Computing Branching Distances With Quantitative Games

Uli Fahrenberg[a,1,*], Axel Legay[b,c], Karin Quaas[d]

[a] *École polytechnique, Palaiseau, France*
[b] *Université catholique de Louvain, Belgium*
[c] *Aalborg University, Denmark*
[d] *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 [55], stochastic process algebras [47], timed automata [2], hybrid automata [1], timed variants of Petri nets [40, 54], continuous-time Markov chains [56], *etc.* Similarly, there is a number of specification formalisms for expressing quantitative properties: timed computation tree logic [45], probabilistic computation tree logic [41], metric temporal logic [48], stochastic continuous logic [4], *etc.*

*Corresponding author. École polytechnique, Laboratoire d'informatique (LIX), Bâtiment Alan Turing, 1 rue Honoré d'Estienne d'Orves, 91120 Palaiseau Cedex, France. *Email address:* ulrich.fahrenberg@polytechnique.edu

[1] This author's work is supported by the *Chaire ISC : Engineering Complex Systems* – École polytechnique – Thales – FX – DGA – Dassault Aviation – Naval Group – ENSTA Paris – Télécom Paris

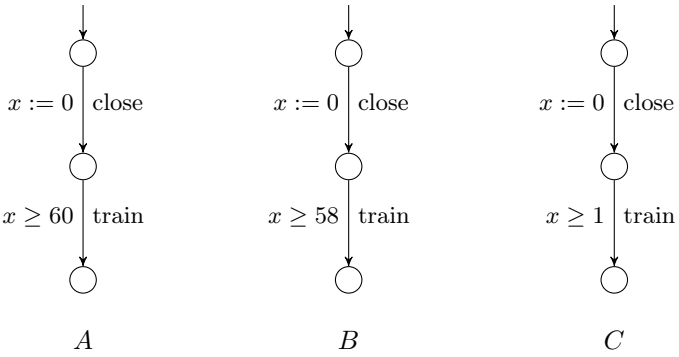

Figure 1: Three timed automata modeling a train crossing.

Quantitative verification, *i.e.*, the checking of quantitative properties for quantitative systems, has also seen rapid development: for probabilistic systems in PRISM [49] and PEPA [37], for real-time systems in Uppaal [51], RED [64], TAPAAL [10] and Romeo [36], and for hybrid systems in HyTech [43], SpaceEx [35] and HySAT [34], 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 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 [3, 5–7, 11, 13, 14, 17, 27, 32, 38, 44, 46, 58–60] 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 [28] 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 [28] generalizes the linear-time–branching-time spectrum of preorders and equivalences from van Glabbeek's [62] 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 [57] 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 [28] 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 [28] 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 [28] in Section 2. Then we show the reduction to path-building games in Sections 3 and 4 and apply this to show how to compute all common branching distances in Section 5. We collect our results in the concluding section 6. The contributions of this paper are the following:

