Parameter Synthesis for Simulation Distances Between Weighted Transition Systems

Anders Mariegaard & Julian Ringsmose
{amarie10, jrings10} @ student.aau.dk

Supervisors: Kim G. Larsen & Radu Mardare

Aalborg University, Department of Computer Science

June 1, 2015

Abstract

This thesis addresses the problem of parameter synthesis for simulation distance checking of parametric weighted transition systems. The usual notion of simulation relations is extended to allow point-wise deviations in transition weight matching, inducing a directed distance between states. The logical implications of distances between states is investigated using a parametric extension of Weighted CTL. The main contribution of this work is the utilization of parametric symbolic dependency graphs to represent the problem of checking whether or not the distance between two transition system states are below some threshold using parameter synthesis of the problem. To this end we present a global fixed point algorithm which has been implemented with a web-based front-end.

1 Introduction

Recently, within the area of embedded and distributed systems, a significant work has been invested in various modeling formalisms for specification of quantitative properties. A particularly popular modeling formalism for analyzing timing constraints of systems is timed automata [1] used in tools such as UPPAAL [17] and KRONOS [7]. In addition to timing constraints, general resource consumption and production has also been studied in the context of Hybrid Automata [14, 15], Priced Timed Automata [3] and Weighted Timed Automata [6]. The semantics of such quantitative systems is often defined as a Weighted (Labelled) Transition System, which is a traditional transition system with a transition-labelling function that associates to each transition a *cost*.

In this thesis we extend the concept of Weighted Transition Systems (WTS) to *Parametric* Weighted Transition Systems (PWTS), where transition labels are linear expressions in parameters. These parameters allow for specification of *unknown* or *unspecified* cost of quantitative behavior. A single PWTS thus represents an infinite set of WTSs - one for each possible instantiation of the parameters.

An example PWTS is shown in Figure 1. It depicts two systems, the left one being a regular coffee machine with states *start*, $coffee$ and *done*. The coffee machine accepts a request by button press after which it takes 200ms before it starts brewing the coffee, formalised by the transition with weight 200 going to the `coffee` state. It then takes 5 seconds (5000 ms) before the user can pick up the coffee, after which the machine returns to the *start* state after 100ms.

A group of computer scientists have been using this machine for a long time and appreciate its coffee but consider buying a new advanced beverage machine. This machine can serve regular coffee as well as espresso and soda and is depicted (with timing quantities) in the right part of Figure 1. For this advanced machine, the start delay before it can brew is longer than for the

regular coffee machine. The time spend for the brewing part of the process is unknown, but the manufacturer knows for a fact that espresso, tea and coffee have different brewing times, represented by the parameters p, q, r. Furthermore the manufacturer is quite certain that the new machine is at least as slow as the old one.

The scientists are eager to know how large the deviation in time spend by the new machine before a hot beverage is ready compared to that of the old machine. We can see this time deviation as an error-term ε in a relation where the old machine is simulated approximately by the new machine. For this example the error ε is characterized by the following constraints on the parameters p, q, r:

$$250 - 200 \leq \varepsilon \text{ and } 5000 - p \leq \varepsilon \text{ or}$$
$$230 - 200 \leq \varepsilon \text{ and } 5000 - q \leq \varepsilon \text{ or}$$
$$210 - 200 \leq \varepsilon \text{ and } 5000 - r \leq \varepsilon$$

Depending on the how the performance of the new machine turns out in the different brewing disciplines (e.g. the actual value of p, q, r) either the first, second or third pair of constraints ends up characterizing the maximal point-wise difference between the quantitative behavior of the beverage machines.

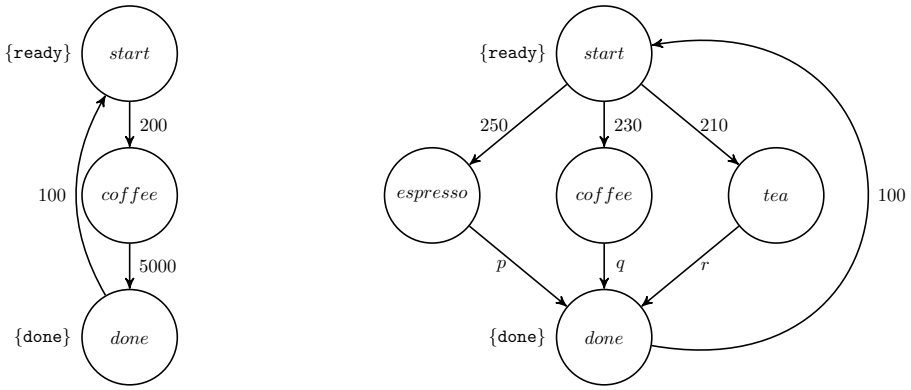


Figure 1: Old coffee machine (left) and new beverage machine (right)

To formally discuss problems of this nature we present the classical notion of simulation-relations between states to reason about correctness of an implementation with respect to a specification. As the specification or implementation may contain weight estimates or unknown values, we expand the notion of simulation by allowing for deviations in the transition weight matching by introducing an error term ε. The transition matching is then changed to either *absolute* and *relative* matching of weights, with ε as an upper bound on the error.

These error terms are then formally captured as directed distances between states of a PWTS. These distances are shown to be computable as least fixed points by use of Tarski's fixed point theorem for complete lattices [19]. We also investigate logical properties of similar systems by relating the notion of distance to logical properties by considering a parametric extension to Weighted CTL [8]. This extension allows for parametric reasoning by introducing parametric expressions on lower and upper bounds on path formulae and we show that if the distance between a state s and a state t is less than some ε, it is the case that if s satisfies a formula Φ, t is guaranteed to satisfy a formula Φ^ε. The formula Φ^ε is a modified version of Φ, where the path formula bounds have been stretched to allow for an error of ε. For the coffee machine example, we can this logic to state that the time delay before the machine starts brewing is exactly 200ms and the time spend at brewing must be exactly 500ms as follows:

$$\Phi = EX_{[200,200]}(EX_{[5000,5000]}\texttt{done})$$

As we know that the old machine is ε-simulated by the new, we thus know that

$$\Phi^\varepsilon = EX_{[200-\varepsilon,200+\varepsilon]}(EX_{[5000-\varepsilon,5000+\varepsilon]}\texttt{done})$$

is satisfied by the *start* state in the new machine. Notice that ε in the above Φ^{ε} is given by the constraints on the parameters from the discussion on distance between the old and new beverage machine.

The main contribution of this thesis is an extension to the online tool, originally introduced by [9] on `pvtool.dk` to model-check Parametric Weighted CTL on PWTS. Our extension uses the concept of *Parametric Symbolic Dependency Graphs* (PSDGs) to represent the problem of determining the point-wise distance between PWTS states. To this end we use a fixed point algorithm on cost *assignments* to PSDG nodes to synthesize parameter valuations in order to answer whether or not the point-wise distance meets some upper bound requirement ε. We prove termination and correctness of our approach and briefly discuss implementation details concerning the efficiency of our approach.

2 Related Work

In this thesis we contribute to the area of parametric quantitative modelling. In recent time classical quantitative modelling formalisms such as Timed Automata [1], Weighted Timed Automata [6], Priced Timed Automata [3] and Hybrid Systems [14] have been extended by allowing unknown or parametric values as quantitites, thus adding parametric analysis to the well known quantitative analysis of systems. Alur et. al presents the notion of Parametric Timed Automata [2] also used in [4] by Benes et. al where the decidability for the language emptiness problem is examined by varying the number of (parametric) clocks allowed as well as the number of parameters used in clock comparisons. In this work we look at the basic semantics of these systems; the Weighted Transition System and extend it with parametric expressions as weights on transitions.

Through accumulated and point-wise simulation distances, Fahrenberg et. al. provides notions of approximate simulation relations between weighted transition systems [13]. We instantiate the proposed point-wise distance in our parametric setting to measure both *absolute* and *relative* point-wise distance from one state to another. In a discussion on logical implications of the distance between Weighted Kripke Structures, Fahrenberg et. al [12] introduces a notion of discounted satisfiability to reason about how "close" to satisfaction of a WCTL formula a given state is. Inspired by that we introduce a parametric weighted extension of CTL with both upper and lower bounds on path formulae. To relate the simulation-distance sfrom one state to another to the satisfiability of logical formulae, we introduce the notion of ε-satisfiability, allowing for expansion of lower and upper bounds in the path formulae.

Finnemann et. al [16] extends the theory of fixed point computations on *Dependency Graphs* introduced by Smolka et. al [18] to the quantitative setting with *Symbolic Dependency Graphs* which allows for upper bound constraints on quantitative properties. Symbolic Dependency Graphs are then used for model-checking of weighted systems w.r.t weighted CTL by computing fixed points on *cost* assignments to nodes in the graph. Inspired by this, Christoffersen et. al [9] propose *Parametric Dependency Graphs* to allow for parameter synthesis for model-checking *parametric* weighed systems w.r.t a parametric extension of weighted CTL. In this work we consider Parametric Dependency Graphs for characterization of constraints on parameter valuations in the context of simulation-distance checking. We present a fixed point algorithm for computing the exact constraints on valuation of the parameters on transition weights for the distance between two systems to be below some ε. Finally, our method is implemented as an extension to the tool presented by Christoffersen et. al [9].

3 Model Specification and Behavior

In this section we define and discuss modelling formalisms for specification and verification of quantitative systems. The basic formalism used in this work is (Labelled) Weighted Transition Systems with transition weights from the set of non-negative rationals, $\mathbb{Q}_{\geq 0}$. To reason about behavior of systems we define several notions of *simulation* between systems. These notions are then lifted to what we call Parametric Weighted Transition Systems where *unknown* cost

or behavior can be encoded by linear expressions in parameters on transition weights. In both formalisms we let $\mathcal{AP}$ be a fixed finite set of atomic propositions.

3.1 Weighted Transition Systems

The models we consider in this section are based on *weighted transition systems* extended by a state-labelling function.

Definition 1. A *Weighted Transition System (WTS)* $\mathcal{S}$ is a triple

$$\mathcal{S} = (S, \rightarrow, \ell), \text{ where}$$

- S is a finite non-empty set of *states*.
- $\rightarrow \subseteq S \times \mathbb{Q}_{\geq 0} \times S$ is the finite transition relation.
- $\ell : S \longrightarrow 2^{\mathcal{AP}}$ is a labeling function mapping states in S to a set of atomic propositions

We will use $\mathfrak{S}$ to denote the set of all WTSs, $s \xrightarrow{w} s'$ as a shorthand notation for $(s, w, s') \in \rightarrow$ and existence and non-existence of a transition from a state s, with weight w, by $s \xrightarrow{w}$ and $s \not\xrightarrow{w}$ respectively. Finally, we let $\mathcal{W}^{\mathcal{S}} = \{w \mid (s, w, s') \in \rightarrow\}$ denote the set of all weights present in a WTS $\mathcal{S}$.

A path π in a WTS $\mathcal{S} = (S, \rightarrow, \ell)$ is a possibly infinite sequence $\pi = s_0 w_1 s_1 w_2 s_2 ...$ where $\forall i \in \mathbb{N}, s_i \in S$ and for $i > 1, w_i \in \mathbb{Q}_{\geq 0}$ s.t. $s_{i-1} \xrightarrow{w_i} s_i$. $\Pi(s)$ denotes all paths π starting from s and $\Pi(\mathcal{S})$ denotes all paths in $\mathcal{S}$. $\pi(i)_s$ denotes the state of π with index i, i.e s_i, $\pi(i)_w$ denotes the weight with index i, i.e w_i and $|\pi|$ denotes the length of a finite path π given by the number of states. For infinite paths π we let $|\pi| = \infty$.

The *accumulated weight*, $\mathcal{AW}_\pi(j)$, of π at position j is defined by $\mathcal{AW}_\pi(j) = \sum_{i=1}^{j} w_i$ if $j > 0$ and $\mathcal{AW}_\pi(j) = 0$ if $j = 0$. Finally, we denote by $\boldsymbol{out}(s) = \{(w, s') \mid (s, w, s') \in \rightarrow\}$ the set of all weight and successor-state pairs of outgoing transitions from s.

3.2 Simulation

To reason about behavior of WTSs we introduce the usual notion of simulation preorders. The intuition is that a given state s is simulated by another state t if t can "behave" exactly like s. This is useful in correctness checking of systems; one may want that any implementation behavior is allowed by the specification or that the implementation behavior is a superset of the specification behavior.

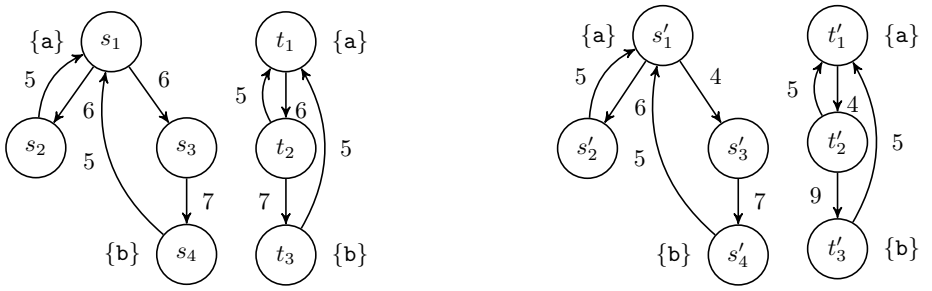
Definition 2. Let $\mathcal{S} = (S, \rightarrow, \ell)$ be a WTS. A *simulation* relation on S is a binary relation $\mathcal{R} \subseteq S \times S$ such that whenever $(s, t) \in \mathcal{R}$,

1. $\ell(s) = \ell(t)$ and
2. if $s \xrightarrow{w} s'$, then there exists a transition $t \xrightarrow{w} t'$ s.t. $(s', t') \in \mathcal{R}$.

From the definition we see that a simulation of a state s by a state t corresponds to a *point-wise* exact matching of transition weights and state labels. Let $\mathcal{S} = (S, \rightarrow, \ell)$ be a WTS. A state s is simulated by another state t, written $(\mathcal{S}, s) \leq (\mathcal{S}, t)$, iff there exists a simulation $\mathcal{R}$ on S s.t $(s, t) \in \mathcal{R}$. If the WTS is clear from the context, we write $s \leq t$.

Example 1. Consider the WTS $\mathcal{S}$ in Figure 2a. One can quickly verify that the relation $\mathcal{R} = \{(s_1, t_1), (s_2, t_2), (s_3, t_2), (s_4, t_3)\}$ is a simulation relation i.e $s_1 \leq t_1, s_2 \leq t_2, s_3 \leq t_2$ and $s_4 \leq t_3$ ($t_1 \leq s_1$ is also the case). For WTS $\mathcal{S}'$ in Figure 2b, the weights have changed and $s'_1 \not\leq t'_1$ as the weights do not match exactly.

The example shows that the usual definition of simulation relations does not allow for deviations in matching of transition weights. This may in some cases be a serious issue as specified weights may be estimates or guesses, leading to the implementation being incorrect



(a) WTS $\mathcal{S}$ (b) WTS $\mathcal{S}'$

Figure 2: Two WTSs

in the traditional boolean sense while its behavior would be "close" to the requirements of the specification in a quantitative setting. We now proceed by presenting two variations of simulation that take these deviations into account.

Definition 3. Let $\mathcal{S} = (S, \to, \ell)$ be a WTS and $\delta \in \mathbb{Q}_{\geq 0}$. A *$\delta$-simulation* relation on S is a binary relation $\mathcal{R} \subseteq S \times S$ such that whenever $(s,t) \in \mathcal{R}$,

1. $\ell(s) = \ell(t)$ [1]
2. if $s \xrightarrow{w} s'$, then there exists a transition $t \xrightarrow{w'} t'$ s.t. $|w - w'| \leq \delta$ and $(s', t') \in \mathcal{R}$.

For δ-simulation we require label matching but now the weight matching has been relaxed to allow for point-wise *absolute* deviations of at most δ. Let $\mathcal{S} = (S, \to, \ell)$ be a WTS. A state $s \in S$ is δ-simulated by another state $t \in S$, written $(\mathcal{S}, s) \leq^a_\delta (\mathcal{S}, t)$, iff there exists a δ-simulation $\mathcal{R}$ on S s.t $(s,t) \in \mathcal{R}$. Again, if the WTS is clear from the context, we write $s \leq^a_\delta t$. It is clear that $\leq^a_0 = \leq$ as no deviation leads to exact matching.

Example 2. Consider Figure 2b. We concluded that $s'_1 \not\leq t'_1$ but it is clear, by Definition 3 that $s'_1 \leq^a_\delta t'_1$ for any $\delta \geq 2$ as $s'_1 \xrightarrow{6} s'_2$ must be matched by $t'_1 \xrightarrow{4} t'_2$ and $s'_3 \xrightarrow{7} s'_4$ by $t'_2 \xrightarrow{9} t'_3$. This means that the largest absolute deviation in a single step is bounded by 2 and the relation $\mathcal{R}$ from Example 1 serves as an δ-simulation relation for any $\delta \geq 2$.

As shown above, δ-simulation is less strict than simulation but it also reveals a possible flaw, depending on the application context. If we again consider the matching of the weight 6 with 4 we see that the difference $6 - 4 = 2$ is equal to a relative deviation of $\frac{2}{6} \approx 33\%$. For 7 and 9 the difference $7 - 9 = 2$ equals an relative deviation of $\frac{2}{9} \approx 22\%$. To capture a relative requirement on the point-wise transition matching we define a notion of simulation called *relative δ-simulation*.

Definition 4. Let $\mathcal{S} = (S, \to, \ell)$ be a WTS with strictly positive weights and $\delta \in \mathbb{Q}_{\geq 0}$. A *relative δ-simulation* relation on S is a binary relation $\mathcal{R} \subseteq S \times S$ such that whenever $(s,t) \in \mathcal{R}$,

1. $\ell(s) = \ell(t)$
2. if $s \xrightarrow{w} s'$, then there exists a transition $t \xrightarrow{w'} t'$ s.t. $|\frac{w'-w}{w}| \leq \delta$ and $(s', t') \in \mathcal{R}$.

Let $\mathcal{S} = (S, \to, \ell)$ be a WTS. A state $s \in S$ is *relatively* δ-simulated by another state $t \in S$, written $(\mathcal{S}, s) \leq^r_\delta (\mathcal{S}, t)$, iff there exists a relative δ-simulation $\mathcal{R}$ on S s.t. $(s,t) \in \mathcal{R}$. If the WTS is clear from the context, we write $s \leq^r_\delta t$. It is clear that $\leq^r_0 = \leq$ as no deviation leads to exact matching. We thus have $\leq^r_0 = \leq^a_0 = \leq$. Note that the WTS must have only positive weights to avoid division by zero.

[1] Weights on labels could also be considered to allow for distances between labels.

