# Simulation Distances*

Pavol Černý, Thomas A. Henzinger, and Arjun Radhakrishna

IST Austria

**Abstract.** Boolean notions of correctness are formalized by preorders on systems. Quantitative measures of correctness can be formalized by real-valued distance functions between systems, where the distance between implementation and specification provides a measure of "fit" or "desirability." We extend the simulation preorder to the quantitative setting, by making each player of a simulation game pay a certain price for her choices. We use the resulting games with quantitative objectives to define three different simulation distances. The *correctness distance* measures how much the specification must be changed in order to be satisfied by the implementation. The *coverage distance* measures how much the implementation restricts the degrees of freedom offered by the specification. The *robustness distance* measures how much a system can deviate from the implementation description without violating the specification. We consider these distances for safety as well as liveness specifications. The distances can be computed in polynomial time for safety specifications, and for liveness specifications given by weak fairness constraints. We show that the distance functions satisfy the triangle inequality, that the distance between two systems does not increase under parallel composition with a third system, and that the distance between two systems can be bounded from above and below by distances between abstractions of the two systems. These properties suggest that our simulation distances provide an appropriate basis for a quantitative theory of discrete systems. We also demonstrate how the robustness distance can be used to measure how many transmission errors are tolerated by error correcting codes.

## 1 Introduction

Standard verification systems return a boolean answer that indicates whether a system satisfies its specification. However, not all correct implementations are equally good, and not all incorrect implementations are equally bad. There is thus a natural question whether it is possible to extend the standard specification frameworks and verification algorithms to capture a finer and more quantitative view of the relationship between specifications and systems.

We focus on extending the notion of simulation to the quantitative setting. For reactive systems, the standard correctness requirement is that all executions

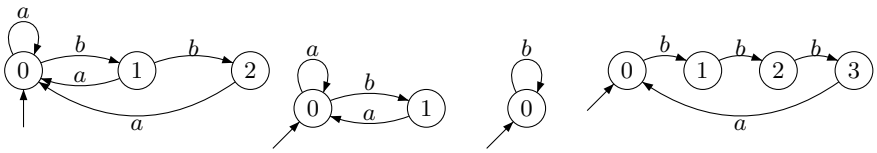
* This work was partially supported by the European Union project COMBEST and the European Network of Excellence ArtistDesign.

of an implementation have to be allowed by the specification. Requiring that the specification simulates the implementation is a stricter condition, but it is computationally less expensive to check. The simulation relation defines a preorder on systems. We extend the simulation preorder to a distance function that given two systems, returns a real-valued distance between them.

Let us consider the definition of simulation of an implementation $I$ by a specification $S$ as a two-player game, where Player 1 (the implementation) chooses moves (transitions) and Player 2 (the specification) tries to match each move. The goal of Player 1 is to prove that simulation does not hold, by driving the game into a state from which Player 2 cannot match the chosen move; the goal of Player 2 is to prove that there exists a simulation relation, by playing the game forever. In order to extend this definition to capture how "good" (or how "bad") the simulation is, we make the players pay a certain price for their choices. The goal of Player 1 is then to maximize the cost of the game, and the goal of Player 2 is to minimize it. The cost is given by an objective function, such as the limit average of transition prizes. For example, for incorrect implementations, i.e., those for which the specification $S$ does not simulate the implementation $I$, we might be interested in how often the specification (Player 2) cannot match an implementation move. We formalize this using a game with a limit-average objective between modified systems. The specification is allowed to "cheat," by following a non-existing transition, while the implementation is left unmodified. More precisely, the specification is modified by giving the transitions from the original system a weight of 0, and adding new "cheating" transitions with a non-zero positive weight. As Player 2 is trying to minimize the value of the game, she is motivated not to cheat. The value of the game measures how often the specification can be forced to cheat by the implementation, that is, how often the implementation violates the specification (i.e., commits an error) in the worst case. We call this distance function *correctness*.

Let us consider the examples in Figure 1. We take the system $S_1$ as the specification. The specification allows at most two symbols $b$ to be output in the row. Now let us consider the two incorrect implementations $I_3$ and $I_4$. The implementation $I_3$ outputs an unbounded number of $b$'s in a row, while the implementation $I_4$ can output three $b$'s in a row. The specification $S_1$ will thus not be able to simulate either $I_3$ or $I_4$, but $I_4$ is a "better" implementation in the sense that it violates the requirement to a smaller degree. We capture this by allowing $S_1$ to cheat in the simulation game by taking an existing edge while outputting a different symbol. When simulating the system $I_3$, the specification $S_1$ will have to output a $b$ when taking the edge from state 2 to state 0. This cheating transition will be taken every third move while simulating $I_3$. The correctness distance from $S_1$ to $I_3$ will therefore be $1/3$. When simulating $I_4$, the specification $S_1$ needs to cheat only one in four times—this is when $I_4$ takes a transition from its state 2 to state 3. The distance from $S_1$ to $I_4$ will be $1/4$.

Considering the implementation $I_2$ from Figure 1, it is easy to see that it is correct with respect to the specification $S_1$. The correctness distance would thus be 0. However, it is also easy to see that $I_2$ does not include all behav-




(a) $S_1$ (b) $I_2$ (c) $I_3$ (d) $I_4$

**Fig. 1.** Example Systems

iors allowed by $S_1$. Our second distance function, *coverage*, is the dual of the correctness distance. It measures how many of the behaviors allowed by the specification are actually implemented by the implementation. This distance is obtained as the value for the implementation in a game in which $I$ is required to simulate $S$, with the implementation being allowed to cheat. Our third distance function is called *robustness*. It measures how robust the implementation $I$ is with respect to the specification $S$ in the following sense: we measure how often the implementation can make an unexpected error (i.e., it performs a transition not present in its transition relation), with the resulting behavior still being accepted by the specification. Unexpected errors could be caused, for example, by a hardware problem, by a wrong environment assumption, or by a malicious attack. Robustness measures how many such unexpected errors are tolerated.

In addition to safety specifications, we consider liveness specifications given by weak (Büchi) fairness constraints or strong (Streett) fairness constrains. In order to define distances to liveness specifications, the notion of quantitative simulation is extended to *fair* quantitative simulation. We study variations of the correctness, coverage, and robustness distances using limit-average and discounted objective functions. Limit-average objectives measure the long-run frequency of errors, whereas discounted objectives count the number of errors and give more weight to earlier errors than later ones.