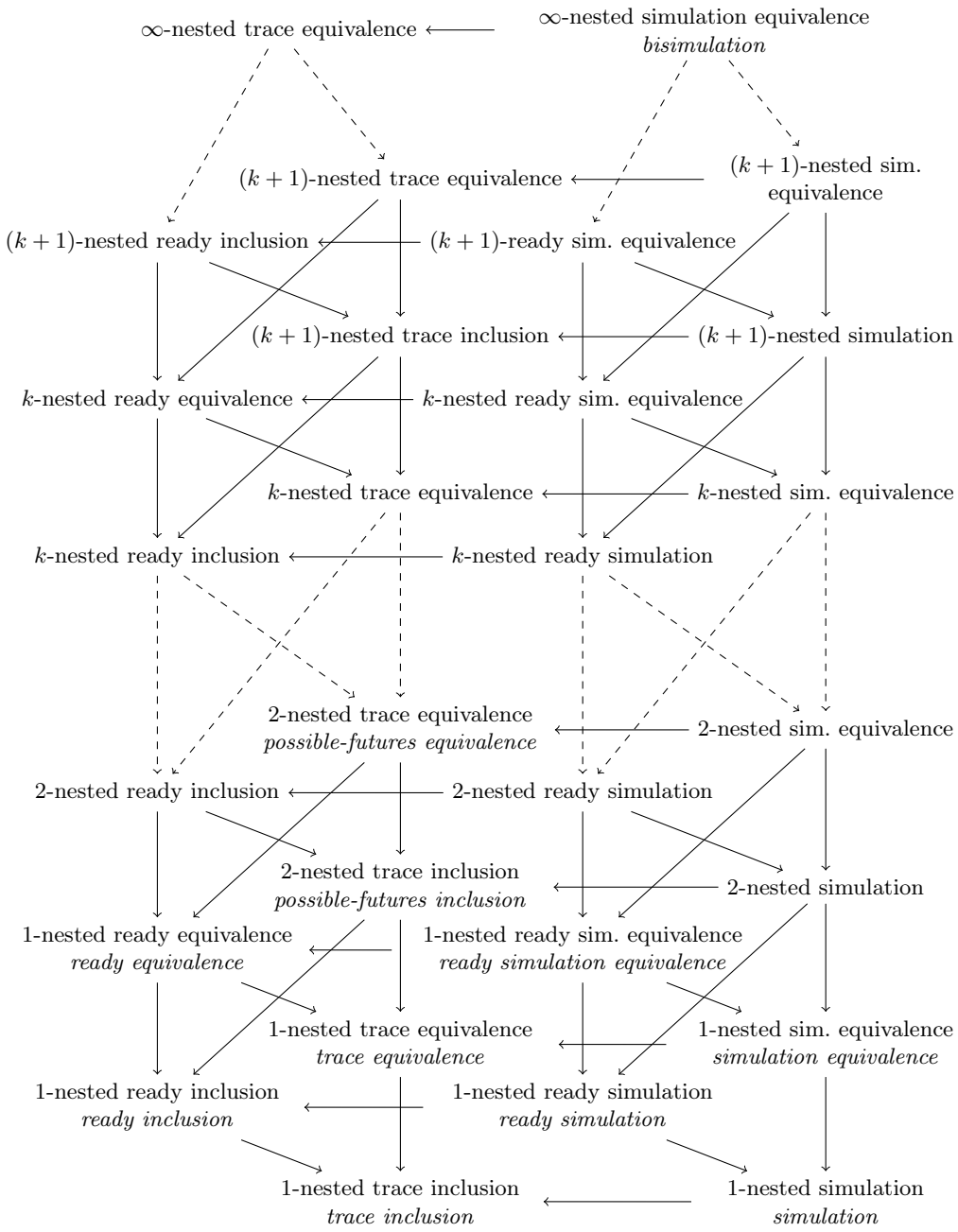

Figure 2: The quantitative linear-time–branching-time spectrum from [28]. The nodes are different system distances, and an edge $d_1 \longrightarrow d_2$ or $d_1 \dashrightarrow d_2$ indicates that $d_1(s,t) \geq d_2(s,t)$ for all states s, t, and that d_1 and d_2 in general are topologically inequivalent.

(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.

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

This paper is an extended version of [31] which has been presented at the 16th International Colloquium on Theoretical Aspects of Computing (ICTAC) in Hammamet, Tunisia. Compared to that paper, the present contribution extends our results to the complete quantitative branching spectrum of [28]. Whereas [31] only shows how to compute simulation and bisimulation distances, we show here how to extend the setting so that one can also compute, for example, ready simulation distances, nested simulation distances, or simulation equivalence distances.

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 $D : \Sigma^\omega \times \Sigma^\omega \to \mathbb{R}_*$, *i.e.*, a function which satisfies $D(\sigma, \sigma) = 0$ and $D(\sigma, \tau) + D(\tau, \upsilon) \geq D(\sigma, \upsilon)$ for all $\sigma, \tau, \upsilon \in \Sigma^\omega$.

The following examples comprise most of the different trace distances which have been used in different applications. Some of them are based on a given arbitrary *label distance* $d : \Sigma \times \Sigma \to \mathbb{R}_*$, but this is not needed in general. We refer to [28] 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. $D_{\sup}(\sigma, \tau) = \sup_{n \geq 0} d(\sigma_n, \tau_n)$, for any given label distance $d : \Sigma \times \Sigma \to \mathbb{R}_*$. This measures the greatest individual symbol distance in the traces and has been used for quantitative verification in, among others, [15, 16, 18, 25, 50, 58].

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, [11, 25, 50, 58].

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, [11, 12]. Both discounted and limit-average distances are well-known from the theory of discounted and mean-payoff games [22, 65].

The Cantor trace distance. $D_{\text{C}}(\sigma,\tau) = \frac{1}{1+\inf\{n \mid \sigma_n \neq \tau_n\}}$. This measures the (inverse of the) length of the common prefix of the traces and has been used for verification in [20]. (A variant of this uses $D_{\text{C}}(\sigma,\tau) = 2^{-\inf\{n \mid \sigma_n \neq \tau_n\}}$ instead.)

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 $-$ and a complete lattice structure, for instance $\Sigma \subseteq \mathbb{R}$. As this measures differences of accumulated labels along runs, it is especially useful for real-time systems, *cf.* [26, 44, 58].

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 [28] 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 [28] was that if the given trace distance has a recursive formulation, which, as we show in [28], 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 [28] 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. Example

Figure 3 displays a toy model of a coffee machine, modeling insertions of coins (ins) and the delivery of standard (std) and extra-large (xxl) coffee. The model contains approximate annotations for the machine's energy consumption: brewing a standard coffee consumes 1 energy unit, plus/minus 10%; brewing an

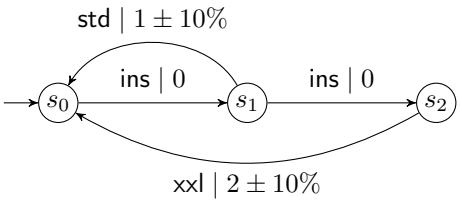

Figure 3: Toy model of a coffee machine.

extra large coffee consumes $2 \pm 10\%$ energy units. (Quite naturally, inserting coins does not consume energy.)

We may now inquire about the *robustness* of this model: given that the energy annotations are approximate, what are the deviations in energy consumption that we should expect? As a simple example, one machine might always consume 1.1 energy units at a std transition and another always 0.9, so that in an infinite run $(\mathsf{ins}, \mathsf{std}, \mathsf{ins}, \mathsf{std}, \dots)$ the two machines would exhibit a difference in energy consumption of 0.2 every second step.

We will come back to this example later; for now we only pick a particular pair of infinite traces and compute their distances depending on the trace distance from Section 2.1 one may want to use. Note that $\Sigma = \{\mathsf{ins}, \mathsf{std}, \mathsf{xxl}\} \times \mathbb{R}_{\geq 0}$; as label distance we use $d((a, x), (a, y)) = |x - y|$ and $d((a, x), (b, y)) = \infty$ for $a \neq b$.

