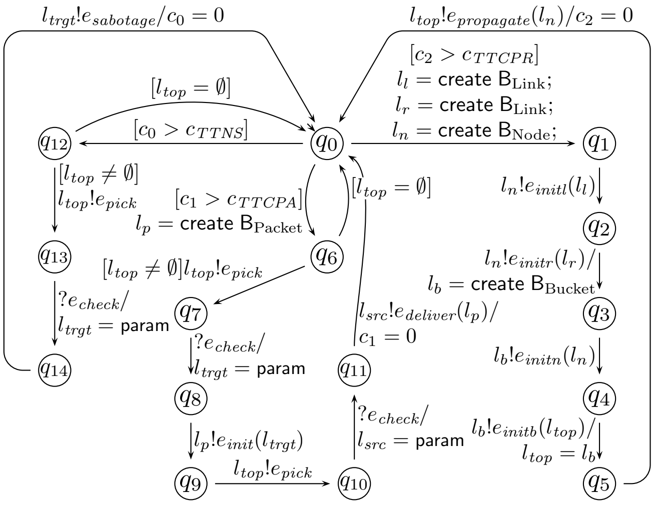

Fig. 5. $\mathsf{B}_{\mathrm{Env}}$

Model The *environment* process $\mathsf{B}_{\mathrm{Env}}$ (cf. Fig. 5) maintains all nodes in the network in a *bucket list* and repeatedly performs the following three tasks:

(1) Node Creation: At most every c_{TTCPR} (time to create process) time units, a new node is initialized with two new link processes. Then, a new bucket process is created and initialized with the newly created node. After this, the new bucket is inserted into the bucket list. Finally, all

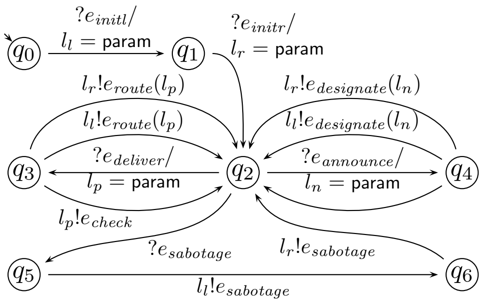

Fig. 6. B_{Node}

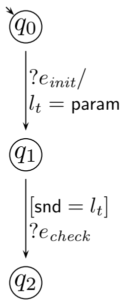

Fig. 7. $\mathsf{B}_{\text{Packet}}$

nodes in the network are made aware of the appearance of the new node by sending a *propagate* message to the head of the bucket list, attaching the newly created node as parameter.

(2) Packet Transmission: At most every c_{TTCPA} (time to create packet) time units, a new packet is created. A target node is picked from the bucket list by sending a *pick* message to the head of the list and awaiting a *check* message from some bucket in the list. The check message then contains the picked target node attached as parameter. The new packet is then initialized with the target node. Finally, a source node is picked in the same manner as the target node and a *deliver* message, with the packet attached, is send to the picked source node.

(3) Link Sabotage: Links may be of decreased reliability and throughput in routing packets through them. For link sabotage, at most every c_{TTNS} (time to next sabotage) time units, a *sabotage* message is sent to a node from the bucket list. The node forwards the message to both of its outgoing links, resulting in a time bounded switch to the degraded sabotage mode of the link processes.

A *node* process (cf. Fig. 6) is initialized with two link processes (not pointing to any node yet). If an initialized node receives a *deliver* message (either from another node or from the environment) with a packet attached as parameter, it can choose to either send a *check* message to the packet, or to route the packet to one of its outgoing links by sending a *route* message. If the node receives an *announce* message with a newly created node as parameter, it can update one of its outgoing links to this node and thereby change the network topology.

A *link* process (cf. Fig. 8) may receive a *designate* message with a node attached as parameter. This node then becomes the target node of the link. A link is either in 'normal mode' (q_0, q_1) or in 'sabotage mode' (q_2, q_3). The sabotage mode is triggered by reception of a *sabotage* message from the environment. The two modes specify different probabilities of message loss and different time intervals (denoted by constants min_{FTN}, max_{FTN} and min_{FTS}, max_{FTS}, respectively) in which a packet, received by a *route* message, can be transmitted to the next node on the path to the target node of the packet. A link has to stay in sabotaged mode for at least c_{TTR} (time to recover) time units.