The correctness, coverage, and robustness distances can be calculated by solving the value problem in the corresponding games. Without fairness requirements, we obtain limit-average games or discounted games with constant weights. The values of such games can be computed in polynomial time [20]. We obtain polynomial complexity also for distances between systems with weak-fairness constraints, whereas for strong-fairness constrains, the best known algorithms require exponential time.

We present composition and abstraction techniques that are useful for computing and approximating simulation distances between large systems. Finally, we present an application of the robustness distance. We consider error correction systems for transmitting data over noisy channels. Three implementations based on the Hamming code, triple modular redundancy, and no error correction with different robustness properties are analyzed.

*Related work* Weighted automata [4, 10] provide a way to assign values to words, and to languages defined by finite-state systems. Distances between systems can be defined using weighted automata, analogically to boolean language inclusion. However, the complexity of computation of such distance is not known [4]. Our solution of using a quantitative version of simulation games corresponds in the

boolean case to the choice of using simulation instead of language inclusion. There have been several attempts to give a mathematical semantics to reactive processes which is based on quantitative metrics rather than boolean preorders [18, 6]. In particular for probabilistic processes, it is natural to generalize bisimulation relations to bisimulation metrics [9, 19], and similar generalizations can be pursued if quantities enter not through probabilities but through discounting [7] or continuous variables [2] (this work uses the Skorohod metric on continuous behaviors to measure the distance between hybrid systems). We consider distances between purely discrete (nonprobabilistic, untimed) systems, and our distances are directed rather than symmetric (based on simulation rather than bisimulation). Software metrics measure properties such as lines of code, depth of inheritance (in an object-oriented language), number of bugs in a module or the time it took to discover the bugs (see for example [12, 16]). These functions measure syntactic properties of the source code, and are fundamentally different from our distance functions that capture the difference in the behavior (semantics) of programs.

## 2 Quantitative Simulation Games

**Transition Systems.** A *transition system* is a tuple $\langle S, \Sigma, E, s_0 \rangle$ where $S$ is a finite set of states, $\Sigma$ is a finite alphabet, $E \subseteq S \times \Sigma \times S$ is a set of labeled transitions, and $s_0$ is the initial state. We require that for every $s \in S$, there exists a transition from $s$. The set of all transition systems is denoted by $\mathcal{S}$. A *weighted transition system* is a transition system along with a weight function $v$ from $E$ to $\mathbb{Q}$. A run in a transition system $T$ is an infinite path $\rho = \rho_0\sigma_0\rho_1\sigma_1\rho_2\sigma_2 \ldots \in (S \cdot \Sigma)^\omega$ where $\rho_0 = s_0$ and for all $i$, $(\rho_i, \sigma_i, \rho_{i+1}) \in E$.
*Fairness Conditions.* A *Büchi (weak fairness) condition* for a (weighted) transition system is set of states $F \subseteq S$. Given a Büchi condition $F$ and a run $\rho = \rho_0\sigma_0\rho_1\sigma_1 \ldots$ of a transition system, the run $\rho$ is *fair* iff $\forall n \geq 0 : (\exists i > n : \rho_i \in F)$. A *Streett (strong fairness) condition* for a (weighted) transition system is a set of request-response pairs $F = \{\langle E_1, F_1\rangle, \langle E_2, F_2\rangle, \ldots, \langle E_d, F_d\rangle\}$ where each $E_i, F_i \in 2^S$. Given a Streett condition, a run $\rho = \rho_0\sigma_0\rho_1\sigma_1 \ldots$ is *fair* iff $\forall k \leq d : \big((|\{i \mid \rho_i \in E_k\}| = \infty) \Rightarrow (|\{i \mid \rho_i \in F_k\}| = \infty)\big)$. We denote a transition system $A$ with a fairness condition $F$ as $A^F$.
**Game Graphs.** A *game graph* $G$ is a tuple $\langle S, S_1, S_2, \Sigma, E, s_0 \rangle$ where $S$, $\Sigma$, $E$ and $s_0$ are as in transition systems and $(S_1, S_2)$ is a partition of $S$. The choice of the next state is made by Player 1 (Player 2) when the current state is in $S_1$ (respectively, $S_2$). A *weighted game graph* is a game graph along with a weight function $v$ from $E$ to $\mathbb{Q}$. A run in the game graph $G$ is called a *play*. The set of all plays is denoted by $\Omega$.

When the two players represent the choices internal to a system, we call the game graph an *alternating transition system*. We only consider alternating transition systems where the transitions from Player 1 states go only to Player 2 states and vice-versa. We use $A^F$ to denote an alternating transition system $A$ with fairness condition $F$.

**Strategies.** Given a game graph $G$, a *strategy* for Player 1 is a function $\pi : (S \cdot \Sigma)^* S_1 \rightarrow S \times \Sigma$ such that $\forall s_0\sigma_0 s_1\sigma_1 \ldots s_i \in (S \cdot \Sigma)^* S_1$, we have that if $\pi(s_0\sigma_0 s_1\sigma_1 \ldots s_i) = (s, \sigma)$, then $(s_i, \sigma, s) \in E$. A strategy for Player 2 is defined similarly. The set of all strategies for Player $p$ is denoted by $\Pi_p$. A play $\rho = \rho_0\sigma_0\rho_1\sigma_1\rho_2\sigma_2 \ldots$ *conforms* to a player $p$ strategy $\pi$ if $\forall i \geq 0 : (\rho_i \in S_p \implies : (\rho_{i+1}, \sigma_{i+1}) = \pi(\rho_0\sigma_0\rho_1\sigma_1 \ldots \rho_i))$. The *outcome* of strategies $\pi_1$ and $\pi_2$ is the unique play $out(\pi_1, \pi_2)$ that conforms to both $\pi_1$ and $\pi_2$.

Two restricted notions of a strategy are sufficient for many classes of games. A *memoryless strategy* is one where the value of the strategy function depends solely on the last state in the history, whereas a *finite-memory strategy* is one where the necessary information about the history can be summarized by a finite amount of information.