Example 3. Consider Figure 3. We can now apply Definition 4 to show that $s_1 \leq_\delta^r t_1$ for any $\delta \geq 1$. This implies that the largest relative deviation in a single step of the simulation is 100%. It is worth noting that the point(s) corresponding to the maximal *relative* difference may not be the same as the point(s) that correspond(s) to the maximal *absolute* difference in weights (as in Figure 2b). Figure 3 depicts a WTS where the maximal absolute difference between weights considered by δ-simulation is $|100 - 110| = 10$. The minimum considered is $|1 - 2| = 1$. For relative δ-simulation, weights 1 and 2 amount to a relative difference of 100% and weights 100 and 110 to a relative difference of 10%.

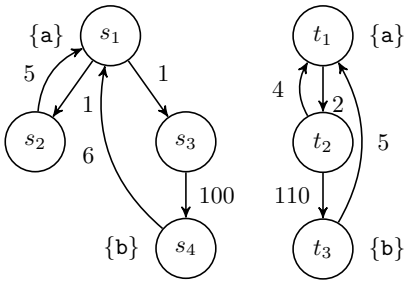


Figure 3: WTS showing (small) big absolute differences that are relatively (big) small

Both δ-similarity and relative δ-similarity intuitively induces a kind of directed distance on WTS states, based on their point-wise behavior. This we formalize in Section 4 by relating certain hemi-metrics to the simulation preorders.

3.3 Parametric Weighted Transition Systems

In this section we lift the the notion of WTS to a parametric setting by allowing specification of *unknown* quantitative behavior directly as linear expressions over parameters, as transition weights in a *Parametric Weighted Transition System* (PWTS). We also lift the three simulation relations described in the previous chapter by considering valuations of parameters (and linear expressions in parameters) so that we can instantiate a given PWTS as a WTS. The problem of verifying whether a given state simulates another state then changes from a simple check of weight and label matching to an investigation of the existence of "good" parameter *valuations*. We let $\mathcal{P} = \{p_1, \ldots p_n\}$ be a fixed finite set of parameters. Then,

$$\mathcal{E} = \{e \mid e = \sum_{i=0}^{n} a_i p_i + b \text{ where } a_i, b \in \mathbb{Q}_{\geq 0}\}$$

denotes the set of all linear expressions over parameters drawn from the set $\mathcal{P}$, with non-negative rational coefficients.

Definition 5. A *Parametric Weighted Transition System* (PWTS) $\mathcal{S}$ is a triple

$$\mathcal{S} = (S, \rightarrow, \ell), \text{ where}$$

- S is a finite non-empty set of *states*.
- $\rightarrow \subseteq S \times \mathcal{E} \times S$ is the finite transition relation.
- $\ell : S \longrightarrow 2^{\mathcal{AP}}$ is a labeling function mapping states in S to a set of atomic propositions

Whenever $(s, e, s') \in \rightarrow$ we use the shorthand notation $s \xrightarrow{e} s'$. We will use $\mathfrak{S}^{\mathcal{P}}$ to denote the set of all PWTSs. We denote existence and non-existence of a transition from a state s, with weight e, by $s \xrightarrow{e}$ and $s \not\xrightarrow{e}$ respectively. Paths, accumulated weights and ***out*** are defined similar to those for WTS.

One can instantiate a PWTS to a WTS by mapping each parameter from the transition weights to a number in $\mathbb{Q}_{\geq 0}$. A PWTS can thus be seen as an infinite set of WTSs. The notion

of mapping parameters to $\mathbb{Q}_{\geq 0}$ will be referred to as *parameter valuations*. We start by defining parameter valuations directly on parameters and proceed by extending the notion to PWTS.

Definition 6. A parameter valuation is a function:

$$v : \mathcal{P} \longrightarrow \mathbb{Q}_{\geq 0}$$

We denote the set of all parameter valuations by $\mathcal{V}$ and extend valuations to the domain $\mathcal{E}$;

$$v(x_1 p_1 + ... + x_k p_k + x_{k+1}) = x_1 v(p_1) + ... + x_k v(p_k) + x_{k+1}$$

We now extend valuations to the domain of PWTS.

Definition 7. Given a PWTS $\mathcal{S} = (S, \rightarrow, \ell)$ and a parameter valuation $v \in \mathcal{V}$, we define its WTS instance w.r.t v to be $\mathcal{S}' = (S, \rightarrow', \ell)$ where

$$\rightarrow' = \{(s, v(e), s') \mid (s, e, s') \in \rightarrow\}$$

Given a valuation v we thus simply map any transition expression to a weight $w \in \mathbb{Q}_{\geq 0}$ according to v to obtain a structurally identical WTS. We now have sufficient notational power to define the simulation preorders on PWTS.

Definition 8. Let $\mathcal{S} = (S, \rightarrow, \ell)$ be a PWTS, $s, t \in S$ two states and $\varepsilon \in \mathcal{E}$. We define parametric versions of the three simulation preorders for arbitrary $v \in \mathcal{V}$ as follows:

$$\begin{array}{lll} (\mathcal{S}, s) \leq_v (\mathcal{S}, t) & \text{iff} & (v(\mathcal{S}), s) \leq (v(\mathcal{S}), t) \\ (\mathcal{S}, s) \leq^a_{\varepsilon,v} (\mathcal{S}, t) & \text{iff} & (v(\mathcal{S}), s) \leq^a_{v(\varepsilon)} (v(\mathcal{S}), t) \\ (\mathcal{S}, s) \leq^r_{\varepsilon,v} (\mathcal{S}, t) & \text{iff} & (v(\mathcal{S}), s) \leq^r_{v(\varepsilon)} (v(\mathcal{S}), t) \end{array}$$

As for WTSs, we omit the system when it is clear from the context.

Example 4. Consider the PWTS depicted in Figure 4. Below we list simulation properties as well as the characteristics of valuation witnesses w.r.t some $\varepsilon \in \mathcal{E}$.

$$s_1 \leq_v t_1 \qquad \text{if } v(r) = v(q) \tag{1}$$

$$s_1 \leq^a_{\varepsilon,v} t_1 \qquad \text{if } |v(p) - v(q)| \leq v(\varepsilon) \text{ and } |v(q) - v(s)| \leq v(\varepsilon) \tag{2}$$

$$s_1 \leq^r_{\varepsilon,v} t_1 \qquad \text{if } \left| \frac{v(q) - v(r)}{v(q)} \right| \leq v(\varepsilon) \tag{3}$$

One should note that multiple disjoint possibilities exist. For (1), another possibility is $v(q) = v(p) = v(s)$ due to the branching right half of the PWTS. Similarly, for (2), $|v(q) - v(r)| \leq \varepsilon$ describes a set of witnesses.

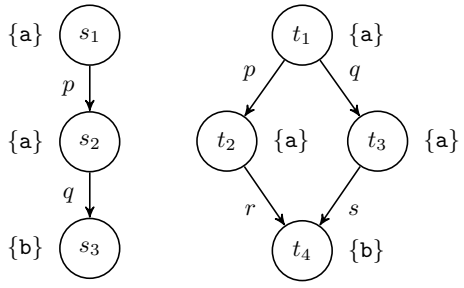


Figure 4: PWTS $\mathcal{S}$

The above example contains parameters both in the specification and implementation. Another example that is interesting from a practical point of view is one where the specification contains no parameters while the implementation has a parametrised component (e.g unknown cost). For this example the problem entails finding an instance of the implementation that respects the specification w.r.t a given simulation preorder. With such a characterization of

the correct implementations, one may be able to reject or accept a given implementation design by incorporating expert knowledge about upper and lower bounds on the cost or resource consumption of the given part.

In the remainder of this thesis we use the models and simulation preorders defined in this section as the basis for reasoning about quantitative systems with possible unknown or unspecified behavior. Concretely, we consider the following four problems for two PWTS states s, t:

1. Decide whether or not a valuation v exists such that $s \leq^a_{\varepsilon,v} t$ ($s \leq^r_{\varepsilon,v} t$).
2. Synthesize a valuation v such that $s \leq^a_{\varepsilon,v} t$ ($s \leq^r_{\varepsilon,v} t$).
3. Characterize the sets of valuations $V_1 = \{v \mid v \in \mathcal{V} \text{ and } s \leq^a_{\varepsilon,v} t\}$ and $V_1 = \{v \mid v \in \mathcal{V} \text{ and } s \leq^r_{\varepsilon,v} t\}$.
4. Synthesize the valuation v that minimizes ε for the relation $s \leq^a_{\varepsilon,v} t$ ($s \leq^r_{\varepsilon,v} t$).

4 Simulation Distances

In this section we will define two directed point-wise distances on WTS states, based on absolute and relative weight differences. We then show the relation between the three simulation preorders and the distances defined. These results are finally lifted to PWTSs by applying parameter valuations to project the problem to the non-parametric setting. We start by defining what a hemi-metric is:

Definition 9. Let X be a set. A function $d : X \times X \longrightarrow \mathbb{Q}_{\geq 0} \cup \{\infty\}$ is called a *hemi-metric* if the following properties hold:

1. $d(x, x) = 0$ for all $x \in X$
2. $d(x, z) \leq d(x, y) + d(y, z)$

If, in addition to (1)-(3), we have that $d(x, y) = d(y, x)$ for all $x, y \in X$; and $d(x, y) = 0$ implies $x = y$ for all $x, y \in X$, d is called a *metric*.

As in [13] we now define the notion of a *point-wise* distance between the states in a WTS as a hemi-metric. Note that we make use of a metric directly on weights in the WTS.

Definition 10. Let $\mathcal{S} = (S, \to, \ell)$ be a WTS and $s, t \in S$ two states. Given a metric $d : \mathcal{W}^{\mathcal{S}} \times \mathcal{W}^{\mathcal{S}} \longrightarrow \mathbb{Q}_{\geq 0} \cup \{\infty\}$, we define the point-wise distance $d_\bullet(s, t)$ from s to t to be the least fixed point of the following equations

$$d_\bullet(s,t) = \begin{cases} \infty & \text{if } \ell(s) \neq \ell(t) \\ \max\limits_{s \xrightarrow{w} s'} \min\limits_{t \xrightarrow{w'} t'} \max\{d(w, w'), d_\bullet(s', t')\} & \text{otherwise} \end{cases}$$

For the WTS $\mathcal{S} = (S, \to, \ell)$ we denote the set of all mappings of the form $S \times S \longrightarrow \mathbb{Q}_{\geq 0} \cup \{\infty\}$ by $\mathfrak{D}$. Then we let $\boldsymbol{H} : \mathfrak{D} \longrightarrow \mathfrak{D}$ be defined for a given metric $d : \mathcal{W}^{\mathcal{S}} \times \mathcal{W}^{\mathcal{S}} \longrightarrow \mathbb{Q}_{\geq 0} \cup \{\infty\}$:

$$\boldsymbol{H}(D)(s,t) = \begin{cases} \infty & \text{if } \ell(s) \neq \ell(t) \\ \max\limits_{s \xrightarrow{w} s'} \min\limits_{t \xrightarrow{w'} t'} \max\{d(w, w'), D(s', t')\} & \text{otherwise} \end{cases}$$

It follows from [13] that we can order elements from $\mathfrak{D}$ by the component-wise ordering $\leq$ to obtain a lattice $(\mathfrak{D}, \leq)$. The lattice is complete as the set of non-negative rationals form a complete lattice $(\mathbb{Q}_{\geq 0}, \leq)$ with meet ($\wedge$) and join ($\vee$) defined by the infimum and supremum, respectively. $\boldsymbol{H}$ is clearly monotone w.r.t $(\mathfrak{D}, \leq)$ and by Tarski's fixed point theorem [19] we thus have that $\boldsymbol{H}$ has a least (pre)fixed point i.e $d_\bullet$. By the definition we know that for a given WTS $\mathcal{S} = (S, \to, \ell)$ there is a finite number of transition matches, $|w - w'|$. This ensures that

the fixed point is found after a finite number of repeated applications. Finally, by [13], we have that $d_\bullet$ is a hemi-metric w.r.t any metric d.

To create point-wise distances that relate to the simulation preorders defined earlier (in Section 3.2), we need the metric d in Definition 10 to calculate either the absolute difference of weights, or the relative difference, i.e $d(w,w') = |w - w'|$ or $d(w,w') = |\frac{w'-w}{w}|$. The two distances we obtain are the least fixed points to the following equations:

$$d^a_\bullet(s,t) = \begin{cases} \infty & \text{if } \ell(s) \neq \ell(t) \\ \max\limits_{s \xrightarrow{w} s'} \min\limits_{t \xrightarrow{w'} t'} \max\{|w - w'|, d^a_\bullet(s',t')\} & \text{otherwise} \end{cases}$$

$$d^r_\bullet(s,t) = \begin{cases} \infty & \text{if } \ell(s) \neq \ell(t) \\ \max\limits_{s \xrightarrow{w} s'} \min\limits_{t \xrightarrow{w'} t'} \max\left\{|\frac{w'-w}{w}|, d^r_\bullet(s',t')\right\} & \text{otherwise} \end{cases}$$

To avoid division by zero, $d^r_\bullet$ is defined only for WTS with strictly positive weights.

Example 5. Consider the WTS in Figure 2b from Example 1. We now want to calculate $d^a_\bullet(s'_1, t'_1)$. Note that $t'_1, t'_3, s'_2, s'_3, s'_4$ have only one outgoing transition.

$$\begin{aligned}
d^a_\bullet(s'_1,t'_1) &= \max\left\{ \begin{array}{l} \max\{|6-4|, d^a_\bullet(s'_2,t'_2)\}, \\ \max\{|4-4|, d^a_\bullet(s'_3,t'_2)\} \end{array} \right\} \\
&= \max\left\{ \begin{array}{l} \max\{|6-4|, \max\{|5-5|, d^a_\bullet(s'_1,t'_1)\}\}, \\ \max\left\{|4-4|, \min\left\{ \begin{array}{l} \max\{|7-9|, d^a_\bullet(s'_4,t'_3)\}, \\ \max\{|7-5|, d^a_\bullet(s'_4,t'_1)\} \end{array} \right\}\right\} \end{array} \right\} \\
&= \max\left\{ \begin{array}{l} \max\{|6-4|, \max\{|5-5|, d^a_\bullet(s'_1,t'_1)\}\}, \\ \max\{|4-4|, \min\{\max\{|7-9|, d^a_\bullet(s'_4,t'_3)\}, \infty\}\} \end{array} \right\} \\
&= \max\left\{ \begin{array}{l} \max\{|6-4|, \max\{|5-5|, d^a_\bullet(s'_1,t'_1)\}\}, \\ \max\{|4-4|, \max\{|7-9|, d^a_\bullet(s'_4,t'_3)\}\} \end{array} \right\} \\
&= \max\left\{ \begin{array}{l} \max\{2, d^a_\bullet(s'_1,t'_1)\}\}, \\ \max\{2, d^a_\bullet(s'_4,t'_3)\}\} \end{array} \right\} \\
&= \max\left\{ \begin{array}{l} \max\{2, d^a_\bullet(s'_1,t'_1)\}\}, \\ \max\{2, \max\{|5-5|, d^a_\bullet(s'_1,t'_1)\}\}\} \end{array} \right\} \\
&= \max\{2, d^a_\bullet(s'_1,t'_1)\}
\end{aligned}$$

This equation has least fixed point $d^a_\bullet(s'_1,t'_1) = 2$. From Example 2 we know that $s'_1 \leq^a_\delta t'_1$ for any $\delta \geq 2$, i.e for any $\delta \geq d^a_\bullet(s'_1,t'_1)$. A similar observation is made using $d^r_\bullet$. Consider Figure 3

from Example 3. Note that s_2, s_3, s_4, t_1, t_3 have only one outgoing transition.

$$
\begin{aligned}
d^r_\bullet(s_1,t_1) &= \max\left\{ \begin{array}{l} \max\left\{\left|\frac{2-1}{1}\right|, d^r_\bullet(s_2,t_2)\right\}, \\ \max\left\{\left|\frac{2-1}{1}\right|, d^r_\bullet(s_3,t_2)\right\} \end{array} \right\} \\
&= \max\left\{ \begin{array}{l} \max\left\{\left|\frac{2-1}{1}\right|, \max\{\left|\frac{4-5}{5}\right|, d^r_\bullet(s_1,t_1)\}\right\}, \\ \max\left\{\left|\frac{2-1}{1}\right|, \min\left\{ \begin{array}{l} \max\{\left|\frac{110-100}{100}\right|, d^r_\bullet(s_4,t_3)\}, \\ \max\{\left|\frac{4-100}{100}\right|, d^r_\bullet(s_4,t_1)\} \end{array} \right\}\right\} \end{array} \right\} \\
&= \max\left\{ \begin{array}{l} \max\left\{\left|\frac{2-1}{1}\right|, \max\{\left|\frac{4-5}{5}\right|, d^r_\bullet(s_1,t_1)\}\right\}, \\ \max\left\{\left|\frac{2-1}{1}\right|, \min\{\max\{\left|\frac{110-100}{100}\right|, d^r_\bullet(s_4,t_3)\}, \infty\}\right\} \end{array} \right\} \\
&= \max\left\{ \begin{array}{l} \max\left\{\left|\frac{2-1}{1}\right|, \max\{\left|\frac{4-5}{5}\right|, d^r_\bullet(s_1,t_1)\}\right\}, \\ \max\left\{\left|\frac{2-1}{1}\right|, \max\{\left|\frac{110-100}{100}\right|, d^r_\bullet(s_4,t_3)\}\right\} \end{array} \right\} \\
&= \max\left\{ \begin{array}{l} \max\left\{1, \frac{1}{5}, d^r_\bullet(s_1,t_1)\}\right\}, \\ \max\left\{1, \frac{1}{10}, d^r_\bullet(s_4,t_3)\}\right\} \end{array} \right\} \\
&= \max\left\{ \begin{array}{l} \max\left\{1, \frac{1}{5}, d^r_\bullet(s_1,t_1)\}\right\}, \\ \max\left\{1, \frac{1}{10}, \left|\frac{5-6}{6}\right|, d^r_\bullet(s_1,t_1)\}\}\right\} \end{array} \right\} \\
&= \max\left\{1, \frac{1}{5}, \frac{1}{6}, \frac{1}{10}, d^r_\bullet(s_1,t_1)\right\} \\
&= \max\left\{1, d^r_\bullet(s_1,t_1)\right\}
\end{aligned}
$$

Note that the second line from below, $\max\left\{1, \frac{1}{5}, \frac{1}{6}, \frac{1}{10}, d^r_\bullet(s_1,t_1)\right\}$, captures exactly the relative differences we need to consider. This equation has least fixed point $d^r_\bullet(s_1,t_1) = 1$. From Example 3 we know that $s_1 \leq^r_\delta t_1$ for any $\delta \geq 1$, i.e for any $\delta \geq d^r_\bullet(s_1,t_1)$.

As the example shows, we can for both simulation preorders define the requirements for a state t to simulate s in terms of the distance from s to t. We now formally prove that this is always the case.

