

Computing Branching Distances Using Quantitative Games

Uli Fahrenberg¹, Axel Legay², and Karin Quaas³

¹ École polytechnique, Palaiseau, France

² Université catholique de Louvain, Belgium

³ Universität Leipzig, Germany

Abstract. We lay out a general method for computing branching distances between labeled transition systems. We translate the quantitative games used for defining these distances to other, path-building games which are amenable to methods from the theory of quantitative games. We then show for all common types of branching distances how the resulting path-building games can be solved. In the end, we achieve a method which can be used to compute all branching distances in the linear-time-branching-time spectrum.

Keywords: Quantitative verification, branching distance, quantitative game, path-building game

1 Introduction

During the last decade, formal verification has seen a trend towards modeling and analyzing systems which contain quantitative information. This is motivated by applications in real-time systems, hybrid systems, embedded systems and others. Quantitative information can thus be a variety of things: probabilities, time, tank pressure, energy intake, *etc.*

A number of quantitative models have hence been developed: probabilistic automata [43], stochastic process algebras [36], timed automata [2], hybrid automata [1], timed variants of Petri nets [30, 42], continuous-time Markov chains [44], *etc.* Similarly, there is a number of specification formalisms for expressing quantitative properties: timed computation tree logic [35], probabilistic computation tree logic [31], metric temporal logic [37], stochastic continuous logic [3], *etc.*

Quantitative verification, *i.e.*, the checking of quantitative properties for quantitative systems, has also seen rapid development: for probabilistic systems in PRISM [38] and PEPA [27], for real-time systems in Uppaal [40], RED [50], TAPAAL [5] and Romeo [26], and for hybrid systems in HyTech [33], SpaceEx [25] and HySAT [24], to name but a few.

Quantitative verification has, however, a problem of *robustness*. When the answers to model checking problems are Boolean—either a system meets its

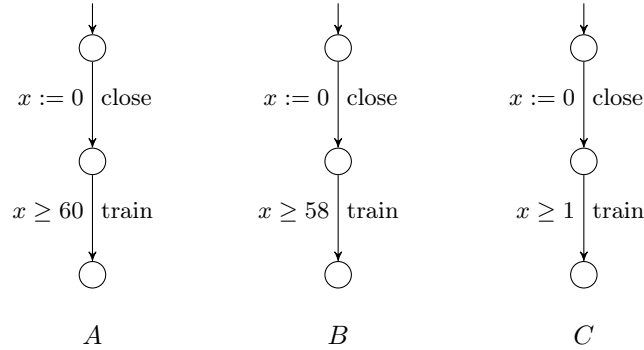


Fig. 1. Three timed automata modeling a train crossing.

specification or it does not—then small perturbations in the system’s parameters may invalidate the result. This means that, from a model checking point of view, small, perhaps unimportant, deviations in quantities are indistinguishable from larger ones which may be critical.

As an example, Fig. 1 shows three simple timed-automaton models of a train crossing, each modeling that once the gates are closed, some time will pass before the train arrives. Now assume that the specification of the system is

The gates have to be closed 60 seconds before the train arrives.

Model *A* does guarantee this property, hence satisfies the specification. Model *B* only guarantees that the gates are closed 58 seconds before the train arrives, and in model *C*, only one second may pass between the gates closing and the train.

Neither of models *B* and *C* satisfies the specification, so this is the result which a model checker like for example Uppaal would output. What this does not tell us, however, is that model *C* is dangerously far away from the specification, whereas model *B* only violates it slightly (and may be acceptable from a practical point of view given other constraints on the system which we have not modeled here).

In order to address the robustness problem, one approach is to replace the Boolean yes-no answers of standard verification with distances. That is, the Boolean co-domain of model checking is replaced by the non-negative real numbers. In this setting, the Boolean **true** corresponds to a distance of zero and **false** to the non-zero numbers, so that quantitative model checking can now tell us not only that a specification is violated, but also *how much* it is violated, or *how far* the system is from corresponding to its specification.

In the example of Fig. 1, and depending on precisely how one wishes to measure distances, the distance from *A* to our specification would be 0, whereas the distances from *B* and *C* to the specification may be 2 and 59, for example. The precise interpretation of distance values will be application-dependent; but

in any case, it is clear that C is much farther away from the specification than B is.

The distance-based approach to quantitative verification has been developed in [8, 11, 28, 34, 46–48] and many other papers. Common to all these approaches is that they introduce distances between systems, or between systems and specifications, and then employ these for approximate or quantitative verification. However, depending on the application context, a plethora of different distances are being used. Consequently, there is a need for a general theory of quantitative verification which depends as little as possible on the concrete distances being used.

Different applications foster different types of quantitative verification, but it turns out that most of these essentially measure some type of distances between labeled transition systems. We have in [21] laid out a unifying framework which allows one to reason about such distance-based quantitative verification independently of the precise distance. This is essentially a general metric theory of labeled transition systems, with infinite quantitative games as its main theoretical ingredient and general fixed-point equations for linear and branching distances as one of its main results.

The work in [21] generalizes the linear-time–branching-time spectrum of preorders and equivalences from van Glabbeek’s [49] to a quantitative linear-time–branching-time spectrum of distances, all parameterized on a given distance on traces, or executions; *cf.* Fig. 2. This is done by generalizing Stirling’s bisimulation game [45] along two directions, both to cover all other preorders and equivalences in the linear-time–branching-time spectrum and into a game with quantitative (instead of Boolean) objectives.

What is missing in [21] are actual *algorithms* for computing the different types of distances. (The fixed-point equations mentioned above are generally defined over infinite lattices, hence Tarski’s fixed-point theorem does not help here.) In this paper, we take a different route to compute them. We translate the general quantitative games used in [21] to other, path-building games. We show that under mild conditions, this translation can always be effectuated, and that for all common trace distances, the resulting path-building games can be solved using various methods which we develop.

We start the paper by reviewing the quantitative games used to define linear and branching distances in [21] in Section 2. Then we show the reduction to path-building games in Section 3 and apply this to show how to compute all common branching distances in Section 4. We collect our results in the concluding section 5. The contributions of this paper are the following:

- (1) A general method to reduce quantitative bisimulation-type games to path-building games. The former can be posed as *double* path-building games, where the players alternate to build *two* paths; we show how to transform such games into a form where the players instead build *one* common path.
- (2) A collection of methods for solving different types of path-building games. Standard methods are available for solving discounted games and mean-payoff games; for other types we develop new methods.