**Games and Objectives.** A *game* is a game graph and a boolean or quantitative objective. A *boolean objective* is a function $\Phi : \Omega \rightarrow \{0, 1\}$ and the goal of Player 1 in a game with objective $\Phi$ is to choose a strategy so that, no matter what Player 2 does, the outcome maps to 1; and the goal of Player 2 is to ensure that the outcome maps to 0. A *quantitative objective* is a *value* function $f : \Omega \rightarrow \mathbb{R}$ and the goal of Player 1 is to maximize the value $f$ of the play, whereas the goal of Player 2 is to minimize it. We only consider quantitative objectives with which map plays to values in $[0, 1]$. Given a boolean objective $\Phi$, a play $\rho$ is *winning* for Player 1 (Player 2) if $\Phi(\rho) = 1$ ($\Phi(\rho) = 0$). A strategy $\pi$ is a *winning strategy* for Player $p$ if every play conforming to $\pi$ is winning for Player $p$.

For a quantitative objective $f$, the value of the game for Player 1 is defined as the supremum of the values of plays attainable against any Player 2 strategy, i.e., $\sup_{\pi_1 \in \Pi_1} \inf_{\pi_2 \in \Pi_2} f(out(\pi_1, \pi_2))$. The value of the game for Player 2 is defined analogously. A strategy is an *optimal strategy* for a player if it assures a outcome equal to her value of the game. Similarly, a strategy is an *$\epsilon$-optimal strategy* for a maximizing (resp. minimizing) player if it assures an outcome that is no more that $\epsilon$ smaller (resp. larger) than the value of the game.

We consider $\omega$-regular boolean objectives and the following quantitative objectives. Given a game graph with the weight function $v$ and a play $\rho = \rho_0\rho_1\rho_2 \ldots$, for all $i \geq 0$, let $v_i = v((\rho_i, \sigma_i, \rho_{i+1}))$.

- $LimAvg(\rho) = \liminf_{n\rightarrow\infty} \frac{1}{n} \cdot \sum_{i=0}^{n-1} v_i$
- $Disc_\lambda(\rho) = \liminf_{n\rightarrow\infty} (1 - \lambda) \cdot \sum_{i=0}^{n-1} \lambda^i \cdot v_i$ where $0 < \lambda < 1$.

*LimAvg* is the long-run average of the weights occurring in a play, whereas $Disc_\lambda$ is the discounted sum of the weights. Therefore, *LimAvg* gives more importance to the infinite suffix of a play whereas $Disc_\lambda$ gives more importance to the finite prefix of a play.

Note that for *LimAvg* and *Disc* objectives, optimal memoryless strategies exist for both players [11, 20]. Also, for qualitative objectives specified as Büchi conditions, memoryless winning strategies exist for both players, and for other $\omega$-regular conditions, finite-memory winning strategies exist.

Also, consider the following family of objectives where a boolean $\omega$-regular objective and a quantitative objective $f$ are combined as follows. If a play $\rho$

satisfies the boolean objective, then the value of $\rho$ is the value according to $f$; otherwise, the value of the $\rho$ is the maximum possible value of $f$ (in our case, it is always 1). When $f = \mathit{LimAvg}$ and the $\omega$-regular objective is a parity objective, $\epsilon$-optimal finite-memory strategies exist [5]. This result can be extended to arbitrary $\omega$-regular objectives as all $\omega$-regular objectives can be expressed as parity objectives with the *latest appearance records* memory [13]. Such objectives are called $\omega$-regular *LimAvg* objectives.

### 2.1 Qualitative Simulation Games

The simulation preorder [17] is a useful and polynomially computable relation to compare two transition systems. In [1] this relation was extended to alternating simulation between alternating transition systems. For systems with fairness conditions, the simulation relation was extended to fair simulation in [15]. These relations can be computed by solving games with boolean objectives.

**Simulation and Alternating Simulation.** Consider two transition systems $A = \langle S, \Sigma, E, s_0\rangle$ and $A' = \langle S', \Sigma, E', s'_0\rangle$. The system $A'$ *simulates* the system $A$ if there exists a relation $H \subseteq S \times S'$ such that (a) $(s_0, s'_0) \in H$; and (b) $\forall s, t \in S, s' \in S' : (s, s') \in H \wedge (s, \sigma, t) \in E \Rightarrow (\exists t' : (s', \sigma, t') \in E' \wedge (s', t') \in H)$.

For two alternating transition systems $A = \langle S, S_1, S_2, \Sigma, E, s_0\rangle$ and $A' = \langle S', S'_1, S'_2, \Sigma, E', s'_0\rangle$, *alternating simulation* of $A$ by $A'$ holds if there exists a relation $H \subseteq S \times S'$ such that $(s_0, s'_0) \in H$ and $\forall s \in S, s' \in S' : (s, s') \in H \Rightarrow (s \in S_1 \Leftrightarrow s' \in S'_1)$; and

- $\forall s \in S, s' \in S' : ((s, s') \in H \wedge s \in S_1) \Rightarrow \forall (s, \sigma, t) \in E : (\exists (s', \sigma, t') \in E' : (t, t') \in H)$.
- $\forall s \in S, s' \in S' : ((s, s') \in H \wedge s \in S_2) \Rightarrow \exists (s', \sigma, t') \in E' : (\forall (s, \sigma, t) \in E : (t, t') \in H)$.

**Simulation and Alternating Simulation Games.** Given two (alternating) transition systems, $A$ and $A'$, we can construct a game such that, (alternating) simulation of $A$ by $A'$ holds if and only if Player 2 has a winning strategy in the game

Given two weighted transition systems $A$ and $A'$ with the same alphabet, we define the corresponding *quantitative simulation game graph* $G_{A,A'}$ as $\langle S \times (\Sigma \cup \{\#\}) \times S' \cup \{s_{\text{err}}\}, S_1^G, S_2^G, \Sigma, E^G, (s_0, \#, s'_0)\rangle$ where $S_1^G = (S \times \{\#\} \times S') \cup \{s_{\text{err}}\}$ and $S_2^G = (S \times \Sigma \times S')$. Each transition of the game graph corresponds to a transition in either $A$ or $A'$ as follows:

- $((s, \#, s'), \sigma, (t, \sigma, s')) \in E^G \Leftrightarrow (s, \sigma, t) \in E$
- $((s, \sigma, s'), \sigma, (s, \#, t')) \in E^G \Leftrightarrow (s', \sigma, t') \in E'$

For each of the above transitions, the weight is the same as the weight of the corresponding transition in $A$ or $A'$. If there is no outgoing transition from a particular state, transitions to $s_{\text{err}}$ are added with all symbols. The state $s_{\text{err}}$ is a sink with transitions to itself on all symbols. Each of these transitions has weight 1 (the maximum possible value of a quantitative objective).