Theorem 1. *For a WTS $\mathcal{S} = (S, \to, \ell)$, two states $s, t \in S$ and $\delta \in \mathbb{Q}_{\geq 0}$:*

$$d^a_\bullet(s,t) \leq \delta \text{ iff } s \leq^a_\delta t$$

Proof. ($\Longrightarrow$) :

For this direction we prove that $R = \{(s,t) \mid d^a_\bullet(s,t) \leq \delta\}$ is a δ-simulation relation. Suppose $(s,t) \in R$ and $s \xrightarrow{w} s'$. This implies directly that $d^a_\bullet(s,t) \leq \delta$ and by the fixed point property of $d^a_\bullet$ that

$$d^a_\bullet(s,t) = \max_{s \xrightarrow{w} s'} \min_{t \xrightarrow{w'} t'} \max\{|w - w'|, d^a_\bullet(s',t')\}.$$

We thus have that there exists a transition $t \xrightarrow{w'} t'$ s.t $|w - w'| \leq \delta$ and $d^a_\bullet(s',t') \leq \delta$. By definition of R we immediately have $s' \leq^a_\delta t'$ and we are done.

($\Longleftarrow$) :

For this direction we let

$$D(s,t) = \left\{ \begin{array}{ll} \delta & \text{if } s \leq^a_\delta t \\ \infty & \text{otherwise} \end{array} \right.$$

We now prove that D is a pre-fixed point to $\boldsymbol{H}$ w.r.t the metric on transition weights $|w - w'|$, i.e $d^a_\bullet \leq D$ as $d^a_\bullet$ is the least pre-fixed point for $\boldsymbol{H}$. Hence, if $s \leq^a_\delta t$ then $d^a_\bullet(s,t) \leq D(s,t) = \delta$.

To show that D is a pre-fixed point we must show that $\boldsymbol{H}(D)(s,t) \leq D(s,t)$ for all $s, t \in S$. For arbitrary $s, t \in S$ we have that if $s \not\leq^a_\delta t$ then $D(s,t) = \infty$ and we are done. If $s \leq^a_\delta t$ then for any transition $s \xrightarrow{w} s'$ there exists a transition $t \xrightarrow{w'} t'$ s.t $|w - w'| \leq \delta$ and $s' \leq^a_\delta t'$. Hence, $D(s',t') = \delta$ and

$$\boldsymbol{H}(D)(s,t) = \max_{s \xrightarrow{w} s'} \min_{t \xrightarrow{w'} t'} \max\{|w - w'|, D(s',t')\} \leq \delta = D(s,t)$$

.

■

Theorem 2. *For a WTS $\mathcal{S} = (S, \to, \ell)$, two states $s, t \in S$ and $\delta \in \mathbb{Q}_{\geq 0}$:*

$$\begin{array}{lll} s \leq^r_\delta t & \textit{iff} & d^r_\bullet(s,t) \leq \delta \\ s \leq t & \textit{iff} & d^a_\bullet(s,t) = 0 \\ s \leq t & \textit{iff} & d^r_\bullet(s,t) = 0 \end{array}$$

Proof. By Theorem 1 and $\leq = \leq^r_0 = \leq^a_0$. ■

We now lift the point-wise distance $d_\bullet$ to the parametric setting.

Definition 11. Let $\mathcal{S} = (S, \to, \ell)$ be a PWTS. $s, t \in S$ two states and $v \in \mathcal{V}$. Given a metric $d : \mathcal{W}^{\mathcal{S}} \times \mathcal{W}^{\mathcal{S}} \longrightarrow \mathbb{Q}_{\geq 0}$, we define the point-wise distance from s to t to be the least fixed point of the following equation

$$d_\bullet(s,t,v) = \begin{cases} \infty & \text{if } \ell(s) \neq \ell(t) \\ \max\limits_{s \xrightarrow{e} s'} \min\limits_{t \xrightarrow{e'} t'} \max\left\{d(v(e), v(e')), d_\bullet(s', t', v)\right\} & \text{otherwise} \end{cases}$$

As for the weighted distance, we instantiate the general point-wise distance to account for absolute and relative difference; $d^a_\bullet(s,t,v)$ and $d^r_\bullet(s,t,v)$. We omit the fixed point iterator as it is essentially the same as for the weighted distance as everything is mapped to the weighted setting by some valuation v. For the same reason, the fixed point is guaranteed after a finite number of steps and we also have that the distance is a hemi-metric.

Example 6. Consider the PWTS in Figure 4 from Example 4. We now want to calculate $d^a_\bullet(s_1, t_1, v)$ for some $v \in \mathcal{V}$:

$$\begin{aligned} d^a_\bullet(s,t,v) &= \min \left\{ \begin{array}{l} \max\{|v(p) - v(p)|), d^a_\bullet(s_2, t_2, v)\} \\ \max\{|v(p) - v(q)|, d^a_\bullet(s_2, t_3, v)\} \end{array} \right\} \\ &= \min \left\{ \begin{array}{l} \max\{|v(p) - v(p)|, \max\{|v(q) - v(r)|, d^a_\bullet(s_3, t_4, v)\}\} \\ \max\{|v(p) - v(q)|, \max\{|v(q) - v(s)|, d^a_\bullet(s_3, t_4, v)\}\} \end{array} \right\} \\ &= \min \left\{ \begin{array}{l} \max\{|v(p) - v(p)|, \max\{|v(q) - v(r)|, 0\}\} \\ \max\{|v(p) - v(q)|, \max\{|v(q) - v(s)|, 0\}\} \end{array} \right\} \\ &= \min\{|v(q) - v(r)|, \max\{|v(p) - v(q)|, |v(q) - v(s)|\}\} \end{aligned}$$

If we are interested in the distance being below some $\delta \in \mathbb{Q}_{\geq 0}$ we have to find the exact constraints on v for $d^a_\bullet(s_1, t_1, v) \leq \delta$ to be true. From the above equation we see these are $v(|q - r|) \leq \delta$ or $[v(|p - q|) \leq \delta$ and $v(|q - s|) \leq \delta]$. Similar constraints were found in Example 4 for $s_1 \leq^a_{\varepsilon, v} t_1$ to be true.

We now formally state the relation between parametric distances and parametric simulations.

Theorem 3. *For a PWTS $\mathcal{S} = (S, \to, \ell)$, two states $s, t \in S$ and $\varepsilon \in \mathcal{E}$:*

- $d^a_\bullet(s,t,v) \leq v(\varepsilon)$ *iff* $s \leq^a_{\varepsilon, v} t$
- $d^r_\bullet(s,t,v) \leq v(\varepsilon)$ *iff* $s \leq^r_{\varepsilon, v} t$
- $d^a_\bullet(s,t,v) = 0$ *iff* $s \leq t$
- $d^r_\bullet(s,t,v) = 0$ *iff* $s \leq t$

Proof. Immediate by Theorem 1-2. ■

We can now restate the problems proposed at the end of the previous section by using the relation between the simulation preorders and the distances.

1. Decide whether or not a valuation v exists such that $d^a_\bullet(s,t,v) \leq v(\varepsilon)$ or $d^r_\bullet(s,t,v) \leq v(\varepsilon)$.
2. Synthesize a valuation v such that $d^a_\bullet(s,t,v) \leq v(\varepsilon)$ or $d^r_\bullet(s,t,v) \leq v(\varepsilon)$.
3. Characterize the sets of valuations $V_1 = \{v \mid v \in \mathcal{V} \text{ and } d^a_\bullet(s,t,v) \leq v(\varepsilon)\}$ and $V_1 = \{v \mid v \in \mathcal{V} \text{ and } d^r_\bullet(s,t,v) \leq v(\varepsilon)\}$.
4. Synthesize the valuation v that minimizes ε for $d^a_\bullet(s,t,v) \leq v(\varepsilon)$ or $d^r_\bullet(s,t,v) \leq v(\varepsilon)$.

Before solving the problems we consider the logical implications of a distance between states.

5 Logical Implications of Simulation

In this section we show how the similarity of two states in a system has implications on the logical properties of the states involved. This is useful if one wants a component to replace (simulate) another larger component while still conforming to the logical specification with limited deviation in behavior. To this end we define weighted variants of CTL [10] and prove relations between these variants and the similarity of states. We start in the traditional weighted setting with a general logic and proceed by defining subsets useful for our discussion.

5.1 Weighted Computation Tree Logic

We want a logic that is capable of reasoning about quantitative properties of WTS and furthermore parametric properties of PWTS. To reason about these properties we define *Weighted Computation Tree Logic (WCTL)* [8] with both lower and upper bounds on path formula weights. Such bounds restrict the minimum and maximum allowable accumulated weight on a path in the WTS.

Definition 12. The set of WCTL *state formulae* are given by the abstract syntax:

$$\Phi, \Psi ::= tt \mid \mathtt{a} \mid \neg\Phi \mid \Phi \wedge \Psi \mid \Phi \vee \Psi \mid E\varphi \mid A\varphi$$

and the set of WCTL *path formulae* are given by the abstract syntax:

$$\varphi ::= X_B\Phi \mid \Phi U_B \Psi$$

where $\mathtt{a} \in \mathcal{AP}$ and $B = [l, u]$ with $l, u \in \mathbb{Q}$.

Notice that l represents a lower bound on accumulated weights over paths and u represents an upper bound. This allows for (degenerate) interval specification on the path formulae. A side-effect is that $l > b$ may be the case; such formulae will never be satisfiable.

Whether a WCTL formula is *satisfied* by a state s or a path π of some WTS $\mathcal{S}$, is given by the satisfiability relation $\models$. We denote this by $\mathcal{S}, s \models \Phi$ and $\mathcal{S}, \pi \models \varphi$, respectively.

Definition 13. Given a WCTL formula, a WTS $\mathcal{S} = (S, \to, \ell)$, a state $s \in S$ or a path $\pi \in \Pi(\mathcal{S})$ the *satisfiability relation* $\models$ is inductively defined as:

$$
\begin{array}{lll}
\mathcal{S}, s \models tt & & \text{always} \\
\mathcal{S}, s \models \mathtt{a} & \text{iff} & \mathtt{a} \in \ell(s) \\
\mathcal{S}, s \models \neg\Phi & \text{iff} & \text{it is not the case that } \mathcal{S}, s \models \Phi \\
\mathcal{S}, s \models \Phi \wedge \Psi & \text{iff} & \mathcal{S}, s \models \Phi \text{ and } \mathcal{S}, s \models \Psi \\
\mathcal{S}, s \models \Phi \vee \Psi & \text{iff} & \mathcal{S}, s \models \Phi \text{ or } \mathcal{S}, s \models \Psi \\
\mathcal{S}, s \models E\varphi & \text{iff} & \text{there exists } \pi \in \Pi(s), \text{ such that } \mathcal{S}, \pi \models \varphi \\
\mathcal{S}, s \models A\varphi & \text{iff} & \text{for all } \pi \in \Pi(s), \text{ it is the case that } \mathcal{S}, \pi \models \varphi \\
\mathcal{S}, \pi \models X_{[l,u]}\Phi & \text{iff} & |\pi| > 0,\ l \leq \mathcal{AW}_\pi(1) \leq u \text{ and } \mathcal{S}, \pi(1)_s \models \Phi \\
\mathcal{S}, \pi \models \Phi U_{[l,u]}\Psi & \text{iff} & \text{there exists } j \in \mathbb{N} \text{ s.t. } \mathcal{S}, \pi(j)_s \models \Psi \text{ where} \\
 & & l \leq \mathcal{AW}_\pi(j) \leq u \text{ and } \mathcal{S}, \pi(j')_s \models \Phi \text{ for all } j' < j.
\end{array}
$$

Example 7. To understand WCTL better, we return to the coffee machine example from the introduction of the thesis. Consider the WTS $\mathcal{S}$ in Figure 5 representing the advanced beverage machine which the computer scientists consider buying. For this example we assume that the scientists know the exact timing of the new machine. We want to use WCTL to specify requirements to the machine. For instance the scientists want to specify that the total amount of time spend from the point where a beverage is selected until it is done, does not exceed 5200 ms which is the time spend by the old machine. Reversely the scientists does not trust the quality of beverages if the machine is faster than 1000 ms. This can be encoded in the following WCTL formula:

$$A\ tt\ U_{[1000;5200]}\ \mathtt{done}$$

A quick look should convince oneself that

$$\mathcal{S}, start\ \models\ A\ tt\ U_{[1000;5200]}\ \mathtt{done}$$

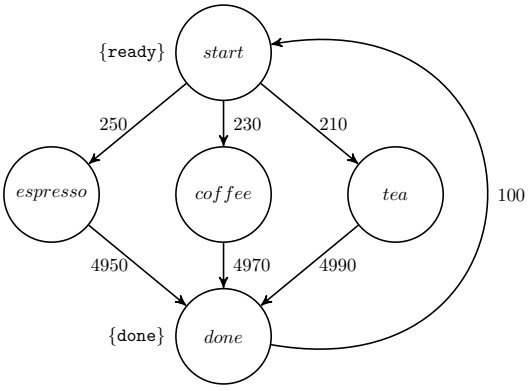


Figure 5: WTS $\mathcal{S}$

WCTL is suitable to reason about all sorts of quantitative properties of the weighted paths but is not powerful enough to take unknown values into account. In the context of the previous example, it would be handy to have the ability to factor in unknown time performance by a machine they have not yet bought. We make this possible by lifting the bounds on the path formula weights from WCTL to encompass parametric expressions from the set

$$\mathcal{E}^- = \{e \mid e = \sum_{i=0}^{n} a_i p_i + b \text{ where } a_i, b \in \mathbb{Q}\}$$

of linear expressions in parameters with rational coefficients. We name the resulting logic *Parametric Weighted CTL (PWCTL)* and we denote the set of all formulae of the logic by PWCTL.

Definition 14. The set of PWCTL *state formulae* are given by the abstract syntax:

$$\Phi, \Psi ::= tt \mid \mathtt{a} \mid \neg\Phi \mid \Phi \wedge \Psi \mid \Phi \vee \Psi \mid E\varphi \mid A\varphi$$

and the set of PWCTL *path formulae* are given by the abstract syntax:

$$\varphi ::= X_B\Psi \mid \Phi U_B \Psi$$

where $\mathtt{a} \in \mathcal{AP}$ and $B = [l, u]$ with $l, u \in \mathcal{E}^-$.

Similar to how we let valuations map PWTS to WTS (Definition 6), we extend valuations to the domain of PWCTL formulae s.t. a PWCTL formula Φ can be instantiated to a WCTL formula given a valuation v.

Definition 15. For a PWCTL formula Φ, and any valuation $v \in \mathcal{V}$, $v(\Phi)$ is defined inductively as follows:

$$\begin{array}{lcl}
v(tt) & \stackrel{\text{def}}{=\!=} & tt \\
v(\mathtt{a}) & \stackrel{\text{def}}{=\!=} & \mathtt{a} \\
v(\neg\Phi) & \stackrel{\text{def}}{=\!=} & \neg v(\Phi) \\
v(\Phi \wedge \Phi') & \stackrel{\text{def}}{=\!=} & v(\Phi) \wedge v(\Phi') \\
v(\Phi \vee \Phi') & \stackrel{\text{def}}{=\!=} & v(\Phi) \vee v(\Phi') \\
v(E\varphi) & \stackrel{\text{def}}{=\!=} & Ev(\varphi) \\
v(A\varphi) & \stackrel{\text{def}}{=\!=} & Av(\varphi) \\
v(X_{[l,u]}\Psi) & \stackrel{\text{def}}{=\!=} & X_{[v(l),v(u)]}v(\Psi) \\
v(\Phi U_{[l,u]}\Psi) & \stackrel{\text{def}}{=\!=} & v(\Phi)U_{[v(l),v(u)]}v(\Psi)
\end{array}$$

The satisfiability of a PWCTL formula relies on the existence of a valuation of the parameters such that we can project the problem to the weighted setting by applying the valuation on the PWTS and PWCTL formula. This problem was addressed in [9] which provided tool support for model-checking of PWCTL on PWTS. For this we extend valuations to paths; let $\pi = s_1e_2s_2e_3s_3...$ be a path with parametric expressions as weights. For arbitrary $v \in \mathcal{V}$ we then define $v(\pi) = s_1v(e_2)s_2(e_3)s_3...$ to be the weighted instance of π w.r.t v. Whether a state or path satisfies a PWCTL formula i now defined by the satisfiability relation $\models_v$.

Definition 16. Given a PWCTL formula, a PWTS $\mathcal{S} = (S, \to, \ell)$, a valuation $v \in \mathcal{V}$ and a state $s \in S$ or a path $\pi \in \Pi(\mathcal{S})$ the *satisfiability relation* $\models_v$ is defined as follows:

$$\mathcal{S}, s \models_v \Phi \text{ iff } v(\mathcal{S}), s \models v(\Phi)$$
$$\mathcal{S}, \pi \models_v \varphi \text{ iff } v(\mathcal{S}), v(\pi) \models v(\varphi)$$

Example 8. Returning to the example with the beverage machine from before, consider now the PWTS $\mathcal{S}'$ in Figure 6. Here the states represent the same as in Figure 5, but the transition weights have parameters associated to represent unknown quantities.

Just as before, the computer scientists want to specify requirements, but as they do not have full knowledge of the advanced beverage machine, they encompass parameters in the PWCTL formula specification:

$$A\ tt\ U_{[1000;5200s]}\ \mathtt{done}$$

By the semantics, we can derive $\mathcal{S}', start' \models_v A\ tt\ U_{[1000;5200s]}$ `done` for any $v \in \mathcal{V}$ s.t

$$1000 \leq 250 + v(p) \leq v(5200s),$$
$$1000 \leq 250 + v(q) \leq v(5200s) \text{ and}$$
$$1000 \leq 250 + v(r) \leq v(5200s).$$

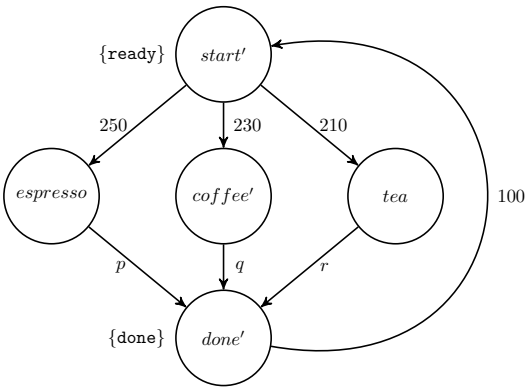


Figure 6: PWTS $\mathcal{S}'$

With PWCTL we are now able to reason about parametric properties of PWTS. Further, we want to relate this ability to the notions of simulation between states proposed in Section 3. Concretely, we want a relation between the PWCTL formulae a PWTS state s satisfies and the formulae a PWTS state t satisfies when $s \leq^a_{\varepsilon,v} t$ or $s \leq^r_{\varepsilon,v} t$. Such results are dependent of a notion of *ε-satisfiability* and likewise we need a notion of *relative ε-satisfiability*. Any definition or result in the parametric context is easily mapped to the weighted context, using the valuation function (Definition 6) and from now on we only consider PWCTL.