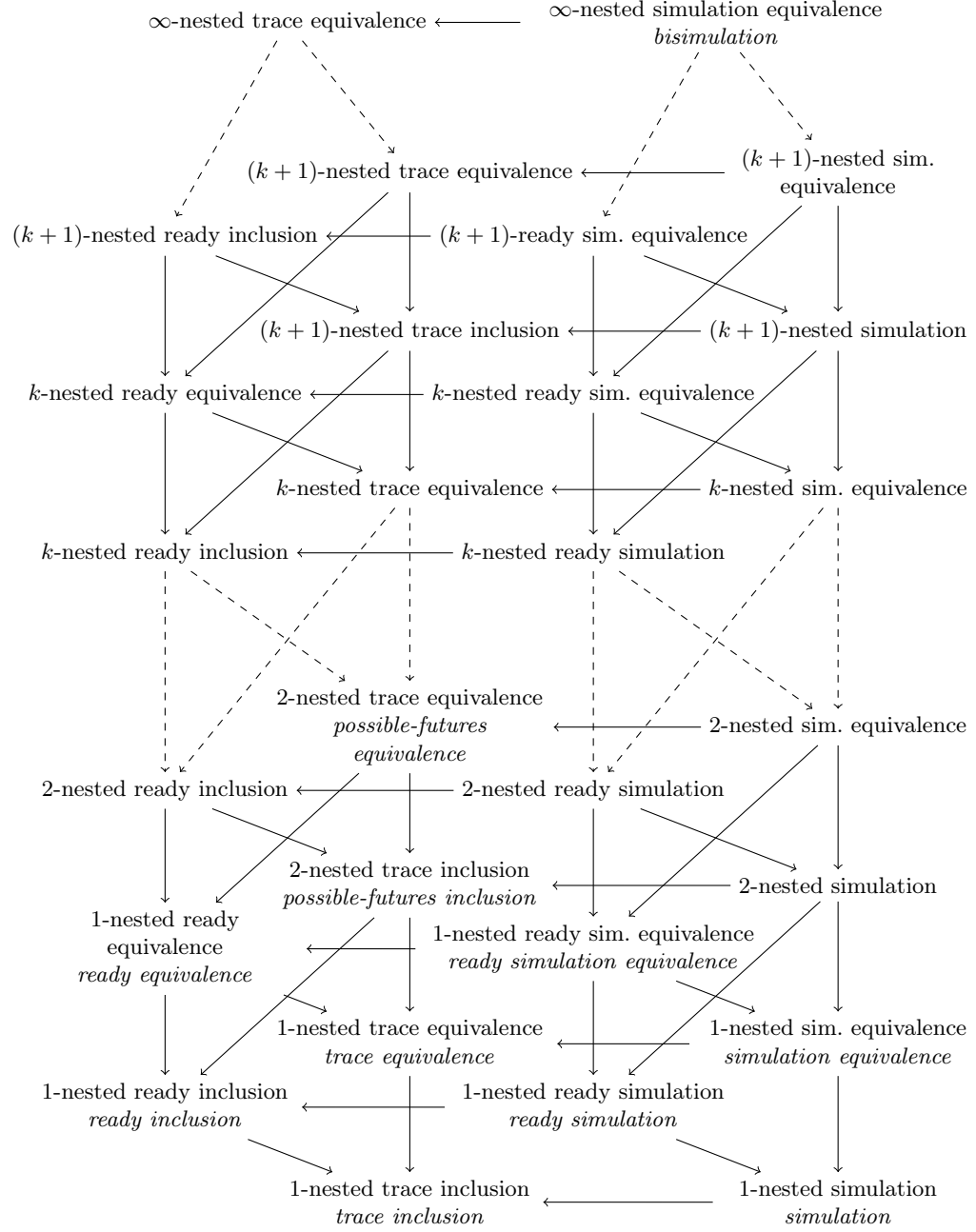


Fig. 2. The quantitative linear-time-branching-time spectrum from [21]. The nodes are different system distances, and an edge $d_1 \rightarrow d_2$ or $d_1 \dashrightarrow d_2$ indicates that $d_1(s, t) \geq d_2(s, t)$ for all states s, t , and that d_1 and d_2 in general are topologically inequivalent.

- (3) The application of the methods in (2) to compute various types of distances between labeled transition systems defined by the games of (1).

2 Linear and Branching Distances

Let Σ be a set of labels. Σ^ω denotes the set of infinite traces over Σ . We generally count sequences from index 0, so that $\sigma = (\sigma_0, \sigma_1, \dots)$. Let $\mathbb{R}_* = \mathbb{R}_{\geq 0} \cup \{\infty\}$ denote the extended non-negative real numbers.

2.1 Trace Distances

A *trace distance* is a hemimetric $D : \Sigma^\omega \times \Sigma^\omega \rightarrow \mathbb{R}_*$, i.e., a function which satisfies $D(\sigma, \sigma) = 0$ and $D(\sigma, \tau) + D(\tau, \nu) \geq D(\sigma, \nu)$ for all $\sigma, \tau, \nu \in \Sigma^\omega$.

The following is an exhaustive list of different trace distances which have been used in different applications. We refer to [21] for more details and motivation.

The discrete trace distance: $D_{\text{disc}}(\sigma, \tau) = 0$ if $\sigma = \tau$ and ∞ otherwise. This is equivalent to the standard Boolean setting: traces are either equal (distance 0) or not (distance ∞).

The point-wise trace distance: $D_{\text{sup}}(\sigma, \tau) = \sup_{n \geq 0} d(\sigma_n, \tau_n)$, for any given label distance $d : \Sigma \times \Sigma \rightarrow \mathbb{R}_*$. This measures the greatest individual symbol distance in the traces and has been used for quantitative verification in, among others, [9, 10, 12, 19, 39, 46].

The discounted trace distance: $D_+(\sigma, \tau) = \sum_{n=0}^{\infty} \lambda^n d(\sigma_n, \tau_n)$, for any given *discounting factor* $\lambda \in [0, 1[$. Sometimes also called *accumulating* trace distance, this accumulates individual symbol distances along traces, using discounting to adjust the values of distances further off. It has been used in, for example, [6, 19, 39, 46].

The limit-average trace distance: $D_{\text{lavg}}(\sigma, \tau) = \liminf_{n \geq 1} \frac{1}{n} \sum_{i=0}^{n-1} d(\sigma_i, \tau_i)$. This again accumulates individual symbol distances along traces and has been used in, among others, [6, 7]. Both discounted and limit-average distances are well-known from the theory of discounted and mean-payoff games [16, 51].