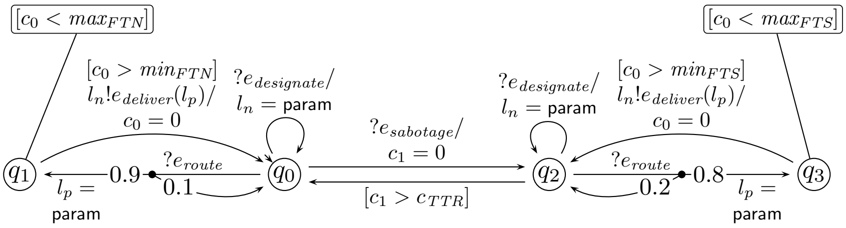

Fig. 8. $\mathsf{B}_{\mathrm{Link}}$

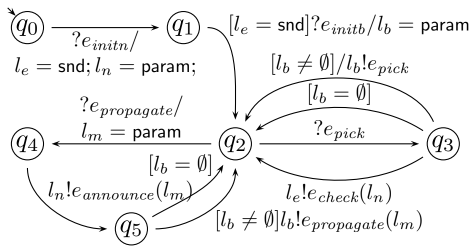

Fig. 9. $\mathsf{B}_{\mathrm{Bucket}}$

A *packet* process (cf. Fig. 7) is first initialized with a target node, i.e. the node where the packet is supposed to be routed to. It then waits for reception of a *check* messages during its route through the network. In case the designated target node is the sender of the message, the packet process enters terminal location q_2 and thereby 'disappears'.

A *bucket* process (cf. Fig. 9) serves as an item in the bucket list. The link variables l_n and l_b hold the identity of a node process and the next bucket in the list (if any), respectively. In case a bucket receives a *pick* message, it can choose to either send a *check* message to the environment or to forward the *pick* message to the next bucket in the list, if any. In case a bucket receives a *propagate* message with a node attached, it sends an *announce* message to its target node with the received node attached as parameter, and forwards the *propagate* message to the next bucket in the list, if any. That is, a *propagate* message 'travels' along the whole list whereas a *pick* message is free to travel up to any bucket in the list. Whenever the environment and all buckets play in the same team, *pick* allows the environment to control the choice of a *single* node, whereas *propagate* enforces the transmission of a node to *all* nodes in the list.

Requirements We now demonstrate how requirements for the DCS^{++} routing model can be formalized in DPTATL in a compact and elegant manner.

$$\langle\langle a \mid a \text{ is } \mathsf{B}_{\mathrm{Env}} \rangle\rangle_{=1}\, \mathbf{G}\, \mathbf{F}_{\leq c_{TTCPR}}\, \text{Ǝ}\, a' : \mathit{true} \tag{1}$$

$$\forall a' \text{ is } \mathsf{B}_{\mathrm{Packet}} : \langle\langle a \mid a \neq a' \rangle\rangle_{>0.5}\, \mathbf{F}_{\leq 20s}\, q_2(a') \tag{2}$$

$$\forall a \text{ is } \mathsf{B}_{\mathrm{Node}} : \mathbf{G}\, \big(\langle\langle a \rangle\rangle_{>0} \neg\, \mathbf{F}\, \exists a' : (l_l(a,a') \vee l_r(a,a')) \big) \tag{3}$$

Formula (1) expresses that the singleton team, consisting of the environment only, can certainly ensure that in each state, a new agent a' is created after at most c_{TTCPR} time units. Formula (2) expresses that for each packet a', the team of all agents except a' can ensure with probability greater than 50% that, after at most 20 seconds, packet a' is in terminal location q_2. Formula (3) expresses that for each node a, it is always the case that a alone can ensure with positive probability that none of its links points to some agent a'. Finally, the routing requirement from above, that is, the team of all nodes can always ensure with probability greater or equal 80% to (successfully) route each packet within 120 seconds, can be formalized in DPTATL as

$$\mathbf{G}\, \big(\langle\langle a \mid a \text{ is } \mathsf{B}_{\mathrm{Node}} \rangle\rangle_{\geq 0.8}\, \forall a' \text{ is } \mathsf{B}_{\mathrm{Packet}} : \mathbf{F}_{\leq 120s}\, q_2(a') \big)$$

5 Conclusion

Systems of systems (SoS) comprising a varying number of communicating processes are a prevalent class of complex and often safety critical systems that call for a rigid formal model-based design process. As of yet, only singular SoS aspects can be expressed by different modeling and requirement specification languages. In this paper, we propose a game-theoretic modeling and requirement specification framework that unifies all relevant aspects of SoS: modularity, real-time-, probabilistic- and dynamic behavior. The framework is unique in that it is a thoroughly designed concise orthogonal extension of well-known formalisms (and combinations thereof). By this, our work does also serves as a survey and meta-model for the relevant existing formalisms.

The framework lays the foundation for developing verification algorithms and tools for SoS. Existing techniques can be used whenever the restrictions to one of the integrated sublanguages apply. For analysing the full language, we currently investigate the decomposition of formal verification tasks (a corresponding article has been accepted for publication [13]). Also, we adapt finitary abstraction techniques to treat the inherent unboundedness in the number of processes (c.f. [25, 24]).