Consider the PWTS $\mathcal{S}$ in Figure 7. It is clear that $s_1 \leq^a_{\varepsilon,v} t_1$ for any $v \in \mathcal{V}$ where $v(p) \leq v(\varepsilon)$ and $s_1 \leq^r_{\varepsilon,v} t_1$ for any $v \in \mathcal{V}$ where $\frac{1}{5} \leq v(\varepsilon')$. Obviously $\mathcal{S}, s_1 \models_v EX_{[5p;5p]}\mathtt{a}$ whereas $\mathcal{S}, t_1 \not\models_v EX_{[5p;5p]}\mathtt{a}$. However if we allow the same inaccuracy in t's satisfiability of the formula as the inaccuracy represented by ε in $\leq^a_{\varepsilon,v}$ and $\leq^r_{\varepsilon,v}$, we can create a new formula that is satisfied by t_1. The inaccuracies can be seen in the bounds of the following two formulae:

$$\mathcal{S}, t_1 \models_v EX_{[5p-\varepsilon,5p+\varepsilon]}\mathtt{a}$$
$$\mathcal{S}, t_1 \models_v EX_{[5p\cdot(1-\varepsilon'),5p\cdot(1+\varepsilon')]}\mathtt{a}$$

The bounds in the first of the two formulae represent the inaccuracy ε given by the $s_1 \leq^a_{\varepsilon,v} t_1$ relation. Similarly, the bounds in the second formula represnts the inaccuracy ε' given by the $s_1 \leq^r_{\varepsilon,v} t_1$ relation.

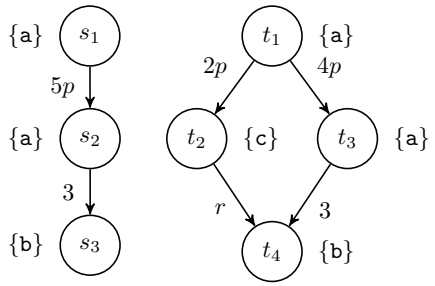


Figure 7: PWTS $\mathcal{S}$

In each of the formulae satisfied by t_1 in the above discussion, ε and ε' have important roles which we can use in the definition of a new satisfiability relation $\models^{\varepsilon}_v$ where a deviation of ε

is allowed in the bounds on path formulae. The deviation can be either absolute or relative. These two approaches are the topics of the following sections, where we consider fragments of PWCTL and relate these to (relative) ε-simulation.

5.2 Relation Between ε-simulation and PWCTL

In this section we consider a subset of PWCTL, called $\mathcal{L}_{abs}$. For a PWTS $\mathcal{S} = (S, \to, \ell)$ and $s, t \in S$ we show how $s \leq^a_{\varepsilon,v} t$ and the satisfaction of formulae by s implies the satisfaction of formulae similar to the formula satisfied by s. Concretely, we intend to loosen the bounds by exactly ε. We start by excluding some of the operators from PWCTL to arrive at a logic with the desired properties. If we once again consider PWTS $\mathcal{S}$ in Figure 7 and the relation $s \leq^a_{\varepsilon,v} t$ it is clear that

$$\mathcal{S}, s_1 \models_v AX_{[5p,5p]}tt \text{ and}$$
$$\mathcal{S}, t_t \not\models_v AX_{[5p-\varepsilon,5p+\varepsilon]}tt.$$

Therefore we will not include universal path formulae in $\mathcal{L}_{abs}$. For until formulae, it is problematic to give semantics for ε-deviated until path formulae as the bounds on an until formula does not restrict the minimal or maximal weight expression on one transition but restrict the accumulated weight of all transitions on a path. Therefore the until operator will not be included in $\mathcal{L}_{abs}$. Moreover we can see in Figure 7 that

$$\mathcal{S}, s_1 \models_v \neg EX_{[2p,4p]}tt \text{ while}$$
$$\mathcal{S}, t_1 \not\models_v \neg EX_{[2p,4p]}tt,$$

which is why we will exclude the negation operator from $\mathcal{L}_{abs}$ as well. Note that we could include negated atomic propositions $\neg a$ in $\mathcal{L}_{abs}$ but one can easily add one extra atomic proposition to the set $\mathcal{AP}$ for each existing atomic proposition, representing the negated proposition.

We now formally define the refined logic $\mathcal{L}_{abs}$ before turning to the relation with $\leq^a_{\varepsilon,v}$.

Definition 17. $\mathcal{L}_{abs}$ denotes the set of all formulae given by the following abstract syntax:

$$\Phi, \Psi ::= tt \mid \mathtt{a} \mid \Phi \wedge \Psi \mid \Phi \vee \Psi \mid EX_B\Psi,$$

where $a \in \mathcal{AP}$ and $B = [l, u]$ with $l, u \in \mathcal{E}^-$.

The satisfiability relation $\models_v$ defined for PWCTL also gives us the semantics of $\mathcal{L}_{abs}$. As discussed above, we need a formalism to specify deviations in $\mathcal{L}_{abs}$. The syntax of such deviated formulae, $\Phi^\varepsilon \in \mathcal{L}_{abs}$, is defined as follows:

Definition 18. For any $\Phi, \Psi \in \mathcal{L}_{abs}$ and an absolute deviation $\varepsilon \in \mathcal{E}$, Φ^ε is defined inductively as follows:

$$\begin{aligned}
tt^\varepsilon &\stackrel{\text{def}}{=\!=} tt \\
\mathtt{a}^\varepsilon &\stackrel{\text{def}}{=\!=} \mathtt{a} \\
(\Phi \wedge \Psi)^\varepsilon &\stackrel{\text{def}}{=\!=} \Phi^\varepsilon \wedge \Psi^\varepsilon \\
(\Phi \vee \Psi)^\varepsilon &\stackrel{\text{def}}{=\!=} \Phi^\varepsilon \vee \Psi^\varepsilon \\
(EX_{[l,u]}\Psi)^\varepsilon &\stackrel{\text{def}}{=\!=} EX_{[l-\varepsilon,u+\varepsilon]}\Psi^\varepsilon
\end{aligned}$$

The subtraction from the lower bound and the addition to the upper bound of the formula allows the PWTS state t to satisfy all the same next formulae as s with a deviation of ε when $s \leq^a_{\varepsilon,v} t$.

We now define the ε-satisfiability relation, $\models^\varepsilon_v$, for $\mathcal{L}_{abs}$ that allows for deviations of ε in the bounds of formulae.

Definition 19. Given a $\mathcal{L}_{abs}$ formula Φ, a PWTS $\mathcal{S} = (S, \to, \ell)$, a state $s \in S$, a valuation $v \in \mathcal{V}$, and an expression $\varepsilon \in \mathcal{E}$, the *$\varepsilon$-satisfiability relation* $\models_v^\varepsilon$ is defined as

$$\mathcal{S}, t \models_v^\varepsilon \Phi \quad \stackrel{\text{def}}{=\!=} \quad \mathcal{S}, s \models_v \Phi^\varepsilon$$

With the syntax and semantics and the ε-deviation of $\mathcal{L}_{abs}$ in place, we are now able to formally state the relation between $\leq_{\varepsilon,v}^a$ and $\models_v^\varepsilon$ for $\mathcal{L}_{abs}$:

Theorem 4. *Given a PWTS $\mathcal{S} = (S, \to, \ell)$, states $s, t \in S$, an arbitrary valuation $v \in \mathcal{V}$ and expression $\varepsilon \in \mathcal{E}$,*

$$s \leq_{\varepsilon,v}^a t \quad \textit{iff} \quad \forall \Phi \in \mathcal{L}_{abs} : [\mathcal{S}, s \models_v \Phi \implies \mathcal{S}, t \models_v^\varepsilon \Phi]$$

Proof. ($\implies$) :

The proof is done inductively over the structure of $\mathcal{L}_{abs}$. The arguments for cases $\Phi = tt, \Phi = \mathtt{a}, \Phi = \Phi' \wedge \Psi$ and $\Phi = \Phi' \vee \Psi$ are trivial. We will therefore only argue for the case $\Phi = EX_B\Psi$.

Assume $s \leq_{\varepsilon,v}^a t$ and $\mathcal{S}, s \models_v EX_B\Psi$ where $B = [l, u]$. Then by the semantics of PWCTL (Definition 16) we know there exists a state s' s.t. $s \xrightarrow{e} s'$ where $v(l) \leq v(e) \leq v(u)$ and $\mathcal{S}, s' \models_v \Psi$. By $s \leq_{\varepsilon,v}^a t$ (Definition 8) we know that for any s' s.t. $s \xrightarrow{e} s'$ there exists a t' s.t. $t \xrightarrow{e'} t'$, $|v(e) - v(e')| \leq v(\varepsilon)$ and $s' \leq_{\varepsilon,v}^a t'$. By induction $\mathcal{S}, s' \models_v \Psi$ implies $\mathcal{S}, t' \models_v^\varepsilon \Psi$ which in turn implies that $\mathcal{S}, t \models_v EX_{[v(l)-v(\varepsilon), v(u)+v(\varepsilon)]}\Psi$. By definition of ε-deviation and the ε-satisfiability relation, $\models_v^\varepsilon$, we can thus conclude that $\mathcal{S}, t \models_v^\varepsilon EX_B\Psi$

($\impliedby$) :

For this direction we show that the relation

$$R = \{(s, t) \mid s, t \in S \text{ and } \forall \Phi \in \mathcal{L}_{abs} : [\mathcal{S}, s \models_v \Phi \implies \mathcal{S}, t \models_v^\varepsilon \Phi]\}$$

is an ε-simulation relation. Let $(s', t') \in R$ and $s' \xrightarrow{w} s''$. Suppose towards a contradiction that $\neg\exists t' \xrightarrow{w'} t'' : |v(w) - v(w')| \leq v(\varepsilon)$ and $(s'', t'') \in R$, i.e $\forall t' \xrightarrow{w'} t'' : |v(w) - v(w')| > v(\varepsilon)$ or $(s'', t'') \notin R$. $|v(w) - v(w')| > v(\varepsilon)$ implies that $S, s' \models_v EX_{[w,w]}tt$ and $S, t' \not\models_v^\varepsilon EX_{[w,w]}tt$, leading to a contradiction as $(s', t') \in R$. $(s'', t'') \notin R$ implies that $\forall t'' : t' \xrightarrow{w'} t''$ there exists a formula $\Phi'' \in \mathcal{L}_{abs}$ such that $S, s'' \models_v \Phi''$ but $S, t'' \not\models_v^\varepsilon \Phi''$. As the transition relation is finite, we can enumerate all such formulae $\Phi'', \Phi_1'', \Phi_2'' \ldots \Phi_n''$ and create the following combined formula:

$$\Phi' = EX_{[w,w]}[\Phi_1'' \wedge \Phi_2'' \wedge \Phi_3'' \ldots \Phi_n'']$$

We now have that $S, s' \models_v \Phi'$ but $S, t' \not\models_v^\varepsilon \Phi'$, leading to a contradiction as $(s', t') \in R$. The assumption that $\neg\exists t' \xrightarrow{w'} t'' : |v(w) - v(w')| \leq v(\varepsilon)$ and $(s'', t'') \in R$ must therefore be wrong and we conclude that $s \leq_{\varepsilon,v}^a t$.

■

Corollary 1. *Given a PWTS $\mathcal{S} = (S, \to, \ell)$, states $s, t \in S$, valuation $v \in \mathcal{V}$ and expression $\varepsilon \in \mathcal{E}$,*

$$s \leq t \;\; \textit{and} \;\; t \leq s \quad \textit{iff} \quad \forall \Phi \in \mathcal{L}_{abs} : [\mathcal{S}, s \models_v \Phi \iff \mathcal{S}, t \models_v \Phi]$$

Proof. Immediate by Theorem 4 and $\leq_0^a = \leq$. ■

5.3 Relation Between Relative ε-simulation and PWCTL

We will now turn to the relation between $\leq_{\varepsilon,v}^r$ and a new subset of PWCTL called $\mathcal{L}_{rel}$. For the same reasons as for $\mathcal{L}_{abs}$ stated above, $\mathcal{L}_{rel}$ will not include the negation state operator and the universal path operator. As opposed to $\mathcal{L}_{abs}$, $\mathcal{L}_{rel}$ includes the until operator and is defined as follows:

Definition 20. $\mathcal{L}_{rel}$ denotes the set of all formulae given by the following abstract syntax:

$$\Phi, \Psi ::= tt \mid \mathtt{a} \mid \Phi \wedge \Psi \mid \Phi \vee \Psi \mid EX_B\Psi \mid E\Phi U_B\Psi,$$

where $a \in \mathcal{AP}$ and $B = [l, u]$ with $l, u \in \mathcal{E}^-$.

The satisfiability relation $\models_v$ defined for PWCTL also gives us the semantics of $\mathcal{L}_{rel}$. To obtain the possibility for ε-satisfiability of $\mathcal{L}_{abs}$ we subtracted ε from lower bounds and added ε to upper bounds on the next operator. In $\mathcal{L}_{rel}$, with similar intentions, we use ε as a factor on lower and upper bounds on the next and until path operators. Using the relative deviation captured by ε in $\leq^r_{\varepsilon,v}$ is what allows us to easily consider deviated until formulae. If we again consider the PWTS $\mathcal{S}$ Figure 7, the idea behind the use of until formulae with a relative deviation on bounds is visible in the following example formulae:

$$\mathcal{S}, s_1 \models_v E\mathtt{a}U_{[5p,5p]}\mathtt{b}$$
$$\mathcal{S}, t_1 \models_v E\mathtt{a}U_{[5p\cdot(1-\varepsilon),5p\cdot(1+\varepsilon)]}\mathtt{b}$$

If t deviates by e.g 10% in each step when trying to simulate s, the accumulated weight over a possible matching path must also deviate by 10%. We therefore extend the bounds in both directions by exactly 10%.

Example 9. Consider the PWTS $\mathcal{S}$ in Figure 8. We have two deviations if t_1 was to simulate s_1; 10% and 100%. Clearly $\mathcal{S}, s_1 \models_v E\mathtt{a}U_{[101,101]}\mathtt{b}$ however $\mathcal{S}, t_1 \not\models_v E\mathtt{a}U_{[101,101]}\mathtt{b}$. If we modify the bounds by extending them by the maximal point-wise relative deviation (100%), we get

$$\mathcal{S}, t_1 \models_v E\mathtt{a}U_{[0,202]}\mathtt{b}$$

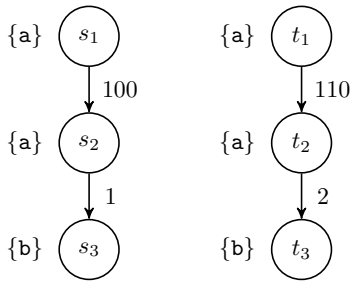


Figure 8: PWTS $\mathcal{S}$

We now define relative deviation in $\mathcal{L}_{rel}$ formulae:

Definition 21. For any $\Phi, \Psi \in \mathcal{L}_{rel}$ and a relative deviation $\varepsilon \in \mathcal{E}$, Φ^ε is defined inductively as follows:

$$\begin{aligned}
tt^\varepsilon &\stackrel{\text{def}}{=} tt \\
\mathtt{a}^\varepsilon &\stackrel{\text{def}}{=} \mathtt{a} \\
(\Phi \wedge \Psi)^\varepsilon &\stackrel{\text{def}}{=} \Phi^\varepsilon \wedge \Psi^\varepsilon \\
(\Phi \vee \Psi)^\varepsilon &\stackrel{\text{def}}{=} \Phi^\varepsilon \vee \Psi^\varepsilon \\
(EX_{[l,u]}\Psi)^\varepsilon &\stackrel{\text{def}}{=} EX_{[l\cdot(1-\varepsilon),u\cdot(1+\varepsilon)]}\Psi^\varepsilon \\
(E\Phi U_{[l,u]}\Psi)^\varepsilon &\stackrel{\text{def}}{=} E\Phi^\varepsilon U_{[l\cdot(1-\varepsilon),u\cdot(1+\varepsilon)]}\Psi^\varepsilon
\end{aligned}$$

The relative deviated semantics for $\mathcal{L}_{rel}$ is given by the ε-satisfiability relation $\models^\varepsilon_v$, defined as follows.

Definition 22. Given a $\mathcal{L}_{rel}$ formula Φ, a PWTS $\mathcal{S} = (S, \to, \ell)$, a state $s \in S$, a valuation $v \in \mathcal{V}$ and some $\varepsilon \in \mathcal{E}$, the *$\varepsilon$-satisfiability relation* $\models_v^\varepsilon$ is defined as

$$\mathcal{S}, t \models_v^\varepsilon \Phi \quad \stackrel{\text{def}}{=\!=} \quad \mathcal{S}, s \models_v \Phi^\varepsilon$$

We will now formally show the relation between $\leq_{\varepsilon,v}^r$ and $\models_v^\varepsilon$ for $\mathcal{L}_{rel}$:

Theorem 5. *Given a PWTS $\mathcal{S} = (S, \to, \ell)$, states $s, t \in S$, arbitrary $v \in \mathcal{V}$ and $\varepsilon \in \mathcal{E}$.*

$$s \leq_{\varepsilon,v}^r t \quad \textit{iff} \quad \forall \Phi \in \mathcal{L}_{rel} : [\mathcal{S}, s \models_v \Phi \implies \mathcal{S}, t \models_v^\varepsilon \Phi]$$

Proof. ($\implies$) :

The proof is done inductively over the structure of $\mathcal{L}_{rel}$. The proofs for cases $\Phi = tt, \Phi = \mathtt{a}, \Phi = \Phi' \wedge \Psi$ and $\Phi = \Phi' \vee \Psi$ are trivial and the proof of the case $\Phi = EX_B\Psi$ follows the argument from Theorem 4. We will therefore only prove the case $\Phi = E\Phi' U_B \Psi$:

Assume $s \leq_{\varepsilon,v}^r t$ and $\mathcal{S}, s \models_v E\Phi' U_{[l,u]}\Psi$. Then we know that there exist a path $\pi \in \Pi(s)$ and a $j \in \mathbb{N}$ s.t. $\mathcal{S}, \pi(j)_s \models_v \Psi$ and $v(l) \leq v(AW_\pi(j)) \leq v(u)$. Similarly we know that $\mathcal{S}, \pi(j')_s \models_v \Phi'$ for any $j' < j$.

As $s \leq_{\varepsilon,v}^r t$, we know that the transitions represented by the path π can be relatively matched by transitions from t and for a given successor pair (s', t') we have $s' \leq_{\varepsilon,v}^r t'$. Therefore there must exist a path $\pi' \in \Pi(t)$ such that, for any $i \in \mathbb{N}$, $\pi(i)_s \leq_{\varepsilon,v}^r \pi'(i)_s$ and $\left|\frac{v(\pi'(i)_w) - v(\pi(i)_w)}{v(\pi(i)_w)}\right| \leq v(\varepsilon)$. This means that $v(l)\cdot(1 - v(\varepsilon)) \leq v(AW_{\pi'}(j)) \leq v(u)\cdot(1 + v(\varepsilon))$. By induction we then know that $\mathcal{S}, \pi'(j)_s \models_v^\varepsilon \Psi$ and $\mathcal{S}, \pi'(j')_s \models_v^\varepsilon \Phi'$ for any $j' < j$. We can then conclude that

$$\mathcal{S}, t \models_v^\varepsilon E\Phi' U_{[l,u]}\Psi.$$

($\impliedby$) :

Follows the arguments for $\impliedby$ of Theorem 4.