The Cantor trace distance: $D_C(\sigma, \tau) = \frac{1}{1 + \inf\{n \mid \sigma_n \neq \tau_n\}}$. This measures the (inverse of the) length of the common prefix of the traces and has been used for verification in [14].

The maximum-lead trace distance: $D_{\pm}(\sigma, \tau) = \sup_{n \geq 0} \left| \sum_{i=0}^n (\sigma_i - \tau_i) \right|$. Here it is assumed that Σ admits arithmetic operations of $+$ and $-$, for instance $\Sigma \subseteq \mathbb{R}$. As this measures differences of accumulated labels along runs, it is especially useful for real-time systems, cf. [20, 34, 46].

2.2 Labeled Transition Systems

A *labeled transition system* (LTS) over Σ is a tuple (S, i, T) consisting of a set of states S , with initial state $i \in S$, and a set of transitions $T \subseteq S \times \Sigma \times S$. We often write $s \xrightarrow{a} t$ to mean $(s, a, t) \in T$. We say that (S, i, T) is *finite* if S and T are finite. We assume our LTS to be *non-blocking* in the sense that for every state $s \in S$ there is a transition $(s, a, t) \in T$.

We have shown in [21] how any given trace distance D can be lifted to a quantitative linear-time-branching-time spectrum of distances on LTS. This is done via quantitative games as we shall review below. The point of [21] was that if the given trace distance has a recursive formulation, which, as we show in [21], every commonly used trace distance has, then the corresponding linear and branching distances can be formulated as fixed points for certain monotone functionals.

The fixed-point formulation of [21] does not, however, give rise to actual algorithms for computing linear and branching distances, as it happens more often than not that the mentioned monotone functionals are defined over infinite lattices. Concretely, this is the case for all but the point-wise trace distances in Section 2.1. Hence other methods are required for computing them; developing these is the purpose of this paper.

2.3 Quantitative Ehrenfeucht-Fraïssé Games

We review the quantitative games used in [21] to define different types of linear and branching distances for any given trace distance D . For conciseness, we only introduce *simulation games* and *bisimulation games* here, but similar definitions may be given for all equivalences and preorders in the linear-time-branching-time spectrum [49].

Quantitative Simulation Games Let $\mathcal{S} = (S, i, T)$ and $\mathcal{S}' = (S', i', T')$ be LTS and $D : \Sigma^\omega \times \Sigma^\omega \rightarrow \mathbb{R}_*$ a trace distance. The *simulation game* from $\mathcal{S}$ to $\mathcal{S}'$ is played by two players, the maximizer and the minimizer. A play begins with the maximizer choosing a transition $(s_0, a_0, s_1) \in T$ with $s_0 = i$. Then the minimizer chooses a transition $(s'_0, a'_0, s'_1) \in T'$ with $s'_0 = i'$. Now the maximizer chooses a transition $(s_1, a_1, s_2) \in T$, then the minimizer chooses a transition $(s'_1, a'_1, s'_2) \in T'$, and so on indefinitely. Hence this is what should be called a *double path-building game*: the players each build, independently, an infinite path in their respective LTS.

A *play* hence consists of two infinite paths, π starting from i , and π' starting from i' . The *utility* of this play is the distance $D(\sigma, \sigma')$ between the traces σ, σ' of the paths π and π' , which the maximizer wants to maximize and the minimizer wants to minimize. The *value* of the game is, then, the utility of the play which results when both maximizer and minimizer are playing optimally.

To formalize the above intuition, we define a *configuration* for the maximizer to be a pair (π, π') of finite paths of equal length, π in $\mathcal{S}$ and starting in i , π' in $\mathcal{S}'$ starting in i' . The intuition is that this covers the *history* of a play; the choices both players have made up to a certain point in the game. Hence a configuration for the minimizer is a similar pair (π, π') of finite paths, but now π is one step longer than π' .

A *strategy* for the maximizer is a mapping from maximizer configurations to transitions in $\mathcal{S}$, fixing the maximizer's choice of a move in the given configuration. Denoting the set of maximizer configurations by Conf , such a strategy is hence a mapping $\theta : \text{Conf} \rightarrow T$ such that for all $(\pi, \pi') \in \text{Conf}$ with $\theta(\pi, \pi') = (s, a, t)$, we have $\text{end}(\pi) = s$. Here $\text{end}(\pi)$ denotes the last state of π . Similarly, and denoting the set of minimizer configurations by Conf' , a strategy for the minimizer is a mapping $\theta' : \text{Conf}' \rightarrow T'$ such that for all $(\pi, \pi') \in \text{Conf}'$ with $\theta'(\pi, \pi') = (s', a', t')$, $\text{end}(\pi') = s'$.

Denoting the sets of these strategies by Θ and Θ' , respectively, we can now define the *simulation distance* from $\mathcal{S}$ to $\mathcal{S}'$ induced by the trace distance D , denoted $D^{\text{sim}}(\mathcal{S}, \mathcal{S}')$, by

$$D^{\text{sim}}(\mathcal{S}, \mathcal{S}') = \sup_{\theta \in \Theta} \inf_{\theta' \in \Theta'} D(\sigma(\theta, \theta'), \sigma'(\theta, \theta')),$$

where $\sigma(\theta, \theta')$ and $\sigma'(\theta, \theta')$ are the traces of the paths $\pi(\theta, \theta')$ and $\pi'(\theta, \theta')$ induced by the pair of strategies (θ, θ') .

Remark 1. If the trace distance D is discrete, *i.e.*, $D = D_{\text{disc}}$ as in Section 2.1, then the quantitative game described above reduces to the well-known *simulation game* [45]: The only choice the minimizer has for minimizing the value of the game is to always choose a transition with the same label as the one just chosen by the maximizer; similarly, the maximizer needs to try to force the game into states where she can choose a transition which the minimizer cannot match. Hence the value of the game will be 0 if the minimizer always can match the maximizer's labels, that is, iff $\mathcal{S}$ is simulated by $\mathcal{S}'$.

Quantitative Bisimulation Games There is a similar game for computing the *bisimulation distance* between LTS $\mathcal{S}$ and $\mathcal{S}'$. Here we give the maximizer the choice, at each step, to either choose a transition from s_k as before, or to “switch sides” and choose a transition from s'_k instead; the minimizer then has to answer with a transition on the other side.

Hence the players are still building two paths, one in each LTS, but now they are *both* contributing to *both* paths. The utility of such a play is still the distance between these two paths, which the maximizer wants to maximize and the minimizer wants to minimize. The *bisimulation distance* between $\mathcal{S}$ and $\mathcal{S}'$, denoted $D^{\text{bisim}}(\mathcal{S}, \mathcal{S}')$, is then defined to be the value of this quantitative bisimulation game.