For classical simulation games, we consider the same game graph without weights. The objective for Player 2 is to reach $s_{\text{err}}$ and for Player 1 to avoid it. Intuitively, in every state, Player 1 chooses a transition of $A$ and Player 2 has to

match it by picking a transition of $A'$. If Player 2 cannot match at some point, Player 1 wins that play. It is easy to see that $A'$ simulates $A$ iff there is a winning strategy for Player 2 in this game.

We can extend the simulation game to an alternating simulation game. We informally define the *quantitative alternating simulation game graph*. The formal definition can be found in the companion report [3]. Given two quantitative alternating transition systems $A$ and $A'$ , we define the quantitative alternating simulation game graph $H_{A,A'}$ as follows. If $A$ is at state $s$ and $s \in S_1$, Player 1 chooses a transition of $A$ and Player 2 has to match it with a transition of $A'$; and if $A$ is at $s$ and $s \in S_2$, Player 2 has to choose a transition of $A'$ and Player 1 has to choose a transition of $A$ to match it. If there cannot be a match, the control moves to the error state $s_{\text{err}}$. As before, the transitions have the same weight as in the individual systems.

We consider the game graph without weights to define the alternating simulation game $\mathcal{H}^{A,A'}$ and the objective of the Player 1 is to ensure that the play reaches $s_{\text{err}}$. It can be seen that alternating simulation holds iff there exists a winning strategy for Player 2 .

**Fair Simulation.** Given two (alternating) transitions systems with fairness conditions $A^F$ and $A'^{F'}$, the fair simulation game is played in the same game graph $G_{A,A'}$ ($H_{A,A'}$) as the simulation game. However, in addition to matching the symbol in each step, Player 2 has to ensure that if the sequence of transitions of $A$ chosen by Player 1 satisfies the fairness condition $F$, then the sequence of $A'$ transitions chosen satisfy the fairness condition $F'$.

### 2.2 Quantitative Simulation Games

We define a generalized notion of simulation games called quantitative simulation games where the simulation objectives are replaced by quantitative objectives.

**Quantitative Simulation Games.** Given two quantitative (alternating) transition systems $A$ and $A'$, and $f \in \{LimAvg, Disc_\lambda\}$, the *quantitative (alternating) simulation game* is played on the quantitative (alternating) simulation game graph $G_{A,A'}$ ($H_{A,A'}$) with the objective of Player 1 being to maximize the $f$ value of the play. We denote this game as $\mathcal{Q}^f_{A,A'}$ ($\mathcal{P}^f_{A,A'}$).

**Quantitative Fair Simulation Games.** Analogous to quantitative (alternating) simulation games, the fair versions between two transition systems with fairness conditions. The quantitative objective for this game is the $\omega$-regular *LimAvg* objective which is the combination of *LimAvg* objective and the boolean fair (alternating) simulation game objective.

We do not use $f = Disc_\lambda$ along with fairness conditions as the two objectives are independent. The $Disc_\lambda$ objectives mainly consider the finite prefix of a play, whereas fairness conditions consider only the infinite suffix.

### 2.3 Modification Schemes

We will use quantitative simulation games to measure various properties of systems. For computing these properties, we need to use small modifications of

the original systems. For example, when trying to compute the distance as the number of errors an implementation commits, we add to the specification some error recovery behavior. However, we impose strict rules on these modifications to ensure that the modified system retains the structure of the original system.

A *modification scheme* is a function $m$ from transition systems to quantitative (alternating) transition systems, which can be computed using the following steps: (a) Edges may be added to the transition system and each state may be replaced by a local subgraph. All edges of the graph have to be preserved; (b) Every edge of the system is associated with a weight from $\mathbb{Q}$. We present two examples of modification schemes.

**Output Modification.** This scheme is used to add behavior to a system that allows it to output an arbitrary symbol while moving to a state specified by an already existing transition. For every transition $(s, \sigma, s')$, transitions with different symbols are added to the system i.e., $\{(s, \alpha, s') \mid \alpha \in \Sigma\}$. These transitions are given a weight of 2 to prohibit their free use. All other transitions have the weight zero. Given a system $T$, we denote the modified system as $\mathit{OutMod}(T)$.

**Error Modification.** In a perfectly functioning system, errors may occur due to unpredictable events. We model this with an alternating transition system with one player modeling the original system (Player 1) and the other modeling the controlled error (Player 2). At every state, Player 2 chooses whether or not a error occurs by choosing one of the two successors. From one of these states, Player 1 can choose the original successors of the state and from the other, she can choose either one of the original successors or one of the error transitions. We penalize Player 2 for the choice of not allowing errors to happen.

Given $T = \langle S, \Sigma, E, s_0 \rangle$ we define $\mathit{ErrMod}(T)$ to be the quantitative alternating transition system obtained by replacing each state $s$ by the graph in Figure 2. If an error is allowed (modeled by the $c$ edge), then all transitions that differ from original transitions only in the symbol are added (represented by $X(s)$ in Figure 2). Only the transitions labeled $\neg c$ are given the weight 2. The rest are given the weight 0. The system $\mathit{ErrMod}_\emptyset(T)$ denotes a system where no additional transitions where introduced, only the states were replaced by a subgraph from Figure 2 (with $X$ being the empty set).

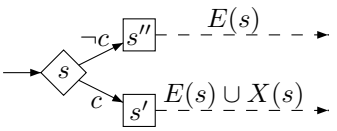
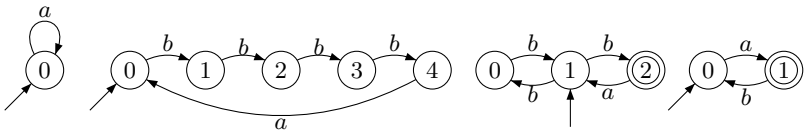



**Fig. 2.** Graph for *ErrMod*

In addition to the above schemes, we define the trivial modification scheme *NoMod* where no changes are made except to give every edge the weight 0.

## 3 Simulation Distances

**Correctness** Given a specification $T_2$ and an implementation $T_1$, such that $T_1$ is incorrect with respect to $T_2$, the correctness distance measures the degree of "incorrectness" of $T_1$. Even a single nonconformant behavior can destroy the boolean simulation relation. Here we present a game which is not as strict and




(a) $I_1$ (b) $I_5$ (c) $S_L$ (d) $I_L$

**Fig. 3.** Example Systems

measures the minimal number of required errors, i.e. the minimal number of times the specification has to use nonmatching symbols when simulating the implementation.