Let σ and τ be the following infinite traces in our model:

$$\sigma = ((\mathsf{ins}, 0), (\mathsf{std}, 1.1), (\mathsf{ins}, 0), (\mathsf{std}, 0.9))^\omega$$
$$\tau = ((\mathsf{ins}, 0), (\mathsf{std}, 0.9), (\mathsf{ins}, 0), (\mathsf{std}, 1.1))^\omega$$

That is, both traces repeat the ins-std loop infinitely often, alternating between the lowest and highest possible energy consumptions.

The discrete trace distance. $D_{\text{disc}}(\sigma, \tau) = \infty$: even though the two traces are equal in their discrete components, their weights differ.

The point-wise trace distance. $D_{\sup}(\sigma, \tau) = 0.2$, detecting that their are instances in our traces at which energy consumption is off by 0.2.

The discounted trace distance. $D_+(\sigma, \tau) = 0 + \lambda \cdot 0.2 + \lambda^2 \cdot 0 + \dots = 0.2\frac{\lambda}{1-\lambda^2}$, which evaluates to 0.95 for a standard discounting factor of $\lambda = 0.9$.

The limit-average trace distance. $D_{\text{lavg}}(\sigma, \tau) = \liminf\{0, \frac{1}{2} \cdot 0.2, \frac{1}{3} \cdot 0.2, \frac{1}{4} \cdot 0.4, \frac{1}{5} \cdot 0.4, \dots\} = 0.1$, conforming to the intuition that the difference in energy consumption is 0.2 every second step.

The Cantor trace distance. $D_{\text{C}}(\sigma, \tau) = \frac{1}{2}$ (in both variants): the traces differ in their second element.

The maximum-lead trace distance. Ignoring labels and setting $\Sigma = \mathbb{R}_{\geq 0}$, we have $D_{\pm}(\sigma, \tau) = \sup\{0, 0.2, 0.2, 0, \dots\} = 0.2$: at the second step, σ acquires a "lead" in energy consumption of 0.2 compared to τ, but then at the fourth step, τ "catches up" again and the difference goes back to 0.

2.4. Quantitative Ehrenfeucht-Fraïssé Games

We review the quantitative games used in [28] to define different types of linear and branching distances for any given trace distance D.

Quantitative Simulation Games

Let $\mathcal{S} = (S, i, T)$ and $\mathcal{S}' = (S', i', T')$ be LTS and $D : \Sigma^\omega \times \Sigma^\omega \to \mathbb{R}_*$ 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 $\theta : \mathsf{Conf} \to T$ such that for all $(\pi, \pi') \in \mathsf{Conf}$ with $\theta(\pi, \pi') = (s, a, t)$, we have $\mathsf{end}(\pi) = s$. Here $\mathsf{end}(\pi)$ denotes the last state of π. Similarly, and denoting the set of minimizer configurations by Conf', a strategy for the minimizer is a mapping $\theta' : \mathsf{Conf}' \to T'$ such that for all $(\pi, \pi') \in \mathsf{Conf}'$ with $\theta'(\pi, \pi') = (s', a', t')$, $\mathsf{end}(\pi') = s'$.

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 $D^{\mathrm{sim}}(\mathcal{S}, \mathcal{S}')$, by

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

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* [57]: 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 $D^{\text{bisim}}_{\text{disc}}(\mathcal{S}, \mathcal{S}') = 0$ iff $\mathcal{S}$ and $\mathcal{S}'$ are *bisimilar*. The game which results being played is precisely the bisimulation game of [57], which also has been introduced by Fraïssé [33] and Ehrenfeucht [21] 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 $\mathsf{sc} = 0$, then the players are playing a simulation game; if there is no limit ($\mathsf{sc} = \infty$), they are playing a bisimulation game. Other limits $\mathsf{sc} \leq k$, for $k \in \mathbb{N}$, can be used to define *k-nested simulation distances*, generalizing the equivalences and preorders from [39, 42].

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 [52] and, combined with the switch counter method above, the *ready k-nested simulation distances*.

Finally, the (ready k-nested) simulation *equivalence* distances in the spectrum are given as maxima of two distances: to compute, for example, the simulation equivalence distance between $\mathcal{S}$ and $\mathcal{S}'$, one takes the maximum of the

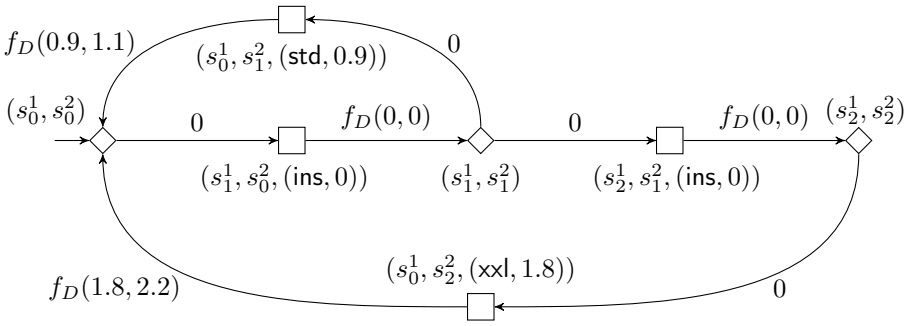

Figure 4: Game for computing simulation distance for example in Fig. 3.

simulation distance from $\mathcal{S}$ to $\mathcal{S}'$ and the one from $\mathcal{S}'$ to $\mathcal{S}$. We refer to [28] 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 first for the quantitative simulation game and then generalize this to the other branching distances in the spectrum in Section 4.