References

1. Alur, R., Courcoubetis, C., Dill, D.L.: Model-checking in dense real-time. Inf. Comput. 104(1) (1993)
2. Alur, R., Dill, D.L.: A theory of timed automata. Theor. Comput. Sci. 126(2), 183–235 (1994)
3. Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-time temporal logic. Journal of the ACM 49 (2002)
4. de Alfaro et. at., L.: The element of surprise in timed games. In: Amadio, R.M., Lugiez, D. (eds.) CONCUR. LNCS, vol. 2761, pp. 142–156. Springer (2003)
5. Attie, P.C., Lynch, N.A.: Dynamic input/output automata. In: Larsen, K.G., Nielsen, M. (eds.) CONCUR. LNCS, vol. 2154, pp. 137–151. Springer (2001)
6. Baier, C., Brázdil, T., Größer, M., Kucera, A.: Stochastic game logic. In: QEST. IEEE Computer Society (2007)
7. Baier, C., Ciesinski, F., Größer, M.: Probmela: a modeling language for communicating probabilistic processes. In: MEMOCODE. pp. 57–66. IEEE (2004)
8. Bauer, J., Schaefer, I., Toben, T., Westphal, B.: Specification and verification of dynamic communication systems. In: Goossens, K., Petrucci, L. (eds.) ACSD 2006. IEEE, Turku, Finland (Jun 2006)
9. Behrmann, G., David, A., Larsen, K.G., Mller, O., Pettersson, P., Yi, W.: UPPAAL - present and future. In: Proc. of 40*th* IEEE Conference on Decision and Control. IEEE Computer Society Press (2001)
10. Bohnenkamp, H.C., D'Argenio, P.R., Hermanns, H., Katoen, J.P.: Modest: A compositional modeling formalism for hard and softly timed systems. IEEE Trans. Software Eng. 32(10), 812–830 (2006)
11. Brand, D., Zafiropulo, P.: On Communicating Finite-State Machines. J. ACM 30(2), 323–342 (1983)
12. Chen, T., Lu, J.: Probabilistic alternating-time temporal logic and model checking algorithm. In: Lei, J. (ed.) FSKD (2). pp. 35–39. IEEE Computer Society (2007)
13. Damm, W., Peter, H.J., Rakow, J., Westphal, B.: Can we build it: Formal synthesis of control strategies for cooperative driver assistance systems. Mathematical Structures in Computer Science, Special Issue on Practical and Lightweight Formal Methods for the Design, Modeling and Analysis of Software Systems (2011), accepted for publication
14. Distefano, D., Katoen, J.P., Rensink, A.: On a temporal logic for object-based systems. In: Smith, S.F., Talcott, C.L. (eds.) FMOODS. vol. 177. Kluwer (2000)
15. Hansson, H., Jonsson, B.: A logic for reasoning about time and reliability. Formal Aspects of Computing 6, 102–111 (1994)
16. Henzinger, T.A., Prabhu, V.S.: Timed alternating-time temporal logic. In: Asarin, E., Bouyer, P. (eds.) FORMATS. LNCS, vol. 4202, pp. 1–17. Springer (2006)
17. Katoen, J.P., Baier, C.: Principles of Model Checking. MIT Press (2008)
18. Kwiatkowska, M., Norman, G., Sproston, J., Wang, F.: Symbolic model checking for probabilistic timed automata. Information and Computation 205(7) (2007)
19. Kwiatkowska, M.Z., Norman, G., Segala, R., Sproston, J.: Automatic verification of real-time systems with discrete probability distributions. Theor. Comput. Sci. 282(1), 101–150 (2002)

20. Kwiatkowska, M.Z., Norman, G., Trivedi, A.: Quantitative games on probabilistic timed automata. CoRR abs/1001.1933 (2010)
21. Milner, R.: The Pi Calculus. CU Press (1999)
22. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley-Interscience (1994)
23. Rozenberg, G. (ed.): Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1: Foundations. World Scientific (1997)
24. Toben, T.: Analysis of Dynamic Evolution Systems by Spotlight Abstraction Refinement. Ph.D. thesis, Carl von Ossietzky Universität Oldenburg, Germany (2009)
25. Westphal, B.: Specification and Verification of Dynamic Topology Systems. Ph.D. thesis, Carl von Ossietzky Universität Oldenburg, Germany (2008)
26. Yahav, E., Reps, T., Sagiv, M., Wilhelm, R.: Verifying temporal heap properties specified via evolution logic. In: In ESOP 2003, LNCS. pp. 204–222 (2003)
27. Yi, W.: Real-time behaviour of asynchronous agents. In: Baeten, J.C.M., Klop, J.W. (eds.) CONCUR. LNCS, vol. 458, pp. 502–520. Springer (1990)