**Definition 3.1** (Correctness distance)**.** *Let $f = LimAvg$ or $f = Disc_\lambda$. The* correctness distance $d^f_{cor}(T_1, T_2)$ *from system $T_1$ to system $T_2$ is the Player 1 value of the quantitative simulation game $\mathcal{C}^f_{T_1,T_2} = \mathcal{Q}^f_{NoMod(T_1),OutMod(T_2)}$.*

The game $\mathcal{C}$ can be intuitively understood as follows. Given two systems $T_1$ and $T_2$, we are trying to simulate the system $T_1$ by $T_2$, but the specification $T_2$ is allowed to make errors, to "cheat", but she has to pay a price for such a choice. As the simulating player is trying to minimize the value of the game, she is motivated not to cheat. The value of the game can thus be seen as measuring how often she can be forced to cheat, that is, how often on average the implementation commits an error. If the implementation is correct ($T_2$ simulates $T_1$), then the correctness distance is 0. The value of the game is either the *LimAvg* or the *Disc* of the number of errors. If the objective $f$ is *LimAvg*, then the value is the long run average of the errors, whereas if the objective $f$ is $Disc_\lambda$, the errors which occur earlier are given more importance and the value is the discounted sum of the position of the errors. Therefore, the *Disc* and *LimAvg* games are concerned with prefixes and infinite suffixes of the behaviors respectively.

We present a few example systems and their distances here to demonstrate the fact that the above game measures distances that correspond to intuition. In Figure 3 and Figure 1, $S_1$ is the specification system against which we want to measure the systems $I_1$ through $I_5$. In this case, the specification says that there cannot be more than two $b$'s in a row. Also, we have a specification with a liveness condition $S_L$ against which we want to measure the implementation $I_L$. The distances between these systems according to the *LimAvg* correctness game are summarized in Table 1.

| $T_1$ | $T_2$ | $d^{LimAvg}_{\text{cor}}(T_1,T_2)$ | $d^{LimAvg}_{\text{cov}}(T_1,T_2)$ | $d^{LimAvg}_{\text{rob}}(T_1,T_2)$ |
|---|---|---|---|---|
| $S_1$ | $S_1$ | 0 | 0 | 1 |
| $S_1$ | $I_1$ | 0 | 2/3 | 1/3 |
| $S_1$ | $I_2$ | 0 | 1/3 | 2/3 |
| $S_1$ | $I_3$ | 1/3 | 1 | 1 |
| $S_1$ | $I_4$ | 1/4 | 3/4 | 1 |
| $S_1$ | $I_5$ | 1/5 | 4/5 | 1 |
| $S_L$ | $I_L$ | 1/2 | 1 | 1 |

**Table 1.** Distances according to the Correctness, Coverage and Robustness game

Among the systems which do not satisfy the specification $S_1$, i.e. $I_3$, $I_4$ and $I_5$, we showed in the introduction that the distance from $I_3$ to $S_1$ is 1/3, while

the distance from $I_4$ to $S_1$ is 1/4. However, surprisingly the distance from $I_5$ to $S_1$ is less than the distance from $I_4$. In fact, the distances reflect on the long run the number of times the specification has to err to simulate the implementation.

In case of the specification $S_L$ and implementation $I_L$ with liveness conditions, the specification can take the left branch to state 0 to get a penalty of $\frac{1}{2}$ or take the right branch to state 2 to get a penalty of 1. However, it needs to take the right branch infinitely often to satisfy the liveness condition. To achieve the distance of $\frac{1}{2}$, the specification needs infinite memory so that it can take the right branch lesser and lesser number of times. In fact, if the specification has a strategy with finite-memory of size $m$, it can achieve a distance of $\frac{1}{2} + \frac{1}{2m}$.

**Coverage** We present the dual game of the one presented above. Here, we measure the behaviors that are present in one system but not in the other system. Given a specification $T_2$ and an implementation $T_1$, the coverage distance corresponds to the behavior of the specification which is farthest from any behaviour of the implementation. Hence, we have that the coverage distance from a system $T_1$ to a system $T_2$ is the correctness distance from $T_2$ to $T_1$.

**Definition 3.2** (Coverage distance)**.** *Let $f = LimAvg$ or $f = Disc_\lambda$. The* coverage distance $d^f_{cov}(T_1, T_2)$ *from system $T_1$ to system $T_2$ is the Player 1 value of the quantitative simulation game $\mathcal{V}^f_{T_1,T_2} = \mathcal{Q}^f_{NoMod(T_2),OutMod(T_1)}$.*

$\mathcal{V}$ measures the distance from $T_1$ to $T_2$ as the minimal number of errors that have to be committed by $T_1$ to cover all the behaviors of $T_2$. We present examples of systems and their distances according to $\mathcal{V}^{LimAvg}$. We use the example systems in Figures 3 and 1. The distances are summarized in Table 1.

**Robustness** Given a specification system and a correct implementation, the notion of robustness presented here is a measure of the number of errors by the implementation that makes it nonconformant to the specification. The more such errors tolerated by the specification, the more robust the implementation. In other words, the distance measures the number of critical points, or points where an error will lead to an unacceptable behavior. The lower the value of the robustness distance, the more robust an implementation is. In case of an incorrect implementation, the simulation of the implementation does not hold irrespective of implementation errors. Hence, in that case, the robustness distance will be 1.

**Definition 3.3** (Robustness distance)**.** *Let $f = LimAvg$ or $f = Disc_\lambda$. The* robustness distance $d^f_{rob}(T_1, T_2)$ *from system $T_1$ to system $T_2$ is the Player 1 value of the quantitative alternating simulation game $\mathcal{R}^f_{T_1,T_2} = \mathcal{P}^f_{ErrMod(T_1),ErrMod_\emptyset(T_2)}$.*

The game $\mathcal{R}_{ErrMod(T_1),ErrMod_\emptyset(T_2)}$ is played in the following steps: (a) The specification $T_2$ chooses whether the implementation $T_1$ is allowed to make an error; (b) The implementation chooses a transition on the implementation system. It is allowed to err based on the specification choice in the previous step; and (c) Specification chooses a matching move to simulate the implementation.

The specification tries to minimize the number of moves where it prohibits implementation errors (without destroying the simulation relation), whereas the

implementation tries to maximize it. Intuitively, the positions where the specification cannot allow errors are the critical points for the implementation.