3. Reduction

In order to compute simulation distances, we translate the quantitative simulation game of the previous section to a path-building game à la Ehrenfeucht-Mycielski [22]. In the following section we will generalize the construction to cover all branching distances in the linear-time–branching-time spectrum.

Let $D : \Sigma^\omega \times \Sigma^\omega \to \mathbb{R}_*$ be a trace distance and let $\mathsf{val}_D : \mathbb{R}_*^\omega \to \mathbb{R}_*$ and $f_D : \Sigma \times \Sigma \to \mathbb{R}_*$ be functions for which it holds, for all $\sigma, \tau \in \Sigma^\infty$, that

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

We will need these functions in our translation, and we show in Section 3.2 below how to introduce them 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 $\mathcal{U} = \mathcal{U}(\mathcal{S}, \mathcal{S}') = (U, u_0, \rightarrowtriangle)$ as follows, with $U = U_1 \cup U_2$:

$$U_1 = S \times S' \qquad U_2 = S \times S' \times \Sigma \qquad u_0 = (i, i')$$

$$\begin{aligned} \rightarrowtriangle = {} & \{(s, s') \xrightarrow{0} (t, s', a) \mid (s, a, t) \in T\} \\ & \cup \{(t, s', a) \xrightarrow{f_D(a,a')} (t, t') \mid (s', a', t') \in T'\} \end{aligned}$$

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.

Figure 4 displays the game obtained by applying the construction to the example in Fig. 3. Diamond-shaped states are controlled by the maximizer and box-shaped states by the minimizer, and we have omitted the discrete label components in the f_D expressions in order not to clutter the figure.

The game on $\mathcal{U}$ is played as follows. A play begins with the maximizer choosing a transition $(u_0, a_0, u_1) \in \rightarrowtriangle$ with $u_0 = i$. Then the minimizer chooses a transition $(u_1, a_1, u_2) \in \rightarrowtriangle$. Then the maximizer chooses a transition $(u_2, a_2, u_3) \in \rightarrowtriangle$, 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 $\mathsf{val}_D(\mathcal{U}) := \mathsf{val}_D(a_0, a_1, \dots)$ 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 [22]. Formally, configurations and strategies are given as follows. A configuration of the maximizer is a path π_1 in $\mathcal{U}$ with $\mathsf{end}(\pi_1) \in U_1$, and a configuration of the minimizer is a path π_2 in $\mathcal{U}$ with $\mathsf{end}(\pi_2) \in U_2$. Denote the sets of these configurations by Conf_1 and Conf_2, respectively. A strategy for the maximizer is, then, a mapping $\theta_1 : \mathsf{Conf}_1 \to \rightarrowtriangle$ such that for all $\pi_1 \in \mathsf{Conf}_1$ with $\theta_1(\pi_1) = (u, x, v)$, $\mathsf{end}(\pi_1) = u$. A strategy for the minimizer is a mapping $\theta_2 : \mathsf{Conf}_2 \to \rightarrowtriangle$ such that for all $\pi_2 \in \mathsf{Conf}_2$ with $\theta_2(\pi_2) = (u, x, v)$, $\mathsf{end}(\pi_2) = u$. Denoting the sets of these strategies by Θ_1 and Θ_2, respectively, we can now define

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

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).

Given that our games are zero-sum and turn-based, they are also *determined* [53, 63]; that is, the sup and inf above may be swapped so that $\mathsf{val}_D(\mathcal{U}) = \inf_{\theta_2 \in \Theta_2} \sup_{\theta_1 \in \Theta_1} \mathsf{val}_D(\sigma(\theta_1, \theta_2))$. By the next theorem, the value of $\mathcal{U}$ is precisely the simulation distance from $\mathcal{S}$ to $\mathcal{S}'$.

Theorem 1. *For all LTS $\mathcal{S}$, $\mathcal{S}'$, $D^{\mathrm{sim}}(\mathcal{S}, \mathcal{S}') = \mathsf{val}_D(\mathcal{U}(\mathcal{S}, \mathcal{S}'))$.*

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 $(\pi, \pi') \in \mathsf{Conf}$ in the simulation distance game, with $\pi = (s_0, a_0, s_1), \dots, (s_{n-1}, a_{n-1}, s_n)$ and $\pi' =$

$(s'_0, a'_0, s'_1), \ldots, (s'_{n-1}, a'_{n-1}, s'_n)$, define

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

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

Now for every strategy $\theta : \mathsf{Conf} \to T$ in the simulation distance game, define a strategy $\psi_1(\theta) = \theta_1 \in \Theta_1$ as follows. For $\pi_1 \in \mathsf{Conf}_1$, let $(\pi, \pi') = \phi_1^{-1}(\pi_1)$ and $s' = \mathsf{end}(\pi')$. Let $\theta(\pi, \pi') = (s, a, t)$ and define $\theta_1(\pi_1) = ((s, s'), 0, (t, s', a))$. Similarly we define a mapping $\psi_2 : \Theta' \to \Theta_2$ as follows. For $\theta' : \mathsf{Conf}' \to T'$ and $\pi_2 \in \mathsf{Conf}_2$, let $(\pi, \pi') = \phi_2^{-1}(\pi_2)$ with $\pi = (s_0, a_0, s_1), \ldots, (s_n, a_n, s_{n+1})$. 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 $(\theta, \theta') \in \Theta \times \Theta'$, $D(\sigma(\theta, \theta'), \sigma'(\theta, \theta')) = \mathsf{val}_D(\sigma(\psi_1(\theta), \psi_2(\theta')))$ by construction. But then

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

the third equality because ψ_1 and ψ_2 are bijections. □

3.2. Application to example trace distances

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

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

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

   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 $D = D_{\sup}$, we can let

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

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

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

$$\mathsf{val}_D(x) = \sum_{n=0}^{\infty} \sqrt{\lambda}^n x_n , \qquad f_D(a, b) = \frac{1}{\sqrt{\lambda}} d(a, b) ,$$

   then (1) holds. Hence the game on $\mathcal{U}$ is a standard discounted game [65].