Remark 2. If the trace distance $D = D_{\text{disc}}$ is discrete, then using the same arguments as in Remark 1, we see that $D_{\text{disc}}^{\text{bisim}}(\mathcal{S}, \mathcal{S}') = 0$ iff $\mathcal{S}$ and $\mathcal{S}'$ are *bisimilar*. The game which results being played is precisely the bisimulation game of [45], which also has been introduced by Fraïssé [23] and Ehrenfeucht [15] in other contexts.

The Quantitative Linear-Time–Branching-Time Spectrum The above-defined quantitative simulation and bisimulation games can be generalized using different methods. One is to introduce a *switch counter* sc into the game which counts how often the maximizer has switched sides during an ongoing game. Then one can limit the maximizer’s capabilities by imposing limits on sc : if the limit is $\text{sc} = 0$, then the players are playing a simulation game; if there is no limit ($\text{sc} \leq \infty$), they are playing a bisimulation game. Other limits $\text{sc} \leq k$, for $k \in \mathbb{N}$, can be used to define *k-nested simulation distances*, generalizing the equivalences and preorders from [29, 32].

Another method of generalization is to introduce *ready moves* into the game. These consist of the maximizer challenging her opponent by switching sides, but only requiring that the minimizer match the chosen transition; afterwards the game finishes. This can be employed to introduce the *ready simulation distance* of [41] and, combined with the switch counter method above, the *ready k-nested simulation distance*. We refer to [21] for further details on these and other variants of quantitative (bi)simulation games.

For reasons of exposition, we will below introduce our reduction to path-building games only for the quantitative simulation and bisimulation games; but all our work can easily be transferred to the general setting of [21].

3 Reduction

In order to compute simulation and bisimulation distances, we translate the games of the previous section to path-building games à la Ehrenfeucht-Mycielski [16]. Let $D : \Sigma^\omega \times \Sigma^\omega \rightarrow \mathbb{R}_*$ be a trace distance, and assume that there are functions $\text{val}_D : \mathbb{R}_*^\omega \rightarrow \mathbb{R}_*$ and $f_D : \Sigma \times \Sigma \rightarrow \mathbb{R}_*$ for which it holds, for all $\sigma, \tau \in \Sigma^\omega$, that

$$D(\sigma, \tau) = \text{val}_D(0, f_D(\sigma_0, \tau_0), 0, f_D(\sigma_1, \tau_1), 0, \dots). \quad (1)$$

We will need these functions in our translation, and we show in Section 3.2 below that they exist for all common trace distances.

3.1 Simulation Distance

Let $\mathcal{S} = (S, i, T)$ and $\mathcal{S}' = (S', i', T')$ be LTS. We construct a turn-based game $\mathcal{U} = \mathcal{U}(\mathcal{S}, \mathcal{S}') = (U, u_0, \rightarrow)$ as follows, with $U = U_1 \cup U_2$:

$$\begin{aligned} U_1 &= S \times S' & U_2 &= S \times S' \times \Sigma & u_0 &= (i, i') \\ \rightarrow &= \{(s, s') \xrightarrow{0} (t, s', a) \mid (s, a, t) \in T\} \\ &\cup \{(t, s', a) \xrightarrow{f_D(a, a')} (t, t') \mid (s', a', t') \in T'\} \end{aligned}$$

This is a two-player game. We again call the players maximizer and minimizer, with the maximizer controlling the states in U_1 and the minimizer the ones in U_2 . Transitions are labeled with extended real numbers, but as the image of f_D in $\mathbb{R}_*$ is finite, the set of transition labels in U is finite.

The game on $\mathcal{U}$ is played as follows. A play begins with the maximizer choosing a transition $(u_0, a_0, u_1) \in \rightarrow$ with $u_0 = i$. Then the minimizer chooses a transition $(u_1, a_1, u_2) \in \rightarrow$. Then the maximizer chooses a transition $(u_2, a_2, u_3) \in \rightarrow$, and so on indefinitely (note that $\mathcal{U}$ is non-blocking). A play thus induces an infinite path $\pi = (u_0, a_0, u_1), (u_1, a_1, u_2), \dots$ in $\mathcal{U}$ with $u_0 = i$. The goal of the maximizer is to maximize the value $\text{val}_D(\mathcal{U}) := \text{val}_D(a_0, a_1, \dots)$ of the trace of π ; the goal of the minimizer is to minimize this value.

This is hence a path-building game, variations of which (for different valuation functions) have been studied widely in both economics and computer science since Ehrenfeucht-Mycielski's [16]. Formally, configurations and strategies are given as follows. A configuration of the maximizer is a path π_1 in $\mathcal{U}$ with $\text{end}(\pi_1) \in U_1$, and a configuration of the minimizer is a path π_2 in $\mathcal{U}$ with $\text{end}(\pi_2) \in U_2$. Denote the sets of these configurations by Conf_1 and Conf_2 , respectively. A strategy for the maximizer is, then, a mapping $\theta_1 : \text{Conf}_1 \rightarrow \rightarrow$ such that for all $\pi_1 \in \text{Conf}_1$ with $\theta_1(\pi_1) = (u, x, v)$, $\text{end}(\pi_1) = u$. A strategy for the minimizer is a mapping $\theta_2 : \text{Conf}_2 \rightarrow \rightarrow$ such that for all $\pi_2 \in \text{Conf}_2$ with $\theta_2(\pi_2) = (u, x, v)$, $\text{end}(\pi_2) = u$. Denoting the sets of these strategies by Θ_1 and Θ_2 , respectively, we can now define

$$\text{val}_D(\mathcal{U}) = \sup_{\theta_1 \in \Theta_1} \inf_{\theta_2 \in \Theta_2} \text{val}_D(\sigma(\theta_1, \theta_2)),$$

where $\sigma(\theta_1, \theta_2)$ is the trace of the path $\pi(\theta_1, \theta_2)$ induced by the pair of strategies (θ_1, θ_2) .

By the next theorem, the value of $\mathcal{U}$ is precisely the simulation distance from $\mathcal{S}$ to $\mathcal{S}'$.