In the game played between $S_1$ and $S_1$, every position is critical. At each position, if an error is allowed, the system can output three $b$'s in a row by using the error transition to return to state 0 while outputting a $b$. The next two moves can be $b$'s irrespective whether errors are allowed or not. This breaks the simulation. Now, consider $I_1$. This system can be allowed to err every two out of three times without violating the specification. This shows that $I_1$ is more robust than $S_1$ for implementing $S_1$. The list of distances is summarized in Table 1.

**Computation of Simulation Distances** The computational complexity of computing the three distances defined here is the same as solving the value problem for the respective games.

For systems without fairness conditions, the $d_{\text{cor}}$, $d_{\text{cov}}$ and $d_{\text{rob}}$ games are simple graph games with *LimAvg* or $Disc_\lambda$ objectives. The decision problem for these games is in NP $\cap$ co-NP [20], but no PTIME algorithm is known. However, for *LimAvg* objectives the existence of an algorithm polynomial in unary encoded weights implies that the computation of the distances can be achieved in polynomial time as we use constant weights. Using the algorithm of [20], in the case without fairness conditions $d_{\text{cor}}$, $d_{\text{cov}}$ and $d_{\text{rob}}$ distances can be computed in time $O((|S||S'|)^3 \cdot (|E||S'| + |E'||S|))$ where $S$ and $S'$ are state spaces of the two transition systems; and $E$ and $E'$ are the sets of transitions of the two systems. A variation of the algorithm in [20] gives a PTIME algorithm for the $Disc_\lambda$ objectives (given a fixed $\lambda$).

For systems with Büchi (weak fairness) conditions, the corresponding games are graph games with *LimAvg* parity games, for which the decision problem is in NP $\cap$ co-NP. However, the use of constant weights and the fact that the implication of two Büchi conditions can be expressed as a parity condition with no more than 3 priorities leads to a polynomial algorithm. Using the algorithm presented in [5], we get a $O((|S||S'|)^3 \cdot (|E||S'| + |E'||S|))$ algorithm.

For systems with Streett (strong fairness) conditions, the corresponding games are graph games with *LimAvg* $\omega$-regular conditions. For an $\omega$-regular *LimAvg* game of $n$ states, we can use the latest appearance records to convert into an equivalent parity game of $2^{O(n \log(n))}$ states and edges; and $n$ priorities. The algorithm of [5] gives a $2^{O(n \log(n))}$ algorithm where $n = |S| \cdot |S'|$.

## 4 Properties of Simulation Distances

We present quantitative analogues of boolean properties of the simulation preorders. Proofs omitted are included in the companion report [3].

**Triangle Inequality** Classical simulation relations satisfy the reflexivity and transitivity property which makes them preorders. In an analogous way, we show that the correctness and coverage distances satisfy the quantitative reflexivity and the triangle inequality properties. This makes them directed metrics [8].

**Theorem 1.** *$d^f_{cor}$ is a directed metric for $f \in \{LimAvg, Disc_\lambda\}$, i.e.:*
- $\forall S \in \mathcal{S} : d^f_{cor}(S, S) = 0$

– $\forall S_1, S_2, S_3 \in \mathcal{S} : d^f_{cor}(S_1, S_3) \leq d^f_{cor}(S_1, S_2) + d^f_{cor}(S_2, S_3)$

**Proof:** We will prove the result for systems with fairness conditions. The case without fairness conditions is analogous. Consider any $\epsilon > 0$. Let $\tau_2$ and $\tau_3$ be $\frac{\epsilon}{2}$-optimal finite strategies for Player 2 in $\mathcal{C}_{S_1,S_2}$ and $\mathcal{C}_{S_2,S_3}$ respectively. Now, we construct a finite-memory strategy $\tau^*$ for Player 2 in $\mathcal{C}_{S_1,S_3}$. If $M_2$ and $M_3$ are the memories of $\tau_2$ and $\tau_3$ respectively, the memory of $\tau^*$ will be $M_2 \times S_2 \times M_3$. The strategy $\tau^*$ works as follows. Let the state of the game be $(s_1, \#, s_3)$ and the memory of $\tau^*$ be $(m_2, s_2, m_3)$.

– Let Player 1 choose to move according to the $S_1$ transition $(s_1, \sigma_1, s'_1)$ to the game state $(s'_1, \sigma_1, s_3)$. Consider the game position $(s'_1, \sigma_1, s_2)$ in $\mathcal{C}_{S_1,S_2}$ and let the $\tau_2$ memory be at state $m_2$. Say $\tau_2$ updates its memory to $m'_2$ and chooses the successor $(s'_1, \#, s'_2)$ with transition symbol $\sigma_1$. Let the corresponding $OutMod(S_2)$ transition be $(s_2, \sigma_1, s'_2)$.
– If the transition $(s_2, \sigma_1, s'_2)$ exists in $S_2$, then let $\sigma'_2 = \sigma_1$. Otherwise, there will exist $(s_2, \sigma_2, s'_2)$ in $S_2$ for some $\sigma_2$. Let $\sigma'_2 = \sigma_2$. Now, consider the game position $(s'_2, \sigma'_2, s_3)$ in $\mathcal{C}_{S_2,S_3}$ and the memory state $m_3$ of $\tau_3$. Say $\tau_3$ updates its memory to $m'_3$ and chooses the successor $(s'_2, \#, s'_3)$ and the transition symbol $\sigma'_2$. Let the corresponding $OutMod(S_3)$ transition be $(s_3, \sigma'_2, s'_3)$.
– The memory of $\tau^*$ is updated to $(m'_2, s'_2, m'_3)$ and $\tau^*$ chooses the successor $(s'_1, \#, s'_3)$ with the transition symbol $\sigma_1$. The corresponding transition $(s_3, \sigma_1, s'_3)$ exists in $OutMod(S_3)$ as there exists a transition with the same source and destination as $(s_3, \sigma'_2, s'_3)$.

$$\left.\begin{array}{l} \left.\begin{array}{l} s_{1,0} \xrightarrow{\sigma_1(v_{1,0})} s_{1,1} \xrightarrow{\sigma_1(v_{1,1})} s_{1,2} \cdots \\ s_{2,0} \xrightarrow{\sigma_1(v_{2,0})} s_{2,1} \xrightarrow{\sigma_1(v_{2,1})} s_{2,2} \cdots \end{array}\right\} \rho_1 \\ \left.\begin{array}{l} s_{2,0} \xrightarrow{\sigma'_2(v_{2,0})} s_{2,1} \xrightarrow{\sigma'_2(v_{2,1})} s_{2,2} \cdots \\ s_{3,0} \xrightarrow{\sigma_1(v_{3,0})} s_{3,1} \xrightarrow{\sigma_1(v_{3,1})} s_{3,2} \cdots \end{array}\right\} \rho_2 \end{array}\right\} \rho$$