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

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

Hence the game on $\mathcal{U}$ is a mean-payoff game [65]. In order to show that (1) holds, let $(y_n)_{n\geq 1}$ be the sequence $(1, 1, \frac{3}{2}, 1, \frac{5}{4}, \dots)$ and note that $\lim_{n\to\infty} y_n = 1$. Then

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

5. For the Cantor trace distance $D = D_{\mathrm{C}}$, let

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

The objective of the maximizer in this game is to reach a transition with weight 1 *as soon as possible*. (For the variant $D_{\mathrm{C}}(\sigma,\tau) = 2^{-\inf\{n \mid \sigma_n \neq \tau_n\}}$, we can set $\mathsf{val}_D(x) = 2^{-\inf\{n \mid x_n \neq 0\}-1}$ instead.)

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

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

then (1) holds.

4. General Branching Distances

In this section we show how to extend the construction of $\mathcal{U}$ from above in order to compute all branching distances in the linear-time–branching-time spectrum. We start by the bisimulation distance and then proceed to the k-nested simulation distances and their ready and equivalence extensions.

4.1. Bisimulation distance

Let $\mathcal{S} = (S, i, T)$ and $\mathcal{S}' = (S', i', T')$ be LTS and define $\mathcal{V} = \mathcal{V}(\mathcal{S}, \mathcal{S}') = (V, v_0, \rightarrowtriangle)$ as follows, with $V = V_1 \cup V_2$:

$$V_1 = S \times S' \qquad V_2 = S \times S' \times \Sigma \times \{1, 2\} \qquad v_0 = (i, i')$$

$$\begin{aligned}
\rightarrowtriangle = {} & \{(s, s') \xrightarrow{0} (t, s', a, 1) \mid (s, a, t) \in T\} \\
& \cup \{(s, s') \xrightarrow{0} (s, t', a', 2) \mid (s', a', t') \in T'\} \\
& \cup \{(t, s', a, 1) \xrightarrow{f_D(a,a')} (t, t') \mid (s', a', t') \in T'\} \\
& \cup \{(s, t', a', 2) \xrightarrow{f_D(a,a')} (t, t') \mid (s, a, t) \in T\}
\end{aligned}$$

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. The next theorem states that the value of $\mathcal{V}$ is precisely the bisimulation distance between $\mathcal{S}$ and $\mathcal{S}'$.

Theorem 2. *For all LTS $\mathcal{S}$, $\mathcal{S}'$, $D^{\mathrm{bisim}}(\mathcal{S}, \mathcal{S}') = \mathsf{val}_D(\mathcal{V}(\mathcal{S}, \mathcal{S}'))$.*

Proof. This proof is similar to the one of Theorem 1, 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). □

4.2. Nested simulation distances

Let $\mathcal{S} = (S, i, T)$ and $\mathcal{S}' = (S', i', T')$ be LTS and $k \in \mathbb{N}$. Define $\mathcal{W} = \mathcal{W}(\mathcal{S}, \mathcal{S}', k) = (W, w_0, \rightarrowtriangle)$ as follows, with $W = W_1 \cup W_2$:

$$W_1 = S \times S' \times \{1, 2\} \times \{0, \ldots, k\}$$

$$W_2 = S \times S' \times \Sigma \times \{1, 2\} \times \{0, \ldots, k\} \qquad w_0 = (i, i', 1, k)$$

$$\begin{aligned}
\rightarrowtriangle = {} & \{(s, s', 1, n) \xrightarrow{0} (t, s', a, 1, n) \mid (s, a, t) \in T, n \in \{0, \ldots, k\}\} \\
& \cup \{(s, s', 2, n) \xrightarrow{0} (s, t', a', 2, n) \mid (s', a', t') \in T', n \in \{0, \ldots, k\}\} \\
& \cup \{(s, s', 1, n) \xrightarrow{0} (s, t', a', 2, n-1) \mid (s', a', t') \in T', n \in \{1, \ldots, k\}\} \\
& \cup \{(s, s', 2, n) \xrightarrow{0} (t, s', a, 1, n-1) \mid (s, a, t) \in T, n \in \{1, \ldots, k\}\} \\
& \cup \{(t, s', a, 1, n) \xrightarrow{f_D(a,a')} (t, t', 1, n) \mid (s', a', t') \in T', n \in \{0, \ldots, k\}\} \\
& \cup \{(s, t', a', 2, n) \xrightarrow{f_D(a,a')} (t, t', 1, n) \mid (s, a, t) \in T, n \in \{0, \ldots, k\}\}
\end{aligned}$$

We now use both players' states to remember which side the maximizer is currently playing on, and also how many times she is still allowed to switch sides. The first two types of maximizer transitions do not switch sides, hence do not decrease the allowed number of switches; the third and fourth type do switch sides and hence decrease the switch counter.

Similarly to Theorem 2, one can show that $\mathsf{val}_D(\mathcal{W}(\mathcal{S}, \mathcal{S}', k))$ is precisely the k-nested simulation distance from $\mathcal{S}$ to $\mathcal{S}'$.