Theorem 3. *For all LTS $\mathcal{S}, \mathcal{S}'$, $D^{\text{sim}}(\mathcal{S}, \mathcal{S}') = \text{val}_D(\mathcal{U}(\mathcal{S}, \mathcal{S}'))$.*

Proof. Write $\mathcal{S} = (S, i, T)$ and $\mathcal{S}' = (S', i', T')$. Informally, the reason for the equality is that any move $(s, a, t) \in T$ of the maximizer in the simulation distance game can be copied to a move $(s, s') \xrightarrow{0} (t, s', a)$, regardless of s' , in $\mathcal{U}$. Similarly,

any move (s', a', t') of the minimizer can be copied to a move $(t, s', a) \xrightarrow{f_D(a, a')} (t, t')$, and all the moves in $\mathcal{U}$ are of this form.

To turn this idea into a formal proof, we show that there are bijections between configurations and strategies in the two games, and that under these bijections, the utilities of the two games are equal. For $(\pi, \pi') \in \text{Conf}$ in the simulation distance game, with $\pi = (s_0, a_0, s_1), \dots, (s_{n-1}, a_{n-1}, s_n)$ and $\pi' = (s'_0, a'_0, s'_1), \dots, (s'_{n-1}, a'_{n-1}, s'_n)$, define

$$\begin{aligned} \phi_1(\pi, \pi') = & ((s_0, s'_0), 0, (s_1, s'_0, a_0)), ((s_1, s'_0, a_0), f_D(a_0, a'_0), (s_1, s'_1)), \dots, \\ & ((s_n, s'_{n-1}, a_{n-1}), f_D(a_{n-1}, a'_{n-1}), (s_n, s'_n)). \end{aligned}$$

It is clear that this defines a bijection $\phi_1 : \text{Conf} \rightarrow \text{Conf}_1$, and that one can similarly define a bijection $\phi_2 : \text{Conf}' \rightarrow \text{Conf}_2$.

Now for every strategy $\theta : \text{Conf} \rightarrow T$ in the simulation distance game, define a strategy $\psi_1(\theta) = \theta_1 \in \Theta_1$ as follows. For $\pi_1 \in \text{Conf}_1$, let $(\pi, \pi') = \phi_1^{-1}(\pi_1)$ and $s' = \text{end}(\pi')$. Let $\theta(\pi, \pi') = (s, a, t)$ and define $\theta_1(\pi_1) = ((s, s'), 0, (t, s', a))$. Similarly we define a mapping $\psi_2 : \Theta' \rightarrow \Theta_2$ as follows. For $\theta' : \text{Conf}' \rightarrow T'$ and $\pi_2 \in \text{Conf}_2$, let $(\pi, \pi') = \phi_2^{-1}(\pi_2)$ with $\pi = (s_0, a_0, s_1), \dots, (s_n, a_n, s_{n+1})$. Let $\theta'(\pi, \pi') = (s', a', t')$ and define $\psi_2(\theta')(\pi_2) = ((s_{n+1}, s', a_n), f_D(a_n, a'), (s_{n+1}, t'))$.

It is clear that ψ_1 and ψ_2 indeed map strategies in the simulation distance game to strategies in $\mathcal{U}$ and that both are bijections. Also, for each pair $(\theta, \theta') \in \Theta \times \Theta'$, $D(\sigma(\theta, \theta'), \sigma'(\theta, \theta')) = \text{val}_D(\sigma(\psi_1(\theta), \psi_2(\theta')))$ by construction. But then

$$\begin{aligned} D^{\text{sim}}(\mathcal{S}, \mathcal{S}') &= \sup_{\theta \in \Theta} \inf_{\theta' \in \Theta'} D(\sigma(\theta, \theta'), \sigma'(\theta, \theta')) \\ &= \sup_{\theta \in \Theta} \inf_{\theta' \in \Theta'} \text{val}_D(\sigma(\psi_1(\theta), \psi_2(\theta'))) \\ &= \sup_{\theta_1 \in \Theta_1} \inf_{\theta_2 \in \Theta_2} \text{val}_D(\sigma(\theta_1, \theta_2)) = \text{val}_D(\mathcal{U}), \end{aligned}$$

the third equality because ψ_1 and ψ_2 are bijections. $\square$

3.2 Examples

We show that the reduction applies to all trace distances from Section 2.1.

1. For the discrete trace distance $D = D_{\text{disc}}$, we let

$$\text{val}_D(x) = \sum_{n=0}^{\infty} x_n, \quad f_D(a, b) = \begin{cases} 0 & \text{if } a = b, \\ \infty & \text{otherwise,} \end{cases}$$

then (1) holds. In the game on $\mathcal{U}$, the minimizer needs to play 0-labeled transitions to keep the distance at 0.

2. For the point-wise trace distance $D = D_{\text{sup}}$, we can let

$$\text{val}_D(x) = \sup_{n \geq 0} x_n, \quad f_D(a, b) = d(a, b).$$

Hence the game on $\mathcal{U}$ computes the sup of a trace.

3. For the discounted trace distance $D = D_+$, let

$$\text{val}_D(x) = \sum_{n=0}^{\infty} \sqrt{\lambda}^n x_n, \quad f_D(a, b) = \sqrt{\lambda} d(a, b),$$

then (1) holds. Hence the game on $\mathcal{U}$ is a standard discounted game [51].

4. For the limit-average trace distance $D = D_{\text{avg}}$, we can let

$$\text{val}_D(x) = \liminf_{n \geq 1} \frac{1}{n} \sum_{i=0}^{n-1} x_i, \quad f_D(a, b) = 2d(a, b);$$

we will show below that (1) holds. Hence the game on $\mathcal{U}$ is a mean-payoff game [51].

5. For the Cantor trace distance $D = D_C$, let

$$\text{val}_D(x) = \frac{2}{1 + \inf\{n \mid x_n \neq 0\}}, \quad f_D(a, b) = \begin{cases} 0 & \text{if } a = b, \\ 1 & \text{otherwise.} \end{cases}$$

The objective of the maximizer in this game is to reach a transition with weight 1 *as soon as possible*.

6. For the maximum-lead trace distance $D = D_{\pm}$, we can let

$$\text{val}_D(x) = \sup_{n \geq 0} \left| \sum_{i=0}^n x_i \right|, \quad f_D(a, b) = a - b,$$

then (1) holds.