■

Corollary 2. *Given a PWTS $\mathcal{S} = (S, \to, \ell)$, states $s, t \in S$, an arbitrary valuation $v \in \mathcal{V}$ and an expression $\varepsilon \in \mathcal{E}$,*

$$s \leq t \;\; \textit{and} \;\; t \leq s \quad \textit{iff} \quad \forall \Phi \in \mathcal{L}_{rel} : [\mathcal{S}, s \models_v \Phi \iff \mathcal{S}, t \models_v \Phi]$$

Proof. Immediate by Theorem 5 and $\leq_0^r = \leq$. ■

6 Distance Checking

In this section we present formalisms for analysis of point-wise distances between two states of either a WTS or a PWTS using fixed point computations on cost assignments to the nodes of *Symbolic Dependency Graphs* (SDG) and their parametrized counterpart *Parametric Symbolic Dependency Graphs* (PSDG).

For both formalisms we discuss their semantics using the notion of assignments, which are used to represent the weighted or parametric *cost* for some property to hold.

6.1 Symbolic Dependency Graphs

Dependency Graphs in general are structures used to represent *dependencies*. Such dependencies may arise from a problem with optimal substructure where an aggregation of solutions to sub-problems is the solution to the overall problem. The overall problem then *depends* on the solution to all sub-problems. These dependencies we encode in a *hyper-graph* through the notion of *hyper-edges* where the edges have multiple successors (targets), one for each dependency.

These graphs are then extended to *Symbolic* Dependency Graphs to consider problems with quantitative answers (e.g. price for some problem to be solved) and lifted to *Parametric* Symbolic Dependency Graphs, where the solution to a problem may also depend on unknown values.

For now, we will look to SDGs which extend the traditional notion of dependency graphs by allowing hyper-edges to have non-negative rational weights associated to represent the *cost* of solving a given sub-problem. In addition to hyper-edges, SDGs also have *cover-edges*. A cover edge has an associated *bound* as weight on the edge. This bound will be interpreted as an upper bound constraint on the cost of solving the sub-problem represented by the node below.

We now proceed by defining SDGs formally.

Definition 23. A *Symbolic Dependency Graph* (SDG) is a tuple $\mathcal{G} = (N, H, C)$, where:

- N is a finite set of *nodes*,
- $H \subseteq N \times 2^{\mathbb{Q}_{\geq 0} \times N}$ is a finite set of *hyper-edges*.
- $C \subseteq N \times \mathbb{Q}_{\geq 0} \times N$ is a finite set of *cover-edges*.

Whenever $(n, T) \in H$ we refer to T as the *target-set*. For each $n' \in T$ we refer to n' as a *target* node. We will use $n \overset{w}{\dashrightarrow} n'$ whenever $(n, w, n') \in C$ and $n \to \emptyset$ to denote a hyper-edge from n with an empty target-set. The set of all SDGs will be denoted by $\mathfrak{G}$.

To motivate the use of SDGs and to give an intuition of their strength, we will now consider a small example.

Example 10. For the WTS $\mathcal{S}$ in Figure 9 we want to know whether or not two states $s, t \in S$ are separated by an absolute point-wise distance of at most some positive rational number δ. We see the representation of this problem in the SDG $\mathcal{G}$ depicted in Figure 10. Note that we use the notation $\langle\cdot\rangle$ for SDG nodes. Whenever we use the $\langle\cdot\rangle$ notation, anything written inside the angle-brackets is to be read purely syntactically, e.g. as a name of the node.

The intuition behind the structure of $\mathcal{G}$ is that we want to represent the definition of $d^a_\bullet$. Node **1** represents the constraint $d^a_\bullet(s_1, t_1) \leq \delta$ with a cover-edge weighted with δ. The resulting node **2** represents the problem of finding the value of $d^a_\bullet(s_1, t_1)$. The hyper-edge with two targets (**3,4**) is because **2** is dependent of $\ell(s_1) = \ell(t_1)$ *and* the maximum over s_1's transitions $s_1 \xrightarrow{w} s'_1$. The only transition of s_1 is represented in **3**. For this transition, we want to minimize over t_1's transitions $t_1 \xrightarrow{w'} t'_1$. For each of the two transitions of t_1 we thus make one hyper-edge with two targets (**9,10** and **11,12**). The fact that we have two targets on those represents that we want to take the maximal value between $|w - w'|$ and $d^a_\bullet(s'_1, t'_1)$. However, for this example we see that t_3 has no transitions, meaning that s_1's self-loop, $s_1 \xrightarrow{3} s_1$, cannot be matched. We thus want the node **12** to represent a "bad" value, which is why it has no outgoing edges. t_2 can match s_1's self-loop by the transition $t_2 \xrightarrow{1} t_1$ represented in the hyper-edge from **9** to **13** and **2**. The result of evaluating this SDG is the constraint $2 \leq \delta$. We show how to arrive at this conclusion for SDGs with or without circular dependencies later, using fixed point computations.

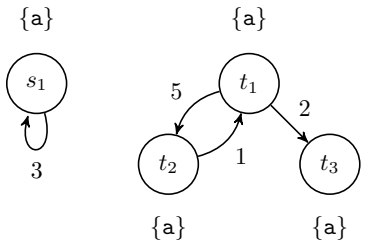


Figure 9: WTS $\mathcal{S} = (S, \to, \ell)$

We provide an in-depth discussion of the general semantics of SDGs as well as the construction of SDGs for distance checking in the following sections.

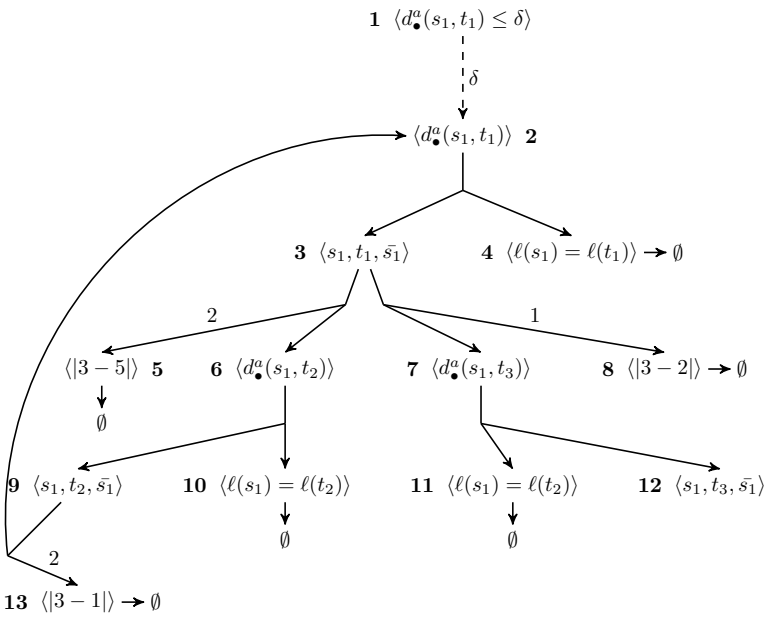


Figure 10: SDG $\mathcal{G} = (N, H, C)$

6.2 Fixed Point Computations on SDGs

We now introduce the semantics of an SDG in the form of cost *assignments* to nodes. These assignments represent the price of solving the problem represented by a given node.

Definition 24. For an SDG $\mathcal{G} = (N, H, C)$, an assignment

$$A : N \longrightarrow \mathbb{Q}_{\geq 0} \cup \{\infty\}$$

on $\mathcal{G}$ is a mapping from each node $n \in N$ to a a positive rational number or ∞.

We denote by $\mathfrak{A}$ the set of all assignments.

Definition 25. The ordering on assignments, $\sqsubseteq$ is defined for arbitrary $A_1, A_2 \in \mathfrak{A}$:

$$A_1 \sqsubseteq A_2 \quad \text{iff} \quad \forall n \in N : A_1(n) \geq A_2(n)$$

The ordering induces a complete lattice, $(\mathfrak{A}, \sqsubseteq)$. We interpret 0 as a "positive" value and ∞ as a "negative" value. The top element is therefore the assignment A^0 which maps to all nodes the value 0, the bottom element is the assignment A^∞ which maps to all nodes the value ∞. The reasoning behind this is that we want to start from a good assumption about the solution to the problem we solve when computing maximal fixed point and reversely for the computation of minimal fixed points.

We are now ready to define the global function, which applied iteratively, updates SDG node assignments. Let $\min\{\emptyset\} = \infty$ and $\max\{\emptyset\} = 0$.

Definition 26. Given an SDG $\mathcal{G} = (N, H, C)$, $\boldsymbol{F} : \mathfrak{A} \longrightarrow \mathfrak{A}$ is a function that, given an assignment A on $\mathcal{G}$, produces a new assignment on $\mathcal{G}$ as follows:

$$\boldsymbol{F}(A)(n) = \begin{cases} \begin{cases} 0 & \text{if } A(n') \leq w \\ \infty & \text{otherwise} \end{cases} & \text{if } n \overset{w}{\dashrightarrow} n' \\ \min\limits_{(n,T) \in H} \max\limits_{(w,n') \in T} \max\{A(n'), w\} & \text{otherwise} \end{cases}$$

It should be clear from the function that SDG nodes without any edge at all will be assigned ∞ and a SDG nodes with one hyper-edge that has no target-set will be assigned 0. This is

essential in the semantics of SDGs, namely that for a node n an empty set of targets in a hyper-edge represents truth and an empty set of hyper-edges represents falsity.

To show that the repeated application of the function $\boldsymbol{F}$ will compute a minimal or maximal fixed point we use Tarski's fixed point theorem [19]. To do so, we show that $\boldsymbol{F}$ is monotone w.r.t. the complete lattice $(\mathfrak{A}, \sqsubseteq)$.

Lemma 1. *The function $\boldsymbol{F}$ is monotone on the complete lattice $(\mathfrak{A}, \sqsubseteq)$.*

Proof. To show that $A \sqsubseteq A'$ implies $\boldsymbol{F}(A) \sqsubseteq \boldsymbol{F}(A')$ for any A and A' in $\mathfrak{A}$, we assume $A^1 \sqsubseteq A^2$ where $A^1, A^2 \in \mathfrak{A}$. Then suppose towards contradiction that $\boldsymbol{F}(A^1) \not\sqsubseteq \boldsymbol{F}(A^2)$ which implies that there exist a SDG node n s.t. $\boldsymbol{F}(A^1)(n) < \boldsymbol{F}(A^2)(n)$. By definition of $\boldsymbol{F}$, this can be true for two different cases:

> n has an outgoing cover-edge $(n \overset{w}{\dashrightarrow} n')$:
> By assumption that $\boldsymbol{F}(A^1)(n) < \boldsymbol{F}(A^2)(n)$ it must be the case that $\boldsymbol{F}(A^1)(n) = 0$ and $\boldsymbol{F}(A^2)(n) = \infty$ which in turn implies that $A^1(n) \leq w < A^2(n)$. This contradicts the assumption that $A^1 \sqsubseteq A^2$.
>
> n has no outgoing cover-edge:
> By assumption that $\boldsymbol{F}(A^1)(n) < \boldsymbol{F}(A^2)(n)$ and by definition of $\boldsymbol{F}$ we get the following contradiction:
>
> $$\min_{(n,T)\in H}\{\max_{(w,n')\in T} \max\{A^1(n'), w\}\} < \min_{(n,T)\in H}\{\max_{(e,n')\in T} \max\{A^2(n'), w\}\}$$

We can thus conclude that $\boldsymbol{F}$ is monotone w.r.t. $(\mathfrak{A}, \sqsubseteq)$. ■

By Tarski's fixed point theorem we know that $\boldsymbol{F}$ computes a maximal or minimal fixed point and thereby the function is indeed theoretically useful. We let the assignments corresponding to these fixed points be denoted by A^{min} and A^{max} and we use $\boldsymbol{F}^i(A)$ to denote i repeated applications of the function $\boldsymbol{F}$ on A, i.e
$\boldsymbol{F}^i(A) = \boldsymbol{F}(\boldsymbol{F}^{i-1}(\dots \boldsymbol{F}^0(A)))$ for $i > 1$ and $\boldsymbol{F}^0(A) = A$.

For $\boldsymbol{F}$ to also be suitable for practical use, we need to make sure that the fixed point is computed in a finite number of repeated applications.

Lemma 2. *$A^{max} = \boldsymbol{F}^i(A^0)$ and $A^{min} = \boldsymbol{F}^j(A^\infty)$ for some $i, j \in \mathbb{N}$.*

Proof. It should be clear from the definition of SDG that a given SDG $\mathcal{G} = (N, H, C)$ has a finite number of nodes $n \in N$ and a finite number of hyper-edges $(n, T) \in H$ and that the number of weights on target-sets $(w, n') \in T$ is also finite. By definition of $\boldsymbol{F}$ (Definition 26) we can thus convince ourselves that the number of repeated applications required to obtain A^{max} and A^{min} is finite regardless of circular dependencies in $\mathcal{G}$ as we can always directly apply the semantics of min and max as all weights are from the set $\mathbb{Q}_{\geq 0}$. ■

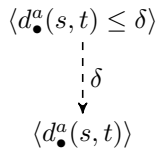
6.3 SDGs for Distance Checking

Now that we have the semantics of SDGs in place through fixed point computations of node assignments, we apply the fixed point function to solve the problem of verifying whether $d^a_\bullet(s,t) \leq \delta$. In the following we will only consider the absolute point-wise distance $d^a_\bullet$ but every method and result presented can be adapted for the relative counterpart $d^r_\bullet$.

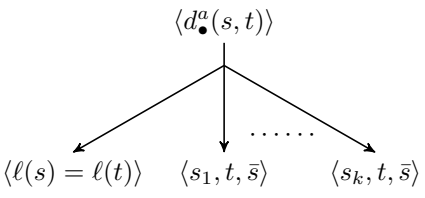
The point-wise absolute distance between states $s, t \in S$ is given by the least fixed point of $d^a_\bullet(s,t)$. Recall that we order all point-wise distances from $\mathfrak{D}$ by $\leq$, which gives us the complete lattice $(\mathfrak{D}, \leq)$. Reversely the ordering over assignments, $\sqsubseteq$, is defined as $A^1 \sqsubseteq A^2$ iff $\forall n \in N : A^1(n) \geq A^2(n)$. Therefore we will represent the problem of finding the least fixed point of $d^a_\bullet(s,t)$ in a SDG where the solution is the maximal fixed point over assignments. For SDG construction we recall the definition of $d^a_\bullet$ for two states s, t from a WTS $\mathcal{S}$:

$$d^a_\bullet(s,t) = \begin{cases} \infty & \text{if } \ell(s) \neq \ell(t) \\ \max\limits_{s \xrightarrow{w} s'} \min\limits_{t \xrightarrow{w'} t'} \max\{|w - w'|, d^a_\bullet(s', t')\} & \text{otherwise} \end{cases}$$

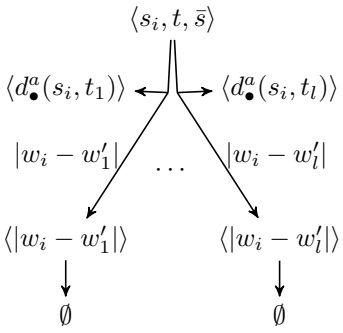
We would like to know whether or not the condition $d^a_\bullet(s,t) \leq \delta$ for some $\delta \in \mathbb{Q}_{\geq 0}$ is satisfied. The construction rules for a SDG with these properties can be found in Figure 11. We let $\boldsymbol{Build}_\bullet(\mathcal{S}, s, t, \delta)$ denote the recursive application of these rules on a WTS $\mathcal{S}$ with states s, t and $\delta \in \mathbb{Q}_{\geq 0}$.



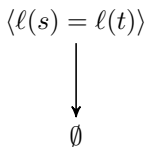
(a) Root node



(b) Maximum over $s \xrightarrow{w} s'$ from $d^a_\bullet(s,t)$



(c) Minimum over $t \xrightarrow{w'} t'$ from $d^a_\bullet(s,t)$



(d) Label matching: $\ell(s) = \ell(t)$

(e) Label matching: $\ell(s) \neq \ell(t)$

Figure 11: Let $\{(w_1, s_1), (w_2, s_2) \dots (w_k, s_k)\} \in \boldsymbol{out}(s)$ and $\{(w'_1, t_1), (w'_2, t_2) \dots (w'_l, t_l)\} \in \boldsymbol{out}(t)$

The root node in Figure 11a represents the upper bound constraint δ on the point-wise distance from s to t. For this we use a cover-edge, labeled with δ. The target node of this cover-edge (Figure 11b) has one hyper-edge with multiple targets. By the semantics of SDG nodes given by the function $\boldsymbol{F}$, this corresponds to a maximum over all target nodes. One of these targets is the node $\langle \ell(s) = \ell(t) \rangle$. If it is the case that $\ell(s) = \ell(t)$, this node will have fixed point assignment 0 and ∞ otherwise. This implies that the fixed point assignment of its source node $\langle d^a_\bullet(s,t) \rangle$ will be ∞ if $\ell(s) \neq \ell(t)$, representing $d^a_\bullet(s,t) = \infty$. Otherwise, we want to consider the maximum over all transitions $s \xrightarrow{w} s'$, and therefore create target nodes for each such transition. Figure 11c then represents a minimum over all transitions $t \xrightarrow{w'} t'$ by constructing a hyper-edge for each transition. Each of these hyper-edges have two targets, one for weight matching and one for recursively checking the distance of pairs of successor

states. By the semantics we thus have that we compute the maximum of the immediate weight difference and the distance between successor pairs, as defined by $d^a_\bullet$. The maximal fixed point assignment of a node on the form $\langle d^a_\bullet(s,t)\rangle$ will in this way represent the least fixed point of $d^a_\bullet(s,t)$.

Before proving this method to be correct we will consider an example.

Example 11. Consider the WTS $\mathcal{S} = (S, \to, \ell)$ in Figure 12. We want to know whether or not $d^a_\bullet(s,t) \leq 10$ for $s,t \in S$. To understand the use of SDGs and assignments we encode this question in the SDG $\mathcal{G} = \boldsymbol{Build}_\bullet(\mathcal{S}, s, t, \delta) = (N, H, C)$ depicted in Figure 13. In Figure 12 we see that the infinite path $s_3\, 2 s_4\, 3 s_3\, 2 s_4 \cdots$ has to be matched by the infinite path $t_2\, 12\, t_3\, 4 t_2\, 12 t_3 \cdots$. The absence of such a match would result in $d^a_\bullet(s,t) = \infty$. The matching of these infinite paths is seen as a loop between nodes **8,12,13,16** in $\mathcal{G}$.