To compute the k-nested simulation *equivalence* distance, we can simply play two k-nested simulation games, $\mathcal{W}(\mathcal{S}, \mathcal{S}', k)$ and $\mathcal{W}(\mathcal{S}', \mathcal{S}, k)$, and compute the maximum of their values.

In order to cover the *ready* distances, we need to give the maximizer the possibility to finish the game by special ready moves. This requires an extension of our setting to Σ^∞, the set of *finite or infinite* traces over Σ. This has been done in [28], so we will not repeat it here; it also permits to lift the restriction on our LTS to be non-blocking.

Once that extension has been set up, we can adapt the definition of $\mathcal{W}$ to include minimizer states reached after the maximizer has made a ready move, so that now,

$$W_2 = S \times S' \times \Sigma \times \{1,2\} \times \{0,\dots,k\} \cup S \times S' \times \Sigma \times \{1,2\}$$

We then include four new types of transitions,

$$\{(s,s',1,n) \xrightarrow{0} (s,t',a',2) \mid (s',a',t') \in T', n \in \{1,\dots,k\}\},$$
$$\{(s,s',2,n) \xrightarrow{0} (t,s',a,1) \mid (s,a,t) \in T, n \in \{1,\dots,k\}\},$$
$$\{(t,s',a,1) \xrightarrow{f_D(a,a')} \bot \mid (s',a',t') \in T'\},$$
$$\{(s,t',a',2) \xrightarrow{f_D(a,a')} \bot \mid (s,a,t) \in T\},$$

where $\bot \in W$ is a new (deadlock) state. The first two transition types allow the maximizer to challenge with a ready move, and the last two types are possible answers of the minimizer to a ready challenge which then reach the deadlock state, so that the game finishes.

5. 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 sections. This will give us algorithms to compute all branching 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 other games introduced in the last section are very similar, and everything we say also applies to them. We also assume our games to be finite; note that the translations of the previous sections convert pairs of finite LTS to finite path-building games.

Discrete distance. The game to compute the discrete simulation distance 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

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

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 $S^* = \mathsf{cpre}^*(S) = \bigcup_{n\geq 0} \mathsf{cpre}^n(S)$. 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.

Lemma 3. $\mathsf{val}_D(\mathcal{U}) = 0$ *iff* $u_0 \notin S^*$.

Proof. As we are working with the discrete distance, we have either $\mathsf{val}_D(\mathcal{U}) = 0$ or $\mathsf{val}_D(\mathcal{U}) = \infty$. Now $u_o \in S^*$ iff the maximizer can force, using finitely many steps, the game into a state from which the minimizer only has ∞-labeled transitions, which is the same as $\mathsf{val}_D(\mathcal{U}) = \infty$. □

Point-wise distance. To compute the value of the point-wise simulation distance game, let $W = \{w_1, \ldots, w_m\}$ be the (finite) set of weights of the minimizer's transitions, ordered such that $w_1 < \cdots < w_m$. For each $i = 1, \ldots, m$, 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 $S_m \subseteq S_{m-1} \subseteq \cdots \subseteq S_1 = U_1$. For each $i = 1, \ldots, m$, compute $S_i^* = \mathsf{cpre}^*(S_i)$, then $S_m^* \subseteq S_{m-1}^* \subseteq \cdots \subseteq S_1^* = U_1$ by monotonicity of cpre.

Lemma 4. *Let p be the greatest index for which $u_0 \in S_p^*$, then $p = \mathsf{val}_D(\mathcal{U})$.*

Proof. For any k, we have $u_0 \in S_k^*$ iff the maximizer can force, using finitely many steps, the game into a state from which the minimizer only has transitions with weight at least w_k. Thus $u_0 \in S_p^*$ iff (1) the maximizer can force the minimizer into a w_p-weighted transition; (2) the maximizer *cannot* force the minimizer into a w_{p+1}-weighted transition. □

Discounted distance. The game to compute the discounted simulation distance is a standard discounted game and can be solved by standard methods [65].

Limit-average distance. For the limit-average simulation distance game, the game is a standard mean-payoff game and can be solved by standard methods, see for example [19].

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, that is,

$$S_1 = \{u_1 \in U_1 \mid \exists u_1 \xrightarrow{0} u_2 : \forall u_2 \xrightarrow{x} u_3 : x = 1\}\,.$$

Now recursively compute $S_{i+1} = S_i \cup \mathsf{cpre}(S_i)$, for $i = 1, 2, \ldots$, 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).

Lemma 5. *Let p be the least index for which $u_0 \in S_p$, or ∞ if $u_0 \notin S_i$ for all i, then $\mathsf{val}_D(\mathcal{U}) = \frac{1}{p}$ (with $\frac{1}{\infty} = 0$).*

Proof. For all i, 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 $\mathsf{val}_D(\mathcal{U}) = 0$ if there is no p for which $u_0 \in S_p$, and otherwise $\mathsf{val}_D(\mathcal{U}) = \frac{1}{p}$, 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 $\sup_{n\geq 0}\left|\sum_{i=0}^{n} x_i\right|$, *i.e.*, wants the accumulated values $\sum_{i=0}^{n} x_i$ or $-\sum_{i=0}^{n} x_i$ to exceed any prescribed bounds. A weighted game in which one player wants to keep accumulated values inside some given bounds, while the opponent wants to exceed these bounds, is called an *interval-bound energy game*. It is shown in [8] that solving general interval-bound energy games is EXPTIME-complete.