3.3 Bisimulation Distance

We can construct a similar turn-based game to compute the bisimulation distance. Let $\mathcal{S} = (S, i, T)$ and $\mathcal{S}' = (S', i', T')$ be LTS and define $\mathcal{V} = \mathcal{V}(\mathcal{S}, \mathcal{S}') = (V, v_0, \rightarrow)$ as follows, with $V = V_1 \cup V_2$:

$$\begin{aligned} V_1 &= S \times S' & V_2 &= S \times S' \times \Sigma \times \{1, 2\} & v_0 &= (i, i') \\ \rightarrow &= \{(s, s') \xrightarrow{0} (t, s', a, 1) \mid (s, a, t) \in T\} \\ &\cup \{(s, s') \xrightarrow{0} (s, t', a', 2) \mid (s', a', t') \in T'\} \\ &\cup \{(t, s', a, 1) \xrightarrow{f_D(a, a')} (t, t') \mid (s', a', t') \in T'\} \\ &\cup \{(s, t', a', 2) \xrightarrow{f_D(a, a')} (t, t') \mid (s, a, t) \in T\} \end{aligned}$$

Here we have used the minimizer's states to both remember the label choice of the maximizer and which side of the bisimulation game she plays on. By suitable modifications, we can construct similar games for all distances in the spectrum of [21]. The next theorem states that the value of $\mathcal{V}$ is precisely the bisimulation distance between $\mathcal{S}$ and $\mathcal{S}'$.

Theorem 4. *For all LTS $\mathcal{S}, \mathcal{S}'$, $D^{\text{bisim}}(\mathcal{S}, \mathcal{S}') = \text{val}_D(\mathcal{V}(\mathcal{S}, \mathcal{S}'))$.*

Proof. This proof is similar to the one of Theorem 3, only that now, we have to take into account that the maximizer may “switch sides”. The intuition is that maximizer moves (s, a, t) in the $\mathcal{S}$ component of the bisimulation distance games are emulated by moves $(s, s') \xrightarrow{0} (t, s', a, 1)$, maximizer moves (s', a', t') in the $\mathcal{S}'$ component are emulated by moves $(s, s') \xrightarrow{0} (s, t', a', 2)$, and similarly for the minimizer. The values 1 and 2 in the last component of the V_2 states ensure that the minimizer only has moves available which correspond to playing in the correct component in the bisimulation distance game (*i.e.*, that ψ_2 is a bijection). $\square$

4 Computing the Values of Path-Building Games

We show here how to compute the values of the different path-building games which we saw in the last section. This will give us algorithms to compute all simulation and bisimulation distances associated with the trace distances of Section 2.1.

We will generally only refer to the games $\mathcal{U}$ for computing simulation distance here, but the bisimulation distance games $\mathcal{V}$ are very similar, and everything we say also applies to them.

Discrete distance: The game to compute the discrete simulation distances is a reachability game, in that the goal of the maximizer is to force the minimizer into a state from which she can only choose ∞ -labeled transitions. We can hence solve them using the standard controllable-predecessor operator defined, for any set $S \subseteq U_1$ of maximizer states, by

$$\text{cpre}(S) = \{u_1 \in U_1 \mid \exists u_1 \xrightarrow{0} u_2 : \forall u_2 \xrightarrow{x} u_3 : u_3 \in S\}.$$

Now let $S \subseteq U_1$ be the set of states from which the maximizer can force the game into a state from which the minimizer only has ∞ -labeled transitions, *i.e.*,

$$S = \{u_1 \in U_1 \mid \exists u_1 \xrightarrow{0} u_2 : \forall u_2 \xrightarrow{x} u_3 : x = \infty\},$$

and compute $S^* = \text{cpre}^*(S) = \bigcup_{n \geq 0} \text{cpre}^n(W)$. By monotonicity of cpre and as the subset lattice of U_1 is complete and finite, this computation finishes in at most $|U_1|$ steps, and we have $\text{val}_D(\mathcal{U}) = 0$ iff $u_0 \notin W^*$.

Point-wise distance: To compute the value of the point-wise simulation distance game, let $W = \{w_1, \dots, w_m\}$ be the (finite) set of weights of the minimizer’s transitions, ordered such that $w_1 < \dots < w_m$. For each $i = 1, \dots, m$, let $S_i = \{u_1 \in U_1 : \exists u_1 \xrightarrow{0} u_2 : \forall u_2 \xrightarrow{x} u_3 : x \geq w_i\}$ be the set of maximizer states from which the maximizer can force the minimizer into a transition with weight at least w_i ; note that $S_m \subseteq S_{m-1} \subseteq \dots \subseteq S_1 = U_1$. For each $i = 1, \dots, m$, compute $S_i^* = \text{cpre}^*(S_i)$, then $S_m^* \subseteq S_{m-1}^* \subseteq \dots \subseteq S_1^* = U_1$. Let p be the greatest index for which $u_0 \in S_p$, then $p = \text{val}_D(\mathcal{U})$.

Discounted distance: The game to compute the discounted simulation distance is a standard discounted game and can be solved by standard methods [51].

Limit-average distance: For the limit-average simulation distance game, let $(y_n)_{n \geq 1}$ be the sequence $(1, 1, \frac{3}{2}, 1, \frac{5}{4}, \dots)$ and note that $\lim_{n \rightarrow \infty} y_n = 1$. Then

$$\begin{aligned} \text{val}_D(x) &= \text{val}_D(x) \lim_{n \rightarrow \infty} y_n = \liminf_{n \geq 1} \frac{y_n}{n} \sum_{i=0}^{n-1} x_i \\ &= \liminf_{2k \geq 1} \frac{1}{2k} \sum_{i=0}^{k-1} f_D(\sigma_i, \tau_i) \\ &= \liminf_{k \geq 1} \frac{1}{k} \sum_{i=0}^{k-1} d(\sigma_i, \tau_i) = D_{\text{lavg}}(\sigma, \tau), \end{aligned}$$

so, indeed, (1) holds. The game is a standard mean-payoff game and can be solved by standard methods, for example [13].

Cantor distance: To compute the value of the Cantor simulation distance game, let $S_1 \subseteq U_1$ be the set of states from which the maximizer can force the game into a state from which the minimizer only has 1-labeled transitions, *i.e.*, $S_1 = \{u_1 \in U_1 \mid \exists u_1 \xrightarrow{0} u_2 : \forall u_2 \xrightarrow{x} u_3 : x = 1\}$. Now recursively compute $S_{i+1} = S_i \cup \text{cpre}(S_i)$, for $i = 1, 2, \dots$, until $S_{i+1} = S_i$ (which, as $S_i \subseteq S_{i+1}$ for all i and U_1 is finite, will happen eventually). Then S_i is the set of states from which the maximizer can force the game to a 1-labeled minimizer transition which is at most $2i$ steps away. Hence $\text{val}_D(\mathcal{U}) = 0$ if there is no p for which $u_0 \in S_p$, and otherwise $\text{val}_D(\mathcal{U}) = \frac{1}{p}$, where p is the least index for which $u_0 \in S_p$.

