

Relations, Metrics and Applications of Stochastic Games

Thesis Proposal

Vishwanath Raman

Computer Science Department
University of California, Santa Cruz
vishwa@soe.ucsc.edu

Abstract

The last few decades have seen the emergence of formal methods for the modeling and analysis of hardware and software systems. The formal models of such systems are finite state automata, also known as labelled transition systems. A state of a system is a vertex in an automaton, labels represent inputs and labelled edges represent state transitions. Finite state automata may be deterministic, non-deterministic or probabilistic. The formal analysis of systems can take various forms, such as, (a) verifying that a model of a system has a certain behavior, (b) checking whether two or more models can be composed to produce some desired behavior, (c) synthesizing an implementation from a specification, (d) whether one model *simulates* another, in that all the behaviors of the second model are also present in the first or if two models are *bisimilar*, in that they produce identical behaviors.

In this thesis, we have two concerns. Our first concern is the quantitative generalization of simulation and bisimulation, of systems modeled as two-player stochastic games. Stochastic games are generalizations of probabilistic automata. At every state, one or more players have a choice of moves. The state and the player moves determine a probability distribution over the successor states. In transition systems, simulation is a reflexive, transitive relation that relates pairs of states, such that, all formulas in a boolean logic that can be satisfied from one state are also satisfied in the other. If the relation is symmetric, it is an equivalence called a *bisimulation* relation.

For probabilistic systems, these relations are too coarse; states that have transition probabilities that are close to each other will not be part of the relations. This motivates the quantitative generalizations of these relations via *metrics*, that give a *distance* between states, taking values in a non-singleton real interval. The *kernel* of these metrics, which are states that are at a distance of zero, are the relations. These metrics are *logically characterized* by quantitative logics; states that are close in the metric, have close valuations of any formula in the characterizing logic. Such generalizations have been studied for Markov chains and Markov decision processes. We extend those results to two-player stochastic games.

Our second concern is to show the usefulness of modeling systems with non-determinism, as games. We do this through two applications. The first is synthesizing resource managers in the context of scheduling multi-threaded C programs and the second is devising strategies to increase coverage in the testing of multi-threaded C programs. The contributions of this thesis are, (a) a definition of *metrics* for two-player stochastic games, (b) a logical characterization of game metrics in terms of the quantitative μ -calculus, a specification language that captures all ω -regular objectives, (c) algorithms and computational issues for game metrics and the relations they induce, (d) applications highlighting the usefulness of game models in the formal analysis of systems.

Contents

1	Introduction	2
1.1	Thesis statement	4
1.2	Outline of the proposal	5
2	Results so far	5
2.1	Definitions	5
2.2	Metrics for concurrent games	7
2.2.1	MDPs and turn-based games	7
2.2.2	Concurrent games	8
2.2.3	Logical characterization	10
2.2.4	The Kernel	11
2.2.5	Discounted and average reward games	11
2.2.6	Related work	13
2.3	Algorithms for the metrics in turn-based games and MDPs	13
2.3.1	Algorithms for the metrics	13
2.3.2	Algorithms for the metric kernels in turn-based games and MDPs	15
2.3.3	Algorithms for the concurrent game metric	16
2.3.4	Computing the concurrent metric kernels	17
2.3.5	Related work	17
2.4	Applications	17
2.4.1	Resource management in multi-threaded applications	17
2.4.2	The Scheduling Game	20
2.4.3	Stochastic Games	21
2.4.4	Properties	23
2.4.5	Related work	23
3	Towards defense of the thesis	24
3.1	Discussion on results so far and remaining work	24
3.2	Timeline	25
4	Conclusion	26

1 Introduction

The last few decades have seen the emergence of formal methods for the modeling and analysis of hardware and software systems. A formal model of such systems is finite state automata, also known as labelled transition systems. A state of a system is a vertex in an automaton, labels, also called actions, represent inputs and labelled edges represent state transitions. Finite state automata may be deterministic; the transition relation is a function that maps every state action pair to a unique successor, non-deterministic; the transition relations maps every state action pair to one or more successors, or probabilistic; where the transition relation is a function that maps every state action pair to a probability distribution over the successor states.

The formal analysis of systems can take various forms, such as, (a) verifying that a model of a system has a certain behavior, (b) checking whether two or more models can be composed to

produce some desired behavior, (c) synthesizing an implementation from a formal specification, such that all the behaviors in the specification are also present in the implementation, (d) whether one model *simulates* another, in that all the behaviors of the second are also present in the first or if two models are *bisimilar*, in that they produce identical behaviors. Given an initial state, we define the behavior of a system as a set of paths that can be induced from that state; a path being a sequence of states. A property of system behavior is then a property of such paths. The specification of these properties requires a logic, such as LTL or CTL [46, 40, 41], and their verification requires a *model checker* [11, 10, 47], which is a decision procedure that checks the validity of statements in the logic.