We can reduce the problem of computing maximum-lead simulation distance to an interval-bound energy game by first non-deterministically choosing a bound k and then checking whether player 1 wins the interval-bound energy game on $\mathcal{U}$ for bounds $[-k, k]$. (There is a slight problem in that in [8], energy games are defined only for *integer*-weighted transition systems, whereas we are dealing with real weights here. However, it is easily seen that the results of [8] also apply to *rational* weights and bounds; and as our transition systems are finite, one can always find a sound and complete rational approximation.)

We thus obtain a non-deterministic exponential upper bound for the time to compute maximum-lead simulation distance. Using ideas similar to the ones in [9], it should be possible to do so in *deterministic* exponential time; we leave this open for future work.

6. 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 implies the well-known fact that simulation and bisimulation are decidable in polynomial time.

Corollary 6.

1. *Discrete branching distances are computable in PTIME.*

2. *Point-wise branching distances are computable in PTIME.*

3. *Discounted branching distances are computable in $NP \cap coNP$.*

4. *Limit-average branching distances are computable in $NP \cap coNP$.*

5. *Cantor branching distances are computable in PTIME.*

6. *Maximum-lead branching distances are computable in NEXPTIME.*

We note that the *linear* part of the spectrum in Fig. 2 is given by turning the maximizer into a *blind* player, so that she cannot see the choices taken by the minimizer, see [28]. Hence we could apply the procedure laid out in the present paper to convert the games for computing linear distances into suitably defined *partial-information* quantitative path-building games. Not much, however, seems to be known about computing values of partial-information quantitative path-building games.

In the future, we intend to expand our work to also cover *quantitative specification theories*. Together with several coauthors, we have in [23, 24] developed a comprehensive setting for satisfaction and refinement distances in quantitative specification theories. Using our work in [29, 30] 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 quantitative path-building games.

Another possible extension of our work contains *probabilistic* systems, for example the probabilistic automata of [55]. A possible starting point for this is [61] which uses simple stochastic games to compute probabilistic bisimilarity.

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] Guy Avni and Orna Kupferman. Making weighted containment feasible: A heuristic based on simulation and abstraction. In Maciej Koutny and Irek Ulidowski, editors, *CONCUR*, volume 7454 of *Lect. Notes Comput. Sci.*, pages 84–99. Springer-Verlag, 2012.

[4] Adnan Aziz, Kumud Sanwal, Vigyan Singhal, and Robert K. Brayton. Model-checking continous-time Markov chains. *ACM Transactions on Computational Logics*, 1(1):162–170, 2000.

[5] Sebastian S. Bauer, Uli Fahrenberg, Line Juhl, Kim G. Larsen, Axel Legay, and Claus Thrane. Quantitative refinement for weighted modal transition systems. In Filip Murlak and Piotr Sankowski, editors, *MFCS*, volume 6907 of *Lect. Notes Comput. Sci.*, pages 60–71. Springer-Verlag, 2011.

[6] Sebastian S. Bauer, Uli Fahrenberg, Line Juhl, Kim G. Larsen, Axel Legay, and Claus Thrane. Weighted modal transition systems. *Formal Meth. Syst. Design*, 42(2):193–220, 2013.

[7] Sebastian S. Bauer, Uli Fahrenberg, Axel Legay, and Claus Thrane. General quantitative specification theories with modalities. In Edward A. Hirsch, Juhani Karhumäki, Arto Lepistö, and Michail Prilutskii, editors, *CSR*, volume 7353 of *Lect. Notes Comput. Sci.*, pages 18–30. Springer-Verlag, 2012.

[8] Patricia Bouyer, Uli Fahrenberg, Kim Guldstrand Larsen, Nicolas Markey, and Jiří 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.

[9] Thomas Brihaye, Gilles Geeraerts, Axel Haddad, and Benjamin Monmege. Pseudopolynomial iterative algorithm to solve total-payoff games and min-cost reachability games. *Acta Inf.*, 54(1):85–125, 2017.

[10] 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.

[11] Pavol Černý, Thomas A. Henzinger, and Arjun Radhakrishna. Simulation distances. *Theor. Comput. Sci.*, 413(1):21–35, 2012.

[12] Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Quantitative languages. *ACM Transactions on Computational Logic*, 11(4), 2010.

[13] Krishnendu Chatterjee, Thomas A. Henzinger, Jan Otop, and Yaron Velner. Quantitative fair simulation games. *Inf. Comput.*, 254:143–166, 2017.

[14] 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.

[15] Luca de Alfaro, Marco Faella, and Mariëlle Stoelinga. Linear and branching system metrics. *IEEE Transactions on Software Engineering*, 35(2):258–273, 2009.

[16] 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, *ICALP*, volume 2719 of *Lect. Notes Comput. Sci.*, pages 1022–1037. Springer-Verlag, 2003.

[17] Josee Desharnais, Vineet Gupta, Radha Jagadeesan, and Prakash Panangaden. Metrics for labelled Markov processes. *Theor. Comput. Sci.*, 318(3):323–354, 2004.

[18] Josée Desharnais, François Laviolette, and Mathieu Tracol. Approximate analysis of probabilistic processes. In *QEST*, pages 264–273. IEEE Computer Society, 2008.

[19] 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.

[20] 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.

[21] Andrzej Ehrenfeucht. An application of games to the completeness problem for formalized theories. *Fund. Math.*, 49:129–141, 1961.

[22] Andrzej Ehrenfeucht and Jan Mycielski. Positional strategies for mean payoff games. *Int. J. Game Theory*, 8:109–113, 1979.

[23] 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.