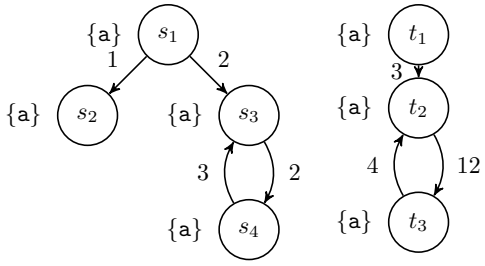


Figure 12: WTS $\mathcal{S}$

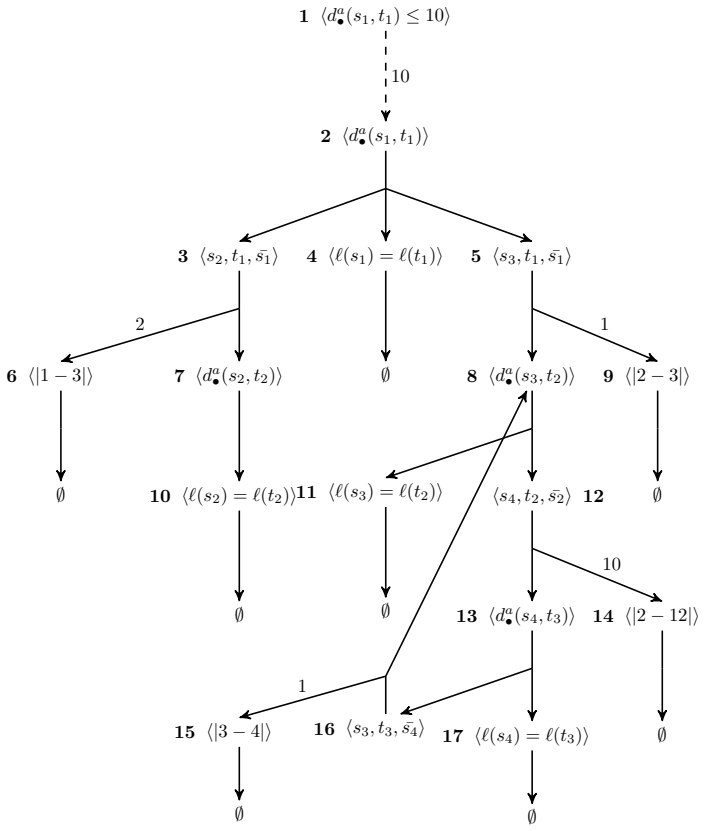


Figure 13: SDG $\mathcal{G} = \boldsymbol{Build}_\bullet(\mathcal{S}, s, t, \delta)$

We now consider Table 1 to see the computation of the maximal fixed point assignments A^{max} for each node $n \in N$ in the SDG $\mathcal{G}$. We let $\boldsymbol{F}^i(n)$ denote $\boldsymbol{F}^i(A^0)(n)$. By looking through

the assignments to each node of each iteration i, it is clear that $\boldsymbol{F}^5(n) = A^{max}(n)$ for all $n \in N$. If we consider $A^{max}(\mathbf{2}) = 10$ we can infer that $d^a_\bullet(s,t) = 10$ and finally $d^a_\bullet(s,t) \leq 10$.

$n, \boldsymbol{F}^i(n)$	$A^0(n)$	$\boldsymbol{F}^1(n)$	$\boldsymbol{F}^2(n)$	$\boldsymbol{F}^3(n)$	$\boldsymbol{F}^4(n)$	$\boldsymbol{F}^5(n)$
1	0	0	0	0	0	0
2	0	0	2	2	10	10
3	0	2	2	2	2	2
4	0	0	0	0	0	0
5	0	1	1	10	10	10
6	0	0	0	0	0	0
7	0	0	0	0	0	0
8	0	0	10	10	10	10
9	0	0	0	0	0	0
10	0	0	0	0	0	0
11	0	0	0	0	0	0
12	0	10	10	10	10	10
13	0	0	1	1	10	10
14	0	0	0	0	0	0
15	0	0	0	0	0	0
16	0	1	1	10	10	10
17	0	0	0	0	0	0

Table 1: Updating assignments to the nodes of SDG $\mathcal{G}$ in Figure 13

Now that we have seen an example of $\boldsymbol{F}$ we turn to a formal proof of the correctness of our method of verifying $d^a_\bullet(s,t) \leq \delta$.

Theorem 6. *Let $\mathcal{S} = (S, \to, \ell)$ be a WTS, $s, t \in S$ and $\mathcal{G} = \boldsymbol{Build}_\bullet(S, s, t, \delta) = (N, H, C)$ be a SDG. For arbitrary $\delta \in \mathbb{Q}_{\geq 0}$,*

(1) $A^{max}(\langle d^a_\bullet(s,t) \leq \delta \rangle) = 0 \iff d^a_\bullet(s,t) \leq \delta$

(2) $A^{max}(\langle d^a_\bullet(s,t) \rangle) = d^a_\bullet(s,t)$

Proof. By definition of $\boldsymbol{F}$, (1) follows from (2). For (2) we will show that

(2*a*) $A^{max}(\langle d^a_\bullet(s,t) \rangle) \leq d^a_\bullet(s,t)$

(2*b*) $A^{max}(\langle d^a_\bullet(s,t) \rangle) \geq d^a_\bullet(s,t)$

(2*a*) :

Let $\boldsymbol{a} : N \longrightarrow \mathbb{Q}_{\geq 0} \cup \{\infty\}$ be defined as:

$$\boldsymbol{a}(n) = \begin{cases} \begin{cases} 0 & \text{if } d^a_\bullet(s,t) \leq \delta \\ \infty & \text{otherwise} \end{cases} & \text{if } n = \langle d^a_\bullet(s,t) \leq \delta \rangle \\ d^a_\bullet(s,t) & \text{if } n = \langle d^a_\bullet(s,t) \rangle \\ \min\limits_{(n,T) \in H} \max\limits_{(w,n') \in T} \max\{\boldsymbol{a}(n'), w\} & \text{otherwise} \end{cases}$$

We will show that $\boldsymbol{a}$ is a post-fixed point of $\boldsymbol{F}$, i.e.

$$\boldsymbol{a} \sqsubseteq \boldsymbol{F}(\boldsymbol{a}).$$

We consider the case $n = \langle d^a_\bullet(s', t') \rangle$ for some $s', t' \in S$. We thus want to show that $\boldsymbol{a}(n) \geq \boldsymbol{F}(\boldsymbol{a})(n)$.

This is trivially done for $\boldsymbol{a}(n) = \infty$. Otherwise by definition of $\boldsymbol{F}$ we can rewrite $\boldsymbol{F}(\boldsymbol{a})(n)$ as follows:

$$\boldsymbol{F}(\boldsymbol{a})(n) = \min_{(n,T) \in H} \max_{(w_i, n_i) \in T} \max\{\boldsymbol{a}(n_i), w_i\}$$

By definition of $\boldsymbol{Build}_\bullet$ we know that there is only one pair $(n, T) \in H$. and for any w_i in pairs $(w_i, n_i) \in T$, $w_i = 0$. For each transition $s' \xrightarrow{w} s''$ there is one target pair in T. Furthermore there is one target-pair where the node n_i represents the proposition $\ell(s') = \ell(t')$. The assignment for this node will always be 0 for the case where $\boldsymbol{a}(n) \neq \infty$. Using this knowledge about H and T we can thus write:

$$\boldsymbol{F}(\boldsymbol{a})(n) = \max_{(w_i,n_i)\in T} \{\boldsymbol{a}(n_i)\}$$

and by expanding $A(n_i)$ using $\boldsymbol{F}$ and $\boldsymbol{a}$ we know that

$$\boldsymbol{F}(\boldsymbol{a})(n) = \max_{(w_i,n_i)\in T} \min_{(n_i,T_i)\in H} \max_{(w_j,n_j)\in T_i} \max\{\boldsymbol{a}(n_j), w_j\}.$$

By $\boldsymbol{Build}_\bullet$ we know that each target-set $T_i \in H$ has only two target-pairs denoted as $T_i = \{(w_j, n_j), (w'_j, n'_j)\}$ and hence we know that

$$\boldsymbol{F}(\boldsymbol{a})(n) = \max_{(w_i,n_i)\in T} \min_{(n_i,T_i)\in H} \max\left\{\boldsymbol{a}(n_j), w_j, \boldsymbol{a}(n'_j), w'_j\right\}.$$

By $\boldsymbol{Build}_\bullet$ we know that for any transition $t' \xrightarrow{w'} t''$, $n_j = \langle d^a_\bullet(s'', t'')\rangle$, $w_j = 0$, $n'_j = \langle |w - w'|\rangle$ and $w'_j = |w - w'|$. Using $\boldsymbol{F}$ and $\boldsymbol{a}$ thus get:

$$\boldsymbol{F}(\boldsymbol{a})(n) = \max_{(w_i,n_i)\in T} \min_{(n_i,T_i)\in H} \max\{\boldsymbol{a}(n_j), |w - w'|\}.$$

As defined in $\boldsymbol{Build}_\bullet$ we know that for all transitions $s' \xrightarrow{w} s''$ there is a node n_i in a pair $(w_i, n_i) \in T$ which has a hyper-edge $(n_i, T_i) \in H$ for each possible match $t' \xrightarrow{w'} t''$. We can thus exchange the maximum over target-pairs of T with a maximum over s' transitions and the minimum over hyper-edges from n_i with a minimum over t' transitions in the definition of $F(\boldsymbol{a})(n)$ as follows:

$$\boldsymbol{F}(\boldsymbol{a})(n) = \max_{s' \xrightarrow{w} s''} \min_{t' \xrightarrow{w'} t''} \max\{d^a_\bullet(s'', t''), |w - w'|\}.$$

Clearly $\boldsymbol{a}(n) \geq \boldsymbol{F}(\boldsymbol{a})(n)$ and we can conclude $A^{max}(\langle d^a_\bullet(s, t)\rangle) \leq d^a_\bullet(s, t)$.

(2b) :

Let $D(s, t) = A^{max}(\langle d^a_\bullet(s, t)\rangle)$ for all $s, t \in S$ s.t. $\langle d^a_\bullet(s, t)\rangle \in N$. Let $n = \langle d^a_\bullet(s', t')\rangle \in N$ and $s', t \in S$. We will now show that D is a pre-fixed point for $\boldsymbol{H}$ i.e $d^a_\bullet(s', t') \leq D(s', t')$. As $D(s', t') = A^{max}(\langle d^a_\bullet(s', t')\rangle)$ we can rewrite $D(s', t')$ using arguments similar to what we did for $\boldsymbol{F}(\boldsymbol{a})(n)$ in (2a). We then get that

$$D(s', t') = \max_{(w_i,n_i)\in T} \min_{(n_i,T_i)\in H} \max\left\{A^{max}(n_j), w_j, A^{max}(n'_j), w'_j\right\}.$$

which can be simplified (by the same arguments as in (2a) to

$$D(s', t') = \max_{(w_i,n_i)\in T} \min_{(n_i,T_i)\in H} \max\{d^a_\bullet(s'', t''), |w - w'|\}.$$

and finally

$$D(s', t') = \max_{s' \xrightarrow{w} s''} \min_{t' \xrightarrow{w'} t''} \max\{d^a_\bullet(s'', t''), |w - w'|\}.$$

Thus $d^a_\bullet(s', t') \leq D(s', t')$ as desired.

By (2a) and (2b) we can conclude that $A^{max}(\langle d^a_\bullet(s, t)\rangle) = d^a_\bullet(s, t)$. ■

6.4 Parametric Symbolic Dependency Graphs

To lift the results regarding SDGs from the rational weighted setting to the parametric setting we use the notion of Parametric Symbolic Dependency Graphs (PSDGs), first introduced in [9]. In PSDGs, hyper-edges and cover-edges have parametric expressions as weights. The

expressions used as weights on hyper-edges are on the form $|e_1 - e_2|$ where $e_1, e_2 \in \mathcal{E}$ and the set of such expressions will be denoted by $\mathcal{E}^a$. Cover-edge weights are drawn from the set $\mathcal{E}$.

Definition 27. A *Parametric Symbolic Dependency Graph* (PSDG) is a tuple $\mathcal{G}^{\mathcal{P}} = (N, H, C)$, where:

- N is a finite set of *nodes*,
- $H \subseteq N \times 2^{\mathcal{E}^a \times N}$ is a finite set of *hyper-edges*.
- $C \subseteq N \times \mathcal{E} \times N$ is a finite set of *cover-edges*.

The set of all PSDGs will be denoted by $\mathfrak{G}^{\mathcal{P}}$. For PSDG nodes and edges we use the same notation as for those of SDGs.

6.5 Fixed Point Computations on PSDGs

We now discuss derivation of constraints on parameter valuations for the problem of checking whether the distance from a state s to a state t is below a given threshold. Instead of lifting the concept of assignments from the weighted to the parametric context with use of the direct semantic mapping through valuations we start by discussing assignments from a purely syntactical perspective. We first define an abstract syntax used to describe assignments syntactically. If we recall the function $\boldsymbol{F}$, it is useful to have syntax for minimum and maximum expressions. We also need syntax to describe the conditions in the cover-edge case stating that a node n gets assignment 0 if $A(n') \leq w$ and ∞ otherwise for a cover-edge $n \overset{w}{\dashrightarrow} n'$. Finally we would like weight expressions from target-sets of hyper-edges on PSDGs to be represented syntactically. The full abstract syntax for assignment expressions is thus as follows:

$$E_1, E_2 ::= \left[\begin{array}{ll} 0 & \text{if } E_1 \leq e \\ \infty & \text{otherwise} \end{array}\right] \mid e^a \mid \min\{E_1, E_2\} \mid \max\{E_1, E_2\}$$

where $e^a \in \mathcal{E}^a \cup \{\infty, 0\}$ and $e \in \mathcal{E}$. Let the set of assignment expressions be denoted by $\mathbb{E}$ and let $v(\infty) = \infty$ and $v(|e_1 - e_2|) = |v(e_1) - v(e_2)|$ for any $v \in \mathcal{V}$. We can now define the (denotational) semantics of the assignment expressions from $\mathbb{E}$:

Definition 28. For any assignment expression $E \in \mathbb{E}$ the denotation $[\![E]\!] : \mathcal{V} \longrightarrow \mathbb{Q}_{\geq 0}$ is defined inductively as follows:

$$\begin{array}{rcl} \left[\!\!\left[\left[\begin{array}{ll} 0 & \text{if } E_1 \leq e_1 \\ \infty & \text{otherwise} \end{array}\right]\right]\!\!\right](v) & = & \left\{\begin{array}{ll} 0 & \text{if } [\![E_1]\!](v) \leq v(e_1) \\ \infty & \text{otherwise} \end{array}\right. \\ [\![e^a]\!](v) & = & v(e^a) \\ [\![\min\{E_1, E_2\}]\!](v) & = & \min\{[\![E_1]\!](v), [\![E_2]\!](v)\} \\ [\![\max\{E_1, E_2\}]\!](v) & = & \max\{[\![E_1]\!](v), [\![E_2]\!](v)\} \end{array}$$

For syntactic assignment expressions $E \in \mathbb{E}$, we propose normal forms E^{nf} and F^{nf} by the following abstract syntax:

$$\begin{array}{rcl} E^{nf} & ::= & \min\left\{\max\left\{F^{nf}_{1.1} \cdots F^{nf}_{1.k}\right\} \cdots \max\left\{F^{nf}_{k.1} \cdots F^{nf}_{k.l}\right\}\right\} \\ F^{nf} & ::= & \left[\begin{array}{ll} 0 & \text{if } E^{nf} \leq e \\ \infty & \text{otherwise} \end{array}\right] \mid e^a \end{array}$$

where $e^a \in \mathcal{E}^a \cup \{\infty, 0\}$ and $e \in \mathcal{E}$. The set of all assignment expressions on such a normal form is denoted by $\mathbb{E}^{nf}$. We now define a function $\boldsymbol{nf}$ that converts an expression into an expression on normal form, based on the fact that, for any $v \in \mathcal{V}$:

$$[\![\max\{\min\{E_1, E_2\}, E_3\}]\!](v) = [\![\min\{\max\{E_1, E_3\}, \max\{E_2, E_3\}\}]\!](v)$$

Definition 29. Let $\boldsymbol{nf} : \mathbb{E} \longrightarrow \mathbb{E}$ be a function converting any assignment expression of $\mathbb{E}$ to normal form:

$$\begin{array}{lcl}
\boldsymbol{nf}\,(e^a) & = & e^a \\
\boldsymbol{nf}\left(\left[\begin{array}{ll} 0 & \text{if } E_1 \leq e_1 \\ \infty & \text{otherwise} \end{array}\right]\right) & = & \left[\begin{array}{ll} 0 & \text{if } \boldsymbol{nf}(E_1) \leq e_1 \\ \infty & \text{otherwise} \end{array}\right] \\
\boldsymbol{nf}\,(\min\{E_1, E_2\}) & = & \min\,\{\,\boldsymbol{nf}(E_1), \boldsymbol{nf}(E_2)\,\} \\
\boldsymbol{nf}\,(\max\{\min\{E_1, E_2\}, E_3\}) & = & \min\left\{\begin{array}{l} \max\{\boldsymbol{nf}(E_1), \boldsymbol{nf}(E_3)\}, \\ \max\{\boldsymbol{nf}(E_2), \boldsymbol{nf}(E_3)\} \end{array}\right\}
\end{array}$$

We are now ready to define the mapping of PSDG nodes to assignment expressions:

Definition 30. An assignment

$$A_{\mathbb{E}} : N \longrightarrow \mathbb{E}^{nf}$$

on a PSDG $\mathcal{G}^{\mathcal{P}} = (N, H, C)$ is a mapping from each node $n \in N$ to an assignment expression.

We denote the set of all assignments $\mathfrak{A}_{\mathbb{E}}$. For distance checking we introduce both syntactic and semantic fixed points on assignments by ordering assignments both syntactically and semantically. We start by ordering assignment expressions.