Maximum-lead distance: For the maximum-lead simulation distance game, we note that the maximizer wants to maximize $\sup_{n \geq 0} |\sum_{i=0}^n x_i|$, *i.e.*, wants the accumulated values $\sum_{i=0}^n x_i$ or $-\sum_{i=0}^n x_i$ to exceed any prescribed bounds. Thus, the maximizer is the opponent in an interval-bound energy game [4]. It is shown in [4] that solving such games is EXPTIME-complete.

5 Conclusion and Future Work

We sum up our results in the following corollary which gives the complexities of the decision problems associated with the respective distance computations. Note that the first part restates the well-known fact that simulation and bisimulation are decidable in polynomial time.

Corollary 5.

1. *Discrete simulation and bisimulation distances are computable in PTIME.*
2. *Point-wise simulation and bisimulation distances are computable in PTIME.*
3. *Discounted simulation and bisimulation distances are computable in $NP \cap coNP$.*
4. *Limit-average simulation and bisimulation distances are computable in $NP \cap coNP$.*
5. *Cantor simulation and bisimulation distances are computable in PTIME.*
6. *Maximum-lead simulation and bisimulation distances are computable in EXPTIME.*

In the future, we intend to expand our work to also cover *quantitative specification theories*. Together with several coauthors, we have in [17, 18] developed a comprehensive setting for satisfaction and refinement distances in quantitative specification theories. Using our work in [22] on a qualitative linear-time–branching-time spectrum of specification theories, we plan to introduce a quantitative linear-time–branching-time spectrum of specification distances and to use the setting developed here to devise methods for computing them through path-building games.

References

1. Rajeev Alur, Costas Courcoubetis, Nicolas Halbwachs, Thomas A. Henzinger, Pei-Hsin Ho, Xavier Nicollin, Alfredo Olivero, Joseph Sifakis, and Sergio Yovine. The algorithmic analysis of hybrid systems. *Theor. Comput. Sci.*, 138(1):3–34, 1995.
2. Rajeev Alur and David L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994.
3. Adnan Aziz, Kumud Sanwal, Vigyan Singhal, and Robert K. Brayton. Model-checking continuous-time Markov chains. *ACM Transactions on Computational Logics*, 1(1):162–170, 2000.
4. Patricia Bouyer, Uli Fahrenberg, Kim Guldstrand Larsen, Nicolas Markey, and Jiri Srba. Infinite runs in weighted timed automata with energy constraints. In Franck Cassez and Claude Jard, editors, *FORMATS*, volume 5215 of *Lect. Notes Comput. Sci.*, pages 33–47. Springer-Verlag, 2008.
5. Joakim Byg, Kenneth Yrke Jørgensen, and Jiří Srba. TAPAAL: editor, simulator and verifier of timed-arc Petri nets. In Zhiming Liu and Anders P. Ravn, editors, *ATVA*, volume 5799 of *Lect. Notes Comput. Sci.*, pages 84–89. Springer-Verlag, 2009.

6. Pavol Černý, Thomas A. Henzinger, and Arjun Radhakrishna. Simulation distances. *Theor. Comput. Sci.*, 413(1):21–35, 2012.
7. Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Quantitative languages. *ACM Transactions on Computational Logic*, 11(4), 2010.
8. Luca de Alfaro, Marco Faella, Thomas A. Henzinger, Rupak Majumdar, and Mariëlle Stoelinga. Model checking discounted temporal properties. *Theor. Comput. Sci.*, 345(1):139–170, 2005.
9. Luca de Alfaro, Marco Faella, and Mariëlle Stoelinga. Linear and branching system metrics. *IEEE Transactions on Software Engineering*, 35(2):258–273, 2009.
10. Luca de Alfaro, Thomas A. Henzinger, and Rupak Majumdar. Discounting the future in systems theory. In Jos C. M. Baeten, Jan Karel Lenstra, Joachim Parrow, and Gerhard J. Woeginger, editors, *Automata, Languages and Programming, 30th International Colloquium, ICALP 2003, Eindhoven, The Netherlands, June 30 - July 4, 2003. Proceedings*, volume 2719 of *Lecture Notes in Computer Science*, pages 1022–1037. Springer, 2003.
11. Josee Desharnais, Vineet Gupta, Radha Jagadeesan, and Prakash Panangaden. Metrics for labelled Markov processes. *Theor. Comput. Sci.*, 318(3):323–354, 2004.
12. Josée Desharnais, François Laviolette, and Mathieu Tracol. Approximate analysis of probabilistic processes. In *QEST*, pages 264–273. IEEE Computer Society, 2008.
13. Vishesh Dhingra and Stephane Gaubert. How to solve large scale deterministic games with mean payoff by policy iteration. In Luciano Lenzini and Rene L. Cruz, editors, *VALUETOOLS*, volume 180 of *ACM Int. Conf. Proc.*, page 12. ACM, 2006.
14. Laurent Doyen, Thomas A. Henzinger, Axel Legay, and Dejan Ničković. Robustness of sequential circuits. In Luís Gomes, Victor Khomenko, and João M. Fernandes, editors, *ACSD*, pages 77–84. IEEE Computer Society, 2010.
15. Andrzej Ehrenfeucht. An application of games to the completeness problem for formalized theories. *Fund. Math.*, 49:129–141, 1961.
16. Andrzej Ehrenfeucht and Jan Mycielski. Positional strategies for mean payoff games. *Int. J. Game Theory*, 8:109–113, 1979.
17. Uli Fahrenberg, Jan Křetínský, Axel Legay, and Louis-Marie Traonouez. Compositionality for quantitative specifications. In Ivan Lanese and Eric Madelaine, editors, *FACS*, volume 8997 of *Lect. Notes Comput. Sci.*, pages 306–324. Springer-Verlag, 2014.
18. Uli Fahrenberg, Jan Křetínský, Axel Legay, and Louis-Marie Traonouez. Compositionality for quantitative specifications. *Soft Comput.*, 22(4):1139–1158, 2018.
19. Uli Fahrenberg, Kim G. Larsen, and Claus Thrane. A quantitative characterization of weighted Kripke structures in temporal logic. *Computing and Informatics*, 29(6+):1311–1324, 2010.
20. Uli Fahrenberg and Axel Legay. A robust specification theory for modal event-clock automata. In Sebastian S. Bauer and Jean-Baptiste Raclet, editors, *Proceedings Fourth Workshop on Foundations of Interface Technologies, FIT 2012, Tallinn, Estonia, 25th March 2012.*, volume 87 of *EPTCS*, pages 5–16, 2012.
21. Uli Fahrenberg and Axel Legay. The quantitative linear-time-branching-time spectrum. *Theor. Comput. Sci.*, 538:54–69, 2014.
22. Uli Fahrenberg and Axel Legay. A linear-time-branching-time spectrum of behavioral specification theories. In Bernhard Steffen, Christel Baier, Mark van den Brand, Johann Eder, Mike Hinchey, and Tiziana Margaria, editors, *SOFSEM*, volume 10139 of *Lect. Notes Comput. Sci.*, pages 49–61. Springer-Verlag, 2017.