If Player 2 cannot match $\sigma_1$ with a zero weight transition while playing according to $\tau^*$, either $\tau_2$ or $\tau_3$ would have also taken a non-zero weight transition. Using this fact, we can easily prove the required property. Fix an arbitrary finite-memory Player 1 strategy $\sigma$. Now, let the play proceed according to the strategy $\tau^*$. From the moves of the game and the state of the memory of $\tau^*$, we can extract four transitions for each round of play as above, i.e. an $S_1$ transition $(s_1, \sigma_1, s'_1)$, an $OutMod(S_2)$ transition $(s_2, \sigma_1, s'_2)$, an $S_2$ transition $(s_2, \sigma'_2, s'_2)$ and an $OutMod(S_3)$ transition $(s_3, \sigma_1, s'_3)$. We depict the situation in the above figure.

The play $\rho$ in $\mathcal{C}_{S_1,S_3}$ corresponds to the transitions in the first and the last rows. This play can be decomposed into plays $\rho_1$ and $\rho_2$ in $\mathcal{C}_{S_1,S_2}$ and $\mathcal{C}_{S_2,S_3}$ by taking only the transitions in the first two and last two rows respectively. Now, by the observation in the previous paragraph, each move in $\rho$ has weight 2 only if one of the corresponding moves in $\rho_1$ or $\rho_2$ have weight 2. Let us denote the $n^{\text{th}}$ move in a play $\eta$ by $\eta^n$. If both $S_1$ and $S_3$ sequence of moves in $\rho$ are fair or if $S_1$ sequence is unfair, we have the following for the *LimAvg* case.

$$\begin{aligned}
\nu(\rho) &= \liminf_{n\to\infty} \frac{1}{n}\sum_{i=0}^{n} v(\rho^i) \leq \liminf_{n\to\infty} \frac{1}{n}\sum_{i=0}^{n} \big(v(\rho_1^i) + v(\rho_2^i)\big) \\
&= \lim_{n\to\infty} \frac{1}{n}\sum_{i=0}^{n} \big(v(\rho_1^i) + v(\rho_2^i)\big) = \lim_{n\to\infty} \frac{1}{n}\sum_{i=0}^{n} v(\rho_1^i) + \lim_{n\to\infty} \frac{1}{n}\sum_{i=0}^{n} v(\rho_2^i) \\
&= \liminf_{n\to\infty} \frac{1}{n}\sum_{i=0}^{n} v(\rho_1^i) + \liminf_{n\to\infty} \frac{1}{n}\sum_{i=0}^{n} v(\rho_2^i) \\
&\leq d_{\mathrm{cor}}(S_1, S_2) + \frac{\epsilon}{2} + d_{\mathrm{cor}}(S_2, S_3) + \frac{\epsilon}{2} = d_{\mathrm{cor}}(S_1, S_2) + d_{\mathrm{cor}}(S_2, S_3) + \epsilon
\end{aligned}$$

All the strategies we are considering are finite-memory, and hence, each sequence of weights is ultimately repeating. Therefore, we can use lim and lim inf interchangeably in the above equations. The case for $Disc_\lambda$ is much simpler and not shown here.

Hence, we have that the value of the play satisfies the required inequality for the case that both $S_1$ and $S_3$ perform fair computations. In the case that $S_1$ sequence is fair and $S_3$ sequence is not fair, the value of the play will be 1. However, by construction the value of either $\rho_1$ or $\rho_2$ will also be 1 and hence the inequality holds.

Therefore, given an $\epsilon$, we have strategy for Player 2 which assures a value less than $d_{\mathrm{cor}}(S_1, S_2) + d_{\mathrm{cor}}(S_2, S_3) + \epsilon$ for both the $LimAvg$ and $Disc_\lambda$ case. Hence, we have the required triangle inequality.

It can be shown by construction of a Player 2 strategy that copies every Player 1 move that $d_{\mathrm{cor}}(S, S) = 0$. Hence, we have the result. ■

**Theorem 2.** *$d^f_{cov}$ is a directed metric when $f \in \{LimAvg, Disc_\lambda\}$, i.e. :*
- $\forall S \in \mathcal{S} : d^f_{cov}(S, S) = 0$
- $\forall S_1, S_2, S_3 \in \mathcal{S} : d^f_{cov}(S_1, S_3) \leq d^f_{cov}(S_1, S_2) + d^f_{cov}(S_2, S_3)$

The robustness distance satisfies the triangle inequality, but not the quantitative reflexivity. The system $S_1$ in Figure 1 is a witness system that violates $d_{\mathrm{rob}}(S_1, S_1) = 0$. In fact, for $LimAvg$ objectives and any rational value $v \in [0, 1]$, it is easy to construct a system $S_v$ such that $d_{\mathrm{rob}}(S_v, S_v) = v$.

**Theorem 3.** *$d^f_{rob}$ conforms to the triangle inequality for $f \in \{LimAvg, Disc_\lambda\}$, i.e. : $\forall S_1, S_2, S_3 \in \mathcal{S} : d^f_{rob}(S_1, S_3) \leq d^f_{rob}(S_1, S_2) + d^f_{rob}(S_2, S_3)$*

**Compositionality** In the qualitative case, compositionality theorems help analyse large systems by decomposing them into smaller components. For example, simulation is preserved when components are composed together. We show that in the quantitative case, the distance between the composed systems is bounded by the sum of the distances between individual systems.

If $A$ and $A'$ are two transition systems, we define *asynchronous and synchronous composition* of the two systems, written as $A \parallel A'$ and $A \times A'$ respectively as follows: (a) The state space is $S \times S'$; (b) $((s, s'), \sigma, (t, t'))$ is a transition

of $A \parallel A'$ iff $(s, \sigma, t)$ is a transition of $A$ and $s' = t'$ or $(s', \sigma, t')$ is a transition of $A'$ and $s = t$, and (c) $((s, s'), \sigma, (t, t'))$ is a transition of $A \times A'$ iff $(s, \sigma, t)$ is a transition of $A$ and $(s', \sigma, t')$ is a transition of $A'$.