[24] Uli Fahrenberg, Jan Křetínský, Axel Legay, and Louis-Marie Traonouez. Compositionality for quantitative specifications. *Soft Comput.*, 22(4):1139–1158, 2018.

[25] 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.

[26] Uli Fahrenberg and Axel Legay. A robust specification theory for modal event-clock automata. In Sebastian S. Bauer and Jean-Baptiste Raclet, editors, *FIT*, volume 87 of *EPTCS*, pages 5–16, 2012.

[27] Uli Fahrenberg and Axel Legay. General quantitative specification theories with modal transition systems. *Acta Inf.*, 51(5):261–295, 2014.

[28] Uli Fahrenberg and Axel Legay. The quantitative linear-time-branching-time spectrum. *Theor. Comput. Sci.*, 538:54–69, 2014.

[29] 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.

[30] Uli Fahrenberg and Axel Legay. A linear-time-branching-time spectrum for behavioral specification theories. *J. Log. Algebr. Meth. Program.*, 110, 2020.

[31] Uli Fahrenberg, Axel Legay, and Karin Quaas. Computing branching distances using quantitative games. In Robert M. Hierons and Mohamed Mosbah, editors, *ICTAC*, volume 11884 of *Lect. Notes Comput. Sci.*, pages 59–75. Springer-Verlag, 2019.

[32] Uli Fahrenberg, Axel Legay, and Claus Thrane. The quantitative linear-time–branching-time spectrum. In Supratik Chakraborty and Amit Kumar, editors, *FSTTCS*, volume 13 of *LIPIcs*, pages 103–114. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011.

[33] Roland Fraïssé. Sur quelques classifications des systèmes de relations. *Publ. Scient. de l'Univ. d'Alger, Série A*, 1:35–182, 1954.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] Antoine Girard and George J. Pappas. Approximation metrics for discrete and continuous systems. *IEEE Trans. Automat. Contr.*, 52(5):782–798, 2007.

[39] Jan Friso Groote and Frits W. Vaandrager. Structured operational semantics and bisimulation as a congruence. *Inf. Comput.*, 100(2):202–260, 1992.

[40] 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.

[41] Hans Hansson and Bengt Jonsson. A logic for reasoning about time and reliability. *Formal Aspects Comput.*, 6(5):512–535, 1994.

[42] Matthew Hennessy and Robin Milner. Algebraic laws for nondeterminism and concurrency. *J. ACM*, 32(1):137–161, 1985.

[43] 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.

[44] Thomas A. Henzinger, Rupak Majumdar, and Vinayak S. Prabhu. Quantifying similarities between timed systems. In Paul Pettersson and Wang Yi, editors, *FORMATS*, volume 3829 of *Lect. Notes Comput. Sci.*, pages 226–241. Springer-Verlag, 2005.

[45] Thomas A. Henzinger, Xavier Nicollin, Joseph Sifakis, and Sergio Yovine. Symbolic model checking for real-time systems. *Inf. Comput.*, 111(2):193–244, 1994.

[46] Thomas A. Henzinger and Jan Otop. From model checking to model measuring. In Pedro R. D'Argenio and Hernán C. Melgratti, editors, *CONCUR*, volume 8052 of *Lect. Notes Comput. Sci.*, pages 273–287. Springer-Verlag, 2013.

[47] Jane Hillston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, 1996.

[48] Ron Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Syst.*, 2(4):255–299, 1990.

[49] 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.

[50] Kim G. Larsen, Uli Fahrenberg, and Claus Thrane. Metrics for weighted transition systems: Axiomatization and complexity. *Theor. Comput. Sci.*, 412(28):3358–3369, 2011.

[51] Kim G. Larsen, Paul Pettersson, and Wang Yi. UPPAAL in a nutshell. *Int. J. Softw. Tools Techn. Trans.*, 1(1-2):134–152, 1997.

[52] Kim G. Larsen and Arne Skou. Bisimulation through probabilistic testing. In *POPL*, pages 344–352. ACM Press, 1989.

[53] Donald A. Martin. Borel determinacy. *Annals of Mathematics*, 102(2):363–371, 1975.

[54] Philip M. Merlin and David J. Farber. Recoverability of communication protocols–implications of a theoretical study. *IEEE Trans. Commun.*, 24(9):1036 – 1043, 1976.

[55] 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.

[56] William J. Stewart. *Introduction to the Numerical Solution of Markov Chains*. Princeton University Press, 1994.

[57] 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.

[58] Claus Thrane, Uli Fahrenberg, and Kim G. Larsen. Quantitative analysis of weighted transition systems. *J. Log. Algebr. Prog.*, 79(7):689–703, 2010.

[59] 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.

[60] Franck van Breugel and James Worrell. A behavioural pseudometric for probabilistic transition systems. *Theor. Comput. Sci.*, 331(1):115–142, 2005.

[61] Franck van Breugel and James Worrell. The complexity of computing a bisimilarity pseudometric on probabilistic automata. In Franck van Breugel, Elham Kashefi, Catuscia Palamidessi, and Jan Rutten, editors, *Horizons of the Mind. A Tribute to Prakash Panangaden*, volume 8464 of *Lect. Notes Comput. Sci.*, pages 191–213. Springer-Verlag, 2014.

[62] 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.

[63] John von Neumann. Zur Theorie der Gesellschaftsspiele. *Mathematische Annalen*, 100:295–320, 1928.

[64] 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.

[65] Uri Zwick and Mike Paterson. The complexity of mean payoff games on graphs. *Theor. Comput. Sci.*, 158(1&2):343–359, 1996.