23. Roland Fraïssé. Sur quelques classifications des systèmes de relations. *Publ. Scient. de l'Univ. d'Alger, Série A*, 1:35–182, 1954.
24. Martin Fränzle and Christian Herde. HySAT: An efficient proof engine for bounded model checking of hybrid systems. *Formal Meth. Syst. Design*, 30(3):179–198, 2007.
25. Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. SpaceEx: Scalable verification of hybrid systems. In Ganesh Gopalakrishnan and Shaz Qadeer, editors, *CAV*, volume 6806 of *Lect. Notes Comput. Sci.*, pages 379–395. Springer-Verlag, 2011.
26. Guillaume Gardey, Didier Lime, Morgan Magnin, and Olivier H. Roux. Romeo: A tool for analyzing time Petri nets. In Kousha Etessami and Sriram K. Rajamani, editors, *CAV*, volume 3576 of *Lect. Notes Comput. Sci.*, pages 418–423. Springer-Verlag, 2005.
27. Stephen Gilmore and Jane Hillston. The PEPA workbench: A tool to support a process algebra-based approach to performance modelling. In Günter Haring and Gabriele Kotsis, editors, *CPE*, volume 794 of *Lect. Notes Comput. Sci.*, pages 353–368. Springer-Verlag, 1994.
28. Antoine Girard and George J. Pappas. Approximation metrics for discrete and continuous systems. *IEEE Trans. Automat. Contr.*, 52(5):782–798, 2007.
29. Jan Friso Groote and Frits W. Vaandrager. Structured operational semantics and bisimulation as a congruence. *Inf. Comput.*, 100(2):202–260, 1992.
30. Hans-Michael Hanisch. Analysis of place/transition nets with timed arcs and its application to batch process control. In Marco Ajmone Marsan, editor, *ATPN*, volume 691 of *Lect. Notes Comput. Sci.*, pages 282–299. Springer-Verlag, 1993.
31. Hans Hansson and Bengt Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5):512–535, 1994.
32. Matthew Hennessy and Robin Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32(1):137–161, 1985.
33. Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. HYTECH: A model checker for hybrid systems. *Int. J. Softw. Tools Techn. Trans.*, 1(1-2):110–122, 1997.
34. Thomas A. Henzinger, Rupak Majumdar, and Vinayak S. Prabhu. Quantifying similarities between timed systems. In Paul Pettersson and Wang Yi, editors, *Formal Modeling and Analysis of Timed Systems, Third International Conference, FORMATS 2005, Uppsala, Sweden, September 26-28, 2005, Proceedings*, volume 3829 of *Lecture Notes in Computer Science*, pages 226–241. Springer, 2005.
35. Thomas A. Henzinger, Xavier Nicollin, Joseph Sifakis, and Sergio Yovine. Symbolic model checking for real-time systems. *Inf. Comput.*, 111(2):193–244, 1994.
36. Jane Hillston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, 1996.
37. Ron Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.
38. Marta Z. Kwiatkowska, Gethin Norman, and David Parker. Probabilistic symbolic model checking with PRISM: A hybrid approach. In Joost-Pieter Katoen and Perdita Stevens, editors, *TACAS*, volume 2280 of *Lect. Notes Comput. Sci.*, pages 52–66. Springer-Verlag, 2002.
39. Kim G. Larsen, Uli Fahrenberg, and Claus Thrane. Metrics for weighted transition systems: Axiomatization and complexity. *Theor. Comput. Sci.*, 412(28):3358–3369, 2011.

40. Kim G. Larsen, Paul Pettersson, and Wang Yi. UPPAAL in a nutshell. *Int. J. Softw. Tools Techn. Trans.*, 1(1-2):134–152, 1997.
41. Kim G. Larsen and Arne Skou. Bisimulation through probabilistic testing. In *POPL*, pages 344–352. ACM Press, 1989.
42. Philip M. Merlin and David J. Farber. Recoverability of communication protocols—implications of a theoretical study. *IEEE Trans. Commun.*, 24(9):1036 – 1043, 1976.
43. Roberto Segala and Nancy A. Lynch. Probabilistic simulations for probabilistic processes. In Bengt Jonsson and Joachim Parrow, editors, *CONCUR*, volume 836 of *Lect. Notes Comput. Sci.*, pages 481–496. Springer-Verlag, 1994.
44. William J. Stewart. *Introduction to the Numerical Solution of Markov Chains*. Princeton University Press, 1994.
45. Colin Stirling. Modal and temporal logics for processes. In Faron Moller and Graham M. Birtwistle, editors, *Banff Higher Order Workshop*, volume 1043 of *Lect. Notes Comput. Sci.*, pages 149–237. Springer-Verlag, 1995.
46. Claus Thrane, Uli Fahrenberg, and Kim G. Larsen. Quantitative analysis of weighted transition systems. *J. Log. Alg. Prog.*, 79(7):689–703, 2010.
47. Franck van Breugel. An introduction to metric semantics: operational and denotational models for programming and specification languages. *Theor. Comput. Sci.*, 258(1-2):1–98, 2001.
48. Franck van Breugel and James Worrell. A behavioural pseudometric for probabilistic transition systems. *Theor. Comput. Sci.*, 331(1):115–142, 2005.
49. Rob J. van Glabbeek. The linear time – branching time spectrum I. In Jan A. Bergstra, Alban Ponse, and Scott A. Smolka, editors, *Handbook of Process Algebra*, pages 3–99. Elsevier, 2001.
50. Farn Wang, Aloysius K. Mok, and E. Allen Emerson. Symbolic model checking for distributed real-time systems. In Jim Woodcock and Peter Gorm Larsen, editors, *FME*, volume 670 of *Lect. Notes Comput. Sci.*, pages 632–651. Springer-Verlag, 1993.
51. Uri Zwick and Mike Paterson. The complexity of mean payoff games on graphs. *Theor. Comput. Sci.*, 158(1&2):343–359, 1996.