The following theorems show that the simulation distances between whole systems is bounded by the sum of distances between the individual components.

**Theorem 4.** *The correctness, coverage and robustness distances satisfy the following property, when* $f \in \{LimAvg, Disc_\lambda\}$*:*

$$d^f(S_1 \times S_2, T_1 \times T_2) \leq d^f(S_1, T_1) + d^f(S_2, T_2)$$

**Theorem 5.** *The correctness, coverage and robustness distances satisfy the following property when* $f = LimAvg$*.*

$$d^f(S_1 \parallel S_2, T_1 \parallel T_2) \leq \alpha.d^f(S_1, T_1) + (1 - \alpha).d^f(S_2, T_2)$$

*where* $\alpha$ *is the fraction of times* $S_1$ *is scheduled in* $S_1 \parallel S_2$ *in the long run, assuming that the fraction has a limit in the long run.*

**Existential and Universal Abstraction.** Classically, properties of systems are studied by studying the properties of over-approximations and under-approximations. In an analogous way, we prove that the distances between systems is bounded from above and below by distances between abstractions of the systems. Given $T = \langle S, \Sigma, E, s_0 \rangle$, an existential (universal) abstraction of it is a system whose states are disjoint subsets of $S$ and an edge exists between two classes iff there exists an edge between one pair (all pairs) of states in the classes.

**Theorem 6.** *Consider a specification* $S$ *and an implementation* $I$*. Let* $S^\exists$ *and* $I^\exists$ *be existential abstractions, and* $S^\forall$ *and* $I^\forall$ *be universal abstractions of* $S$ *and* $I$ *respectively. The correctness, coverage and robustness distances satisfy the three following properties when* $f \in \{LimAvg, Disc_\lambda\}$*:*
*(a)* $d^f_{cor}(I^\forall, S^\exists) \leq d^f_{cor}(I, S) \leq d^f_{cor}(I^\exists, S^\forall)$
*(b)* $d^f_{cov}(I^\exists, S^\forall) \leq d^f_{cov}(I, S) \leq d^f_{cov}(I^\forall, S^\exists)$
*(c)* $d^f_{rob}(I^\forall, S^\exists) \leq d^f_{rob}(I, S) \leq d^f_{rob}(I^\exists, S^\forall)$

## 5 Robustness of Forward Error Correction Systems

Forward Error Correction systems (FECS) are a mechanism of error control for data transmission on noisy channels. A very important characteristic of these error correction systems is the *maximum tolerable bit-error rate*, which is the maximum number of errors the system can tolerate while still being able to successfully decode the message. We show that this property can be measured as the $d_{\text{rob}}$ distance between a system and an ideal system (specification).

We will examine three forward error correction systems: one with no error correction facilities, the Hamming(7,4) code [14], and triple modular redundancy (TMR) that by design can tolerate no errors, one error in seven and three bits respectively. We measure the robustness with respect to an ideal system which can tolerate an unbounded number of errors. For the pseudo-code for the three systems we are examining, the user is referred to the companion report [3]. The only errors we allow are bit flips during transmission.

| $T_1$ | $T_2$ | $d_{\text{rob}}(T_1, T_2)$ |
|---|---|---|
| None | Ideal System | 1 |
| Hamming | Ideal System | 6/7 |
| TMR | Ideal System | 2/3 |

**Table 2.** Robustness of FECS

These systems were modelled and the values of $d_{\text{rob}}$of these systems measured against the ideal system are summarized in Table 2. The robustness values mirror the error tolerance values. In fact, each robustness value is equal to $1 - e$ where $e$ is the corresponding error tolerance value.

## References


1. R. Alur, T. Henzinger, O. Kupferman, and M. Vardi. Alternating refinement relations. In *CONCUR*, pages 163–178, 1998.
2. P. Caspi and A. Benveniste. Toward an approximation theory for computerised control. In *EMSOFT*, pages 294–304, 2002.
3. Pavol Černý, Thomas A. Henzinger, and Arjun Radhakrishna. Simulation distances. Technical report, 2010.
4. K. Chatterjee, L. Doyen, and T. Henzinger. Quantitative languages. In *CSL*, pages 385–400, 2008.
5. K. Chatterjee, T. A. Henzinger, and M. Jurdzinski. Mean-payoff parity games. In *LICS*, pages 178–187, 2005.
6. L. de Alfaro, M. Faella, and M. Stoelinga. Linear and branching system metrics. *IEEE Trans. Software Eng.*, 35(2):258–273, 2009.
7. L. de Alfaro, T. Henzinger, and R. Majumdar. Discounting the future in systems theory. In *ICALP*, pages 1022–1037, 2003.
8. L. de Alfaro, R. Majumdar, V. Raman, and M. Stoelinga. Game refinement relations and metrics. *Logical Methods in Computer Science*, 4(3), 2008.
9. J. Desharnais, V. Gupta, R. Jagadeesan, and P. Panangaden. Metrics for labelled Markov processes. *Theor. Comput. Sci.*, 318(3):323–354, 2004.
10. M. Droste and P. Gastin. Weighted automata and weighted logics. *Theor. Comput. Sci.*, 380(1-2):69–86, 2007.
11. A. Ehrenfeucht and J. Mycielski. Positional strategies for mean payoff games. In *International Journal of Game Theory*, pages 163–178, 1979.
12. N. Fenton. *Software Metrics: A Rigorous and Practical Approach, Revised (Paperback)*. Course Technology, 1998.
13. Y. Gurevich and L. Harrington. Trees, automata, and games. In *STOC*, pages 60–65, 1982.
14. R. W. Hamming. Error detecting and error correcting codes. *Bell System Tech. J.*, 29:147–160, 1950.
15. T. A. Henzinger, O. Kupferman, and S. K. Rajamani. Fair simulation. In *Information and Computation*, pages 273–287, 1997.
16. R. Lincke, J. Lundberg, and W. Löwe. Comparing software metrics tools. In *ISSTA*, pages 131–142, 2008.
17. R. Milner. An algebraic definition of simulation between programs. In *IJCAI*, pages 481–489, 1971.
18. F. 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.
19. F. van Breugel and J. Worrell. Approximating and computing behavioural distances in probabilistic transition systems. *Theo. Comp. Sci.*, 360(1-3):373–385, 2006.
20. U. Zwick and M. Paterson. The complexity of mean payoff games on graphs. *Theor. Comput. Sci.*, 158(1&2):343–359, 1996.