Definition 31. The syntactic ordering on assignment expressions, $\leq_{nf}$ is defined inductively for any $E^{nf} \in \mathbb{E}^{nf}$ as follows:

$$\begin{array}{rcll}
e_1^a \leq_{nf} e_2^a & & \text{iff} & e_1^a = e_2^a \\
\left[\begin{array}{ll} 0 & \text{if } E_1^{nf} \leq e \\ \infty & \text{otherwise} \end{array}\right] & \leq_{nf} \left[\begin{array}{ll} 0 & \text{if } E_2^{nf} \leq e \\ \infty & \text{otherwise} \end{array}\right] & \text{iff} & E_1^{nf} \leq_{nf} E_2^{nf} \text{ and} \\
\min\{E_{1.1}^{nf} \cdots E_{1.k}^{nf}\} & \leq_{nf} \min\{E_{2.1}^{nf} \cdots E_{2.l}^{nf}\} & \text{iff} & \forall j \exists i \text{ s.t. } E_{1.i}^{nf} \leq_{nf} E_{2.j}^{nf} \\
\max\{E_{1.1}^{nf} \cdots E_{1.k}^{nf}\} & \leq_{nf} \max\{E_{2.1}^{nf} \cdots E_{2.l}^{nf}\} & \text{iff} & \forall i \exists j \text{ s.t. } E_{1.i}^{nf} \leq_{nf} E_{2.j}^{nf}
\end{array}$$

We can now order assignments syntactically:

Definition 32. The *syntactic* ordering on assignments, $\sqsubseteq_{\mathbb{E}}$ is defined for any $A_{\mathbb{E}}^1, A_{\mathbb{E}}^2 \in \mathfrak{A}_{\mathbb{E}}$:

$$A_{\mathbb{E}}^1 \sqsubseteq_{\mathbb{E}} A_{\mathbb{E}}^2 \quad \text{iff} \quad \forall n \in N\; A_{\mathbb{E}}^2(n) \leq_{nf} A_{\mathbb{E}}^1(n)$$

For the semantic ordering we use the assignment expression denotations:

Definition 33. The *semantic* ordering on assignments, $\sqsubseteq_{[\![\mathbb{E}]\!]}$ is defined for any $A_{\mathbb{E}}^1, A_{\mathbb{E}}^2 \in \mathfrak{A}_{\mathbb{E}}$:

$$A_{\mathbb{E}}^1 \sqsubseteq_{[\![\mathbb{E}]\!]} A_{\mathbb{E}}^2 \quad \text{iff} \quad \forall n \in N\, \forall v \in \mathcal{V} : [\![A_{\mathbb{E}}^1]\!](v) \geq [\![A_{\mathbb{E}}^2]\!](v)$$

Lemma 3. *For arbitrary* $A_{\mathbb{E}}^1, A_{\mathbb{E}}^2 \in \mathfrak{A}_{\mathbb{E}}$*:*

$$A_{\mathbb{E}}^1 \sqsubseteq_{\mathbb{E}} A_{\mathbb{E}}^2 \implies A_{[\![\mathbb{E}]\!]}^1 \sqsubseteq_{[\![\mathbb{E}]\!]} A_{[\![\mathbb{E}]\!]}^2$$

Proof. Clear by syntactic ordering of min/max expression in Definition 31. ■

We denote by $A_{\mathbb{E}}^0$ and $A_{\mathbb{E}}^\infty$ the mapping of 0 and ∞ to each node in a PSDG. In order to compute a fixed point over assignments we use a modified version of the function $\boldsymbol{F}$ (Definition 26):

Definition 34. Given a PSDG $\mathcal{G}^{\mathcal{P}} = (N, H, C)$, $\boldsymbol{F}_{\mathbb{E}} : \mathfrak{A}_{\mathbb{E}} \longrightarrow \mathfrak{A}_{\mathbb{E}}$ is a function that, given a syntactic assignment $A_{\mathbb{E}}$ on $\mathcal{G}^{\mathcal{P}}$, produces a new assignment on $\mathcal{G}^{\mathcal{P}}$ as follows:

$$[\![\boldsymbol{F}_{\mathbb{E}}(A_{\mathbb{E}})(n)]\!](v) = \begin{cases}
\left\{\begin{array}{ll} 0 & \text{if } [\![A_{\mathbb{E}}(n')]\!](v) \leq v(e) \\ \infty & \text{otherwise} \end{array}\right. & \text{if } n \xrightarrow{\text{e}} n' \\
\min\limits_{(n,T)\in H} \max\limits_{(e^a,n')\in T} \max\{[\![A_{\mathbb{E}}(n')]\!](v), [\![e^a]\!](v)\} & \text{otherwise}
\end{cases}$$

Similar to the notation for $\boldsymbol{F}$ in Section 6.2 we denote by $\boldsymbol{F}^i_{\mathbb{E}}(A_{\mathbb{E}})$ the i'th application of $\boldsymbol{F}_{\mathbb{E}}$ on $A_{\mathbb{E}}$.

For any valuation $v \in \mathcal{V}$, the function is equivalent to the one defined for SDGs (Definition 26). We can thus use the same argument as in Lemma 1 for $\boldsymbol{F}_{\mathbb{E}}$ to be monotone on $(\mathfrak{A}_{\mathbb{E}}, \sqsubseteq_{[\![\mathbb{E}]\!]})$ w.r.t some $v \in \mathcal{V}$, meaning that we again can use Tarski's fixed point theorem [19] to state that the maximal and minimal fixed point are computable by repeated application of $\boldsymbol{F}_{\mathbb{E}}$. Similar to the notation in Section 6.2 we use $A^{min}_{\mathbb{E}}$ and $A^{max}_{\mathbb{E}}$ to denote minimal and maximal fixed point assignments (by the semantic ordering) to a PSDG. We then get that $A^{max}_{\mathbb{E}}$ and $A^{min}_{\mathbb{E}}$ can be computed in a finite number of applications of $\boldsymbol{F}_{\mathbb{E}}$.

Syntactically, $\boldsymbol{F}_{\mathbb{E}}(A_{\mathbb{E}})(n)$ represents an assignment expression E to the node n. Given a valuation v, we can use the denotational semantics to compute a number in $\mathbb{Q}_{\geq 0}$. For our method to be practically applicable we have to make sure that $\boldsymbol{F}_{\mathbb{E}}$, when viewed from a syntactic perspective, also computes a fixed point in a finite number of steps. To do this, we use the syntactic ordering on expression on normal form from the set $\mathbb{E}^{nf}$.

Theorem 7. *Let $\mathcal{G} = (N, H, C)$ be a PSDG. Then there exists natural numbers $i, j \in \mathbb{N}$ s.t:*

1. $\boldsymbol{F}^i_{\mathbb{E}}(A^0_{\mathbb{E}}) \sqsubseteq_{\mathbb{E}} \boldsymbol{F}^{i-1}_{\mathbb{E}}(A^0_{\mathbb{E}})$ *and* $\boldsymbol{F}^{i-1}_{\mathbb{E}}(A^0_{\mathbb{E}}) \sqsubseteq_{\mathbb{E}} \boldsymbol{F}^{i}_{\mathbb{E}}(A^0_{\mathbb{E}})$
2. $\boldsymbol{F}^j_{\mathbb{E}}(A^\infty_{\mathbb{E}}) \sqsubseteq_{\mathbb{E}} \boldsymbol{F}^{j-1}_{\mathbb{E}}(A^\infty_{\mathbb{E}})$ *and* $\boldsymbol{F}^{j-1}_{\mathbb{E}}(A^\infty_{\mathbb{E}}) \sqsubseteq_{\mathbb{E}} \boldsymbol{F}^{j}_{\mathbb{E}}(A^0_{\mathbb{E}})$

Proof. For the proof of (1) we have to show that by starting with the assignment $A^0_{\mathbb{E}}$, the function $\boldsymbol{F}_{\mathbb{E}}$ computes a syntactic fixed point in a finite number of steps. As the PSDG is finite, we have a finite number of syntactic expressions on the form $|e_1 - e_2|$ on the edges of a given PSDG. Furthermore, all assignment expressions are on normal form. Thus, if the function $\boldsymbol{F}$ does not find a fixed point after a finite number of steps, it implies that we can keep finding new expressions on the form $|e_1 - e_2|$ which is a contradiction. (2) follows similar arguments used for (1). ■

6.6 PSDGs for Distance Checking

In this section we discuss the application of PSDGs for synthesis of parameter constrains characterizing the valuations v s.t $d^a_\bullet(s,t,v) \leq v(\varepsilon)$ for some $\varepsilon \in \mathcal{E}$. The construction rules for creating PSDGs suitable for checking the point-wise distance between pairs of PWTS states, are similar to the rules for weighted distance checking in Figure 11. Note that we use the syntax $\langle d^a_\bullet(s,t) \leq \varepsilon \rangle$ instead of $\langle d^a_\bullet(s,t,v) \leq v(\varepsilon) \rangle$. This is done to represent the fact that we want to deduce the characterization of parameter valuations instead of simply applying one specific valuation v to instantiate everything to the weighted setting. For some PWTS $\mathcal{S} = (S, \rightarrow, \ell)$ and $s, t \in S$ we denote by $\boldsymbol{Build}^{\mathcal{P}}_\bullet(\mathcal{S}, s, t, \varepsilon)$ the PSDG $\mathcal{G}^{\mathcal{P}}$ generated for $d^a_\bullet(s,t,v) \leq v(\varepsilon)$. We thus solve problem 3 of the proposed problems in Section 4:

> 3 Characterize the sets of valuations $V_1 = \{v \,|, v \in \mathcal{V}$ and $d^a_\bullet(s,t,v) \leq v(\varepsilon)\}$ and $V_1 = \{v \,|, v \in \mathcal{V}$ and $d^r_\bullet(s,t,v) \leq v(\varepsilon)\}$

Example 12. In this example we consider a parametric version of Example 10. Consider the PWTS in Figure 14 and the PSDG in Figure 15. The assignments after each iteration is shown in Table 2, Table 3 and Table 4. We let $\boldsymbol{F}^i(n)$ denote $\boldsymbol{F}^i(A^0_{\mathbb{E}})(n)$ and remove duplicate elements along with elements dominated by ∞. The exact constraints are given by the fixed point assignment to node **1** as follows:

$$min\{max\{|p-s|, |p-q|, 0\}, max\{|p-q|, |p-s|, |p-r|, 0\}\} \leq \varepsilon$$

As the elements of the first max are also elements of the second max, we can simplify the above expression to get:

$$max\{|p-s|, |p-q|, 0\} \leq \varepsilon$$

From this we get the exact constraint on parameters, characterizing the set of valuations we seek.

$$|p-s| \leq \varepsilon \wedge |p-q| \leq \varepsilon$$

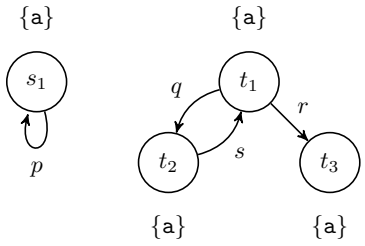


Figure 14: PWTS $\mathcal{S}$

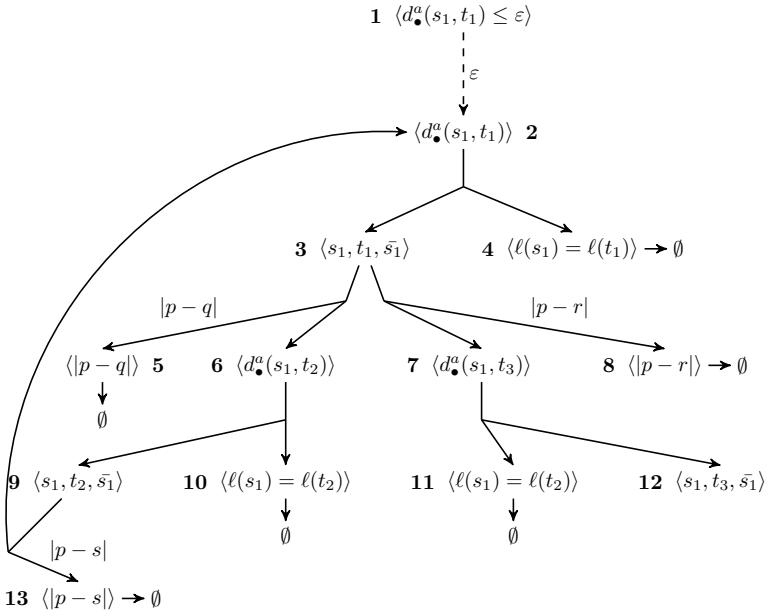


Figure 15: PSDG $\mathcal{G}^{\mathcal{P}}$

Given a valuation $v \in \mathcal{V}$ the correctness of this method for parametric distance checking is given directly by the proof for the weighted counterpart in Theorem 6 and we can thus state the following:

Theorem 8. *Let $\mathcal{S} = (S, \to, \ell)$ be a PWTS, $s, t \in S$ states and $\mathcal{G}^{\mathcal{P}} = \boldsymbol{Build}^{\mathcal{P}}_\bullet(\mathcal{S}, s, t, \varepsilon) = (N, H, C)$ be a PSDG. For arbitrary $\varepsilon \in \mathcal{E}$ and $v \in \mathcal{V}$:*

(1) $[\![A^{max}_{\mathbb{E}}(\langle d^a_\bullet(s,t) \leq \varepsilon\rangle)]\!](v) = 0 \iff d^a_\bullet(s,t,v) \leq v(\varepsilon)$

(2) $[\![A^{max}_{\mathbb{E}}(\langle d^a_\bullet(s,t)\rangle)]\!](v) = d^a_\bullet(s,t,v)$

Proof. Immediate by Theorem 6. ■

We thus have by Lemma 3 that the syntactic fixed point implies a semantic fixed point, which is guaranteed to exist by Theorem 7. Finally, this fixed point is guaranteed to be a solution to $d^a_\bullet(s,t,v) \leq v(\varepsilon)$ by Theorem 8. In the next section we describe how the syntactic fixed point algorithm is implemented in a tool.

n	$A^0(n)$	$\boldsymbol{F}^1_{\mathbb{E}}(n)$	$\boldsymbol{F}^2_{\mathbb{E}}(n)$	$\boldsymbol{F}^3_{\mathbb{E}}(n)$
1	0	$\begin{bmatrix} 0 & \text{if } \boldsymbol{F}^1(\mathbf{2}) \leq \varepsilon \\ \infty & \text{otherwise} \end{bmatrix}$	$\begin{bmatrix} 0 & \text{if } \boldsymbol{F}^2(\mathbf{2}) \leq \varepsilon \\ \infty & \text{otherwise} \end{bmatrix}$	$\begin{bmatrix} 0 & \text{if } \boldsymbol{F}^3(\mathbf{2}) \leq \varepsilon \\ \infty & \text{otherwise} \end{bmatrix}$
2	0	$min\{max\{0\}\}$	$min\{max\{\lvert p-q\rvert, 0\}, max\{\lvert p-r\rvert, 0\}\}$	$min\{max\{\lvert p-q\rvert, 0\}, max\{\lvert p-r\rvert, 0\}\}$
3	0	$min\{max\{\lvert p-q\rvert, 0\}, max\{\lvert p-r\rvert, 0\}\}$	$min\{max\{\lvert p-q\rvert, 0\}, max\{\lvert p-r\rvert, 0\}\}$	$min\{max\{\lvert p-q\rvert, \lvert p-s\rvert, 0\}\}$
4	0	0	0	0
5	0	$min\{max\{0\}\}$	$min\{max\{\lvert p-s\rvert, 0\}\}$	$min\{max\{\lvert p-s\rvert, 0\}\}$
6	0	0	0	0
7	0	$min\{max\{0\}\}$	$min\{max\{\infty\}\}$	$min\{max\{\infty\}\}$
8	0	0	0	0
9	0	$min\{max\{\lvert p-s\rvert, 0\}\}$	$min\{max\{\lvert p-s\rvert, 0\}\}$	$min\{max\{\lvert p-s\rvert, \lvert p-q\rvert, 0\}, max\{\lvert p-s\rvert, \lvert p-r\rvert, 0\}\}$
10	0	0	0	0
11	0	0	0	0
12	0	∞	∞	∞
13	0	0	0	0

Table 2: Iterations 1-3 of assignment updates on $\mathcal{G}^{\mathcal{P}}$ from Figure 15

n	$\boldsymbol{F}^4_{\mathbb{E}}(n)$	$\boldsymbol{F}^5_{\mathbb{E}}(n)$
1	$\begin{bmatrix} 0 & \text{if } \boldsymbol{F}^4(\mathbf{2}) \leq \varepsilon \\ \infty & \text{otherwise} \end{bmatrix}$	$\begin{bmatrix} 0 & \text{if } \boldsymbol{F}^5(\mathbf{2}) \leq \varepsilon \\ \infty & \text{otherwise} \end{bmatrix}$
2	$min\{max\{\lvert p-s\rvert, 0\}\}$	$min\{max\{\lvert p-s\rvert, 0\}\}$
3	$min\{max\{\lvert p-s\rvert, 0\}\}$	$min\{max\{\lvert p-s\rvert, \lvert p-q\rvert, 0\}, max\{\lvert p-q\rvert, \lvert p-s\rvert, \lvert p-r\rvert, 0\}\}$
4	0	0
5	$min\{max\{\lvert p-s\rvert, \lvert p-q\rvert, 0\}, max\{\lvert p-q\rvert, \lvert p-s\rvert, \lvert p-r\rvert, 0\}\}$	$min\{max\{\lvert p-s\rvert, \lvert p-q\rvert, 0\}, max\{\lvert p-q\rvert, \lvert p-s\rvert, \lvert p-r\rvert, 0\}\}$
6	0	0
7	$min\{max\{\infty\}\}$	$min\{max\{\infty\}\}$
8	0	0
9	$min\{max\{\lvert p-s\rvert, \lvert p-q\rvert, 0\}, max\{\lvert p-q\rvert, \lvert p-s\rvert, \lvert p-r\rvert, 0\}\}$	$min\{max\{\lvert p-s\rvert, 0\}\}$
10	0	0
11	0	0
12	∞	∞
13	0	0

Table 3: Iterations 4-5 of assignment updates on $\mathcal{G}^{\mathcal{P}}$ from Figure 15

n	$\boldsymbol{F}^6_{\mathbb{E}}(n)$	$\boldsymbol{F}^7_{\mathbb{E}}(n)$
1	$\begin{bmatrix} 0 & \text{if } \boldsymbol{F}^6(\mathbf{2}) \leq \varepsilon \\ \infty & \text{otherwise} \end{bmatrix}$	$\begin{bmatrix} 0 & \text{if } \boldsymbol{F}^7(\mathbf{2}) \leq \varepsilon \\ \infty & \text{otherwise} \end{bmatrix}$
2	$min\{max\{\lvert p-s\rvert, \lvert p-q\rvert, 0\}, max\{\lvert p-q\rvert, \lvert p-s\rvert, \lvert p-r\rvert, 0\}\}$	$min\{max\{\lvert p-s\rvert, \lvert p-q\rvert, 0\}, max\{\lvert p-q\rvert, \lvert p-s\rvert, \lvert p-r\rvert, 0\}\}$
3	$min\{max\{\lvert p-s\rvert, \lvert p-q\rvert, 0\}, max\{\lvert p-q\rvert, \lvert p-s\rvert, \lvert p-r\rvert, 0\}\}$	$min\{max\{\lvert p-s\rvert, \lvert p-q\rvert, 0\}\}$
4	0	0
5	$min\{max\{\lvert p-s\rvert, \lvert p-q\rvert, 0\}\}$	$min\{max\{\lvert p-s\rvert, \lvert p-q\rvert, 0\}\}$
6	0	0
7	$min\{max\{\infty\}\}$	$min\{max\{\infty\}\}$
8	0	0
9	$min\{max\{\lvert p-s\rvert, \lvert p-q\rvert, 0\}\}$	$min\{max\{\lvert p-s\rvert, \lvert p-q\rvert, 0\}, max\{\lvert p-q\rvert, \lvert p-s\rvert, \lvert p-r\rvert, 0\}\}$
10	0	0
11	0	0
12	∞	∞
13	0	0

Table 4: Iterations 5-7 of assignment updates on $\mathcal{G}^{\mathcal{P}}$ from Figure 15

7 Implementation

In this section we describe the tool-support for distance checking. Originally the tool was made for model checking of PWCTL (a variant with only upper bounds on path formulae) on PWTS using a similar PSDG structure as an abstract representation of the problem [9]. We have now extended the tool with the fixed point algorithm, constraint generation and valuation synthesis for point-wise absolute and relative distance checking between PWTS states presented in this thesis. Furthermore, the tool allows for optimization objectives enabling the possibility to compute the minimal distance possible under the synthesized constraints on parameters as well as generating a valuation w.r.t the constraints. Constraint representation and manipulation is done using $Z3$, a high-end theorem prover by Microsoft Research [11] while the optimization objectives are implemented using a new branch of $Z3$ called νZ [5]. The tool can therefore be used to solve all problems (1-4) proposed in Section 4.

7.1 Example Usage

In Figure 16 we see the simulation checking tab of the tool where the model can be specified along with the pair of states for which the user want to check the distance. In this case the states chosen are s_1 and s_2. The model specification itself is done using a very simple syntax; the statement

$$s_1 := \{a\} < p > s_2 + < q > s_3$$

is a declaration of a state s_1 with atomic proposition a and two transitions with weights p and q; one to s_2 and one to s_3, respectively. Note that only natural numbers are allowed for parameter coefficients. When a model is specified, it is possible to view a graphical representations of both the PWTS and the PSDG by pressing either `Show PWTS` or `Show PSDG`. The PWTS specified can be seen in Figure 17 (PSDG is omitted as it has more than 50 nodes).

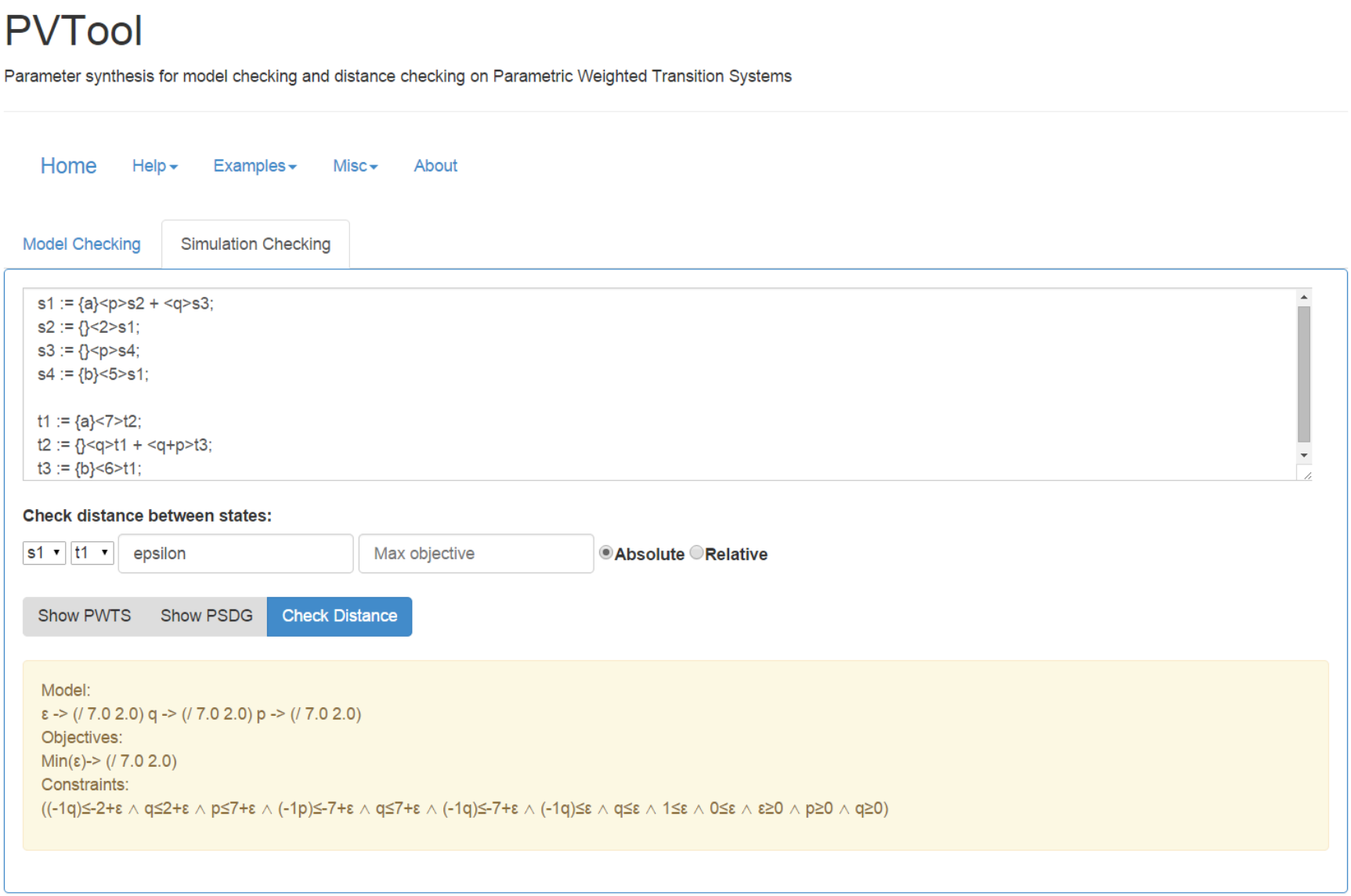


Figure 16: Constraint generation for absolute distance between states

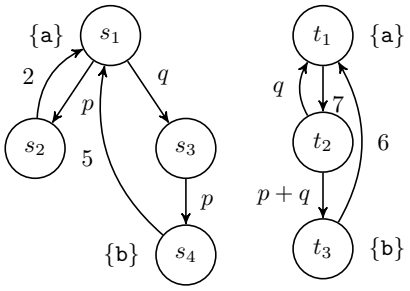


Figure 17: WTS $\mathcal{S}$

The user can also specify minimum and maximum objectives in the two input boxes next to the chosen states. The objectives are linear expressions in parameters with natural coefficients. In the screenshot we see that ε (denoted by "epsilon") has been selected for minimization. ε is a unique parameter in the tool, always representing the bound on the distance between two states. In this example we synthesize constraints on parameters for the absolute point-wise distance between states s_1, t_1; this is done by choosing `Absolute` next to the optimization objectives. We thus want to characterize the valuations v for $d^a_\bullet(s_1, t_1, v) \leq v(\varepsilon)$. The result of the synthesis of valuations is shown in the bottom area of the picture and is contains the following information ((\ 7.0 2.0) should be read as $\frac{7}{2}$):

- Generated model for constraints w.r.t optimization objective. $p \longrightarrow x$ should be read as "the parameter p is evaluated to x.
- Result for optimization objectives e.g $\min(\varepsilon) \longrightarrow \frac{7}{2}$
- Synthesized constraints on parameters.

The minimal possible distance is thus $\frac{7}{2}$, for any $v \in \mathcal{V}$ where $v(p) = v(q) = \frac{7}{2}$. Note that if no optimization objective is chosen, a randomly generated model is presented along with the constraints.

7.2 Manipulating Assignments

In the tool we directly represent the min/max expressions computed by the fixed point algorithm discussed in Section 6.5. As the assignment expressions are not guaranteed to be on normal form after each iteration, we use the rule

$$\max\{\min\{E_1, E_2\}, E_3\} = \min\{\max\{E_1, E_3\}, \max\{E_2, E_3\}\}$$

recursively to normalize them.

We also eliminate any duplicates and any *max* expressions dominated by an ∞ element. Finally, we use the rule

$$\min\{\max\{E_1, E_2\}, \max\{E_1, E_2, E_2\}\} = \min\{\max\{E_1, E_2\}\}$$

to further reduce the size of assignments. In this way we keep the size of assignments low at all times, greatly reducing the amount of time needed for assignment updates. It also speeds up the syntactic fixed point checking which is based on the syntactic ordering of assignments presented in Definition 31.

When the fixed point is found we consider the assignment to the root node. As this is always a cover-node we impose the bound constraint on the assignment to the target of the cover-edge. We do this using the following constraint generation rule:

$$\begin{aligned}&\min\{\max\{E_1, E_2\}, \max\{E_3, E_4\}, \ldots, \max\{E_{n-1}, E_n\}\} \leq \varepsilon = \\ &(E_1 \leq \varepsilon \wedge E_2 \leq \varepsilon) \vee (E_3 \leq \varepsilon \wedge E_4 \leq \varepsilon) \vee \ldots \vee (E_{n-1} \leq \varepsilon \wedge E_n \leq \varepsilon)\end{aligned}$$

7.3 $Z3$ integration

To work with parameter constraints we have integrated our tool with $Z3$, a high-end theorem prover developed at Microsoft Research [11]. The integration consists of:

1. Representation of constraints directly as $Z3$ abstract syntax trees.
2. Simplifying constraints.
3. Checking whether a set of constraints has a model and generate a random model.
4. Optimizing a given goal with respect to parameter constraints and provide a model.

For constraint representation we use the theory of linear real arithmetic. We convert the assignment to constraints as described in the previous section and add the constraint $p \geq 0$ for any parameter p present in the model. This is to ensure positivity of parameter valuations when looking for a model of the constraints. If no model is found, $false$ is the output. Otherwise a valid model is generated and presented to the user along with simplified constraints. The simplification option used is the most efficient and therefore there may in some cases be redundant information. This is however rarely a problem since the assignments on which the constraints are based, are normalized and simplified before constraint generation.

For the problem of verifying $d^a_\bullet(s,t,v) \leq \varepsilon$ or $d^r_\bullet(s,t,v) \leq \varepsilon$, we get as an answer a set of constraints involving the parameters of the PWTS in which s,t are states and the unique parameter ε. It is then interesting to find a model which has the property that it assigns to ε the minimal value possible, without violating the constraints. As shown in Section 5, this has consequences on the logical properties of s and t and is therefore of great interest.

To find such a minimal model (w.r.t ε) we use a new branch of $Z3$ called νZ [5], capable of solving optimization problems modulus theories. Concretely, we allow for both minimization and maximization objectives on the form $a_0 \cdot p_0 + a_1 \cdot p_1 \ldots a_n \cdot p_n$ where each p_i is a parameter from the PWTS under consideration (or the unique parameter ε), and each a_i is a natural number. Each objective is evaluated independently and the minimal and maximal values are generated along with parameter constraints and a model.

8 Conclusion

In this work we have provided formalisms to reason about similarities in behavior of Parametric Weighted Transition Systems. This is done using extensions to the traditional notion of weighted simulation where small deviations of ε in the matching of transitions weights is allowed. This naturally leads to a distance between states, capturing the point-wise deviation allowed by the error term ε.

To examine the logical properties of ε-similarity we provide a discussion on the relations between a parametrized weighted CTL, called PWCTL, and ε-similarity. To this end we define the notion of ε-satisfiability, capturing the error ε directly in formulae. For PWTS states s,t we then show the direct correlation between the ε-similarity of s,t and t's ε-satisfaction of formulae satisfied by s.

Finally we present Parametric Symbolic Dependency Graphs (PSDGs), used to model parametric quantitative dependencies between problems, and fixed point computation on these. The algorithm can be used to both find minimal and maximal fixed points on these graphs by globally updating cost assignments to nodes. The point-wise semantics as well as the finite PSDG structure then ensures that a fixed point is found after a finite number of steps. We then show how the distance checking problem can be characterized as a maximal fixed point on PSDGs specifically created for distance checking. Our method has been implemented in a working tool, available online with a web-based front-end.

In the end of Section 4 we stated four problems which we wanted our methods to solve:

1. Decide whether or not a valuation v exists such that $d^a_\bullet(s,t,v) \leq v(\varepsilon)$ or $d^r_\bullet(s,t,v) \leq v(\varepsilon)$.

2. Synthesize a valuation v such that $d^a_\bullet(s,t,v) \leq v(\varepsilon)$ or $d^r_\bullet(s,t,v) \leq v(\varepsilon)$.

3. Characterize the sets of valuations $V_1 = \{v \mid v \in \mathcal{V} \text{ and } d^a_\bullet(s,t,v) \leq v(\varepsilon)\}$ and $V_1 = \{v \mid v \in \mathcal{V} \text{ and } d^r_\bullet(s,t,v) \leq v(\varepsilon)\}$.

4. Synthesize the valuation v that minimizes ε for $d^a_\bullet(s,t,v) \leq v(\varepsilon)$ or $d^r_\bullet(s,t,v) \leq v(\varepsilon)$.

Each of the four problems have been solved through the implementation of our tool. The combination of tool support for parametric distance checking and the relation between PWCTL and ε-similarity provides a strong framework for reasoning about correctness of Parametric Weighted Transition Systems.

For future work, multiple possibilities exist. One obvious choice is to consider accumulating distances in addition to point-wise distances; in this case the fixed point algorithm of [9] for model-checking PWTS w.r.t PWCTL with only upper bounds may be useful as it is capable of capturing the accumulating nature of path formula satisfiability. For both point-wise and accumulating distances it would also be interesting to add the possibility of more than one *type* of weight on transitions, or distances between atomic propositions such that e.g propositions `coffee` and `espresso` are "close" to each other. Finally, from a practical point of view, an on-the-fly approach and general optimizations would significantly increase the applicability of the proposed techniques for solving complex real world problems.

References

[1] Rajeev Alur and David L. Dill. Automata for modeling real-time systems. In *ICALP*, pages 322–335, 1990.

[2] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. Parametric real-time reasoning. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*, pages 592–601, 1993.

[3] Gerd Behrmann, Ansgar Fehnker, Thomas Hune, Kim Guldstrand Larsen, Paul Pettersson, Judi Romijn, and Frits W. Vaandrager. Minimum-cost reachability for priced timed automata. In *Hybrid Systems: Computation and Control, 4th International Workshop, HSCC 2001, Rome, Italy, March 28-30, 2001, Proceedings*, pages 147–161, 2001.

[4] Nikola Benes, Peter Bezdek, Kim G. Larsen, and Jirí Srba. Language emptiness of continuous-time parametric timed automata. *CoRR*, abs/1504.07838, 2015.

[5] Nikolaj Bjørner, Anh-Dung Phan, and Lars Fleckenstein. νz-an optimizing smt solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 194–199. Springer, 2015.

[6] Patricia Bouyer, Ulrich Fahrenberg, Kim Guldstrand Larsen, Nicolas Markey, and Jirí Srba. Infinite runs in weighted timed automata with energy constraints. In *Formal Modeling and Analysis of Timed Systems, 6th International Conference, FORMATS 2008, Saint Malo, France, September 15-17, 2008. Proceedings*, pages 33–47, 2008.

[7] Marius Bozga, Conrado Daws, Oded Maler, Alfredo Olivero, Stavros Tripakis, and Sergio Yovine. Kronos: A model-checking tool for real-time systems. In *Computer Aided Verification, 10th International Conference, CAV '98, Vancouver, BC, Canada, June 28 - July 2, 1998, Proceedings*, pages 546–550, 1998.

[8] Thomas Brihaye, Véronique Bruyere, and Jean-François Raskin. Model-checking for weighted timed automata. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, pages 277–292. Springer, 2004.

[9] Peter Christoffersen, Mikkel Hansen, Anders Mariegaard, Julian Trier Ringsmose, Kim Guldstrand Larsen, and Radu Mardare. Parametric verification of weighted systems. *http://lipn.univ-paris13.fr/SynCoP2015/Parametric%20Verification%20of%20Weighted%20Systems%20DRAFT.pdf*, To Appear.

[10] Edmund M Clarke and E Allen Emerson. *Design and synthesis of synchronization skeletons using branching time temporal logic*. Springer, 1982.

[11] Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient smt solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340. Springer, 2008.

[12] U. Fahrenberg. A quantitative characterization of weighted kripke structures in temporal logic. 29:1311–, 2012.

[13] Uli Fahrenberg, Kim G Larsen, and Claus Thrane. Metrics for weighted transition systems: Axiomatization and complexity. *Theoretical Computer Science*, 412(28):3358–3369, 2011.

[14] Thomas A. Henzinger. The theory of hybrid automata. In *Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, New Brunswick, New Jersey, USA, July 27-30, 1996*, pages 278–292, 1996.

[15] Thomas A. Henzinger and Howard Wong-Toi. Using hytech to synthesize control parameters for a steam boiler. In Jean-Raymond Abrial, Egon Börger, and Hans Langmaack, editors, *Formal Methods for Industrial Applications*, volume 1165 of *Lecture Notes in Computer Science*, pages 265–282. Springer Berlin Heidelberg, 1996.

[16] Jonas Finnemann Jensen, Kim Guldstrand Larsen, Jiří Srba, and Lars Kaerlund Oestergaard. Local model checking of weighted ctl with upper-bound constraints. In *Model Checking Software*, pages 178–195. Springer, 2013.

[17] Kim G. Larsen, Paul Pettersson, and Wang Yi. Uppaal in a nutshell. *International Journal on Software Tools for Technology Transfer*, 1(1-2):134–152, 1997.

[18] Xinxin Liu and Scott A. Smolka. Simple linear-time algorithms for minimal fixed points. In *Automata, Languages and Programming*, pages 53–66. Springer, 1998.

[19] Alfred Tarski et al. A lattice-theoretical fixpoint theorem and its applications. *Pacific journal of Mathematics*, 5(2):285–309, 1955.

A Resume

This thesis addresses the problem of parameter synthesis for simulation distance checking on parametric weighted transition systems. The traditional weighted transition systems are parametrized by adding linear expressions in parameters as transition weights to allow for specification of unknown or unspecified quantitative behavior. For interpretation of parametric values we introduce a function that maps all parameters to a positive rational value, thus making it possible to map a parametric weighted transition system to an infinite set of traditional weighted transition systems.

We then extend the usual notion of simulation relations by allowing point-wise deviations in transition weight matching, inducing a directed point-wise distance between states of both weighted transition systems and their parametrized counterpart. This simulation-distance is characterized by a least fixed point to a set of equations and is naturally a point-wise distance between pairs of states of (parametric) weighted transition systems.

Furthermore we show how the point-wise distance between states of parametric weighted transition systems have implications to the logical properties of systems by considering a parametric extension of weighted CTL. Concretely, we show that if a set of parametric weighted CTL formulae is satisfied by a state s which is simulated by another state t, the simulating state t satisfies a related set of formulae. For this related set of formulae, the bounds on path formulae have been stretched to capture the error in weight matching, characterized by the point-wise distance between s and t.

As the main contribution of our thesis we consider the problem of characterizing constraints on parameter valuations for the distance between states to be below a specified threshold. To do this we introduce Parametric Symbolic Dependency Graphs suitable for simulation-distance checking, and provide a fixed-point theory for cost assignments to nodes of these graphs. Termination of this method is guaranteed using Tarski's fixed point theorem for complete lattices.

Finally we discuss a web-based implementation with functionality to synthesize the exact constraints on parameters for the point-wise distance between two states to be below a certain threshold. The tool can also be used to generate a model (parameter valuation) for the constraints and applying optimization objectives w.r.t the constraints to e.g find the minimal possible distance between systems. This functionality has been added with the help from $Z3$ which is a high-end theorem prover developed by Microsoft and νZ which is an extensions to $Z3$.