To verify that a property φ holds on all paths of a system, model checking is tantamount to an emptiness check on the automata product of the implementation and the negated property $\neg\varphi$; the automata being non-empty indicates the existence of a path along which φ does not hold. The problem of checking whether two or more models can be composed to produce some desired behavior has been studied in [14, 18]. This has led to the development of *Interface Theories* where systems are modeled in terms of their interaction with an environment. This in turn has led to the analysis of composability, which asks the question “is there an environment which can instantiate the components of a system, satisfying some properties of system behavior?”. The problem of synthesis is the dual of the verification problem, where we seek to use the specification to *synthesize* a system that is correct by construction, satisfying all the properties of the specification, obviating the need for verification.

In this thesis, our first concern is with the problem of equivalence and refinement in formal analysis. The refinement of two states is expressed by a relation, called the simulation relation. One state simulates another, if for every transition that can be taken from the second, there exists a transition that can be taken from the first (the simulating state), such that the destination states are in the same simulation relation. The simulation relation is reflexive and transitive. If in addition the relation is symmetric, then it is called a bisimulation relation. If two states are bisimilar, then they are considered equivalent. We note that in all transition systems, these relations classify states qualitatively; given two states, they are either in the relation or they are not. The usefulness of these relations stems from their *logical characterization*. A logic characterizes a bisimulation relation iff all formulas expressible in the logic carry the same truth value in bisimilar states. Similarly, a logic characterizes a simulation relation iff all formulas that hold in one state also hold in its simulating states.

In the case of probabilistic systems, since transitions are probabilistic, a natural failing of these relations is that two states that have dissimilar transition probabilities, but where the probabilities are very close to each other will end up not being in the relation. We would therefore like a *metric*, which maps every pair of states to a real number in an interval, say the unit interval, called their state *distance*, such that two states that are close in their transition probabilities should be close to each other in distance. Further, since the underlying model we consider is probabilistic, we would like our metric to be logically characterized by a quantitative logic; where every formula is a mapping from the states to a value in the unit interval giving the maximum probability of satisfying the formula at each state. For (finite-branching) transition systems, and for the class of properties Φ expressible in the μ -calculus [37], state equivalence is captured by bisimulation [44]; for Markov decision processes, it is captured by probabilistic bisimulation [49]. This means that for quantitative properties, the metric provides a tight bound for how much the value of a property can differ at states of the system, and provides thus a quantitative notion of similarity between

states. Given a set Φ of properties, the metric distance of two states s and t can then be defined as $\sup_{\varphi \in \Phi} |\varphi(s) - \varphi(t)|$. Metrics for Markov decision processes have been studied in [21, 54, 55, 23, 24]. Further, the metrics and relations are connected, in the sense that the relations are the *kernels* of the metrics (the pairs of states having metric distance 0). The metrics and relations are at the heart of many verification techniques, from approximate reasoning (one can substitute states that are close in the metric) to system reductions (one can collapse equivalent states) to compositional reasoning and refinement (providing a notion of substitutivity of equivalents).

We extend the results on metrics for Markov decision processes by introducing metrics and equivalence relations for concurrent games, with respect to the class of properties Φ expressible in the quantitative μ -calculus [16, 42]. We consider two-player games played for an infinite number of rounds over finite state spaces. At each round, the players simultaneously and independently select moves; the moves then determine a probability distribution over successor states. These games, known variously as *stochastic games* [51] or *concurrent games* [13, 1, 16], generalize many common structures in computer science, from transition systems, to Markov chains [34] and Markov decision processes [20]. The games are *turn-based* if, at each state, at most one of the players has a choice of moves, and *deterministic* if the successor state is uniquely determined by the current state, and by the moves chosen by the players.

Our second concern in this thesis is to show the usefulness of game models for the formal analysis of systems. We do this through two applications. The first is synthesizing resource managers in the context of scheduling multi-threaded C programs [19] and the second is devising strategies to increase coverage in the testing of multi-threaded C programs. The problem of testing multi-threaded Java programs by targeting coverage has been studied in [6, 28]. We propose to use some of the ideas in those papers and extend them to do coverage driven testing for multi-threaded C programs. The contributions of this thesis are, (a) a definition of metrics, as the quantitative generalization of the classical simulation and bisimulation relations, for stochastic games. (b) a logical characterization of game metrics in terms of the quantitative μ -calculus, a specification language that captures all ω -regular objectives. (c) algorithms and computational issues for game metrics and the relations they induce. (d) applications highlighting the usefulness of game models in the formal analysis of systems.

1.1 Thesis statement

Our thesis is the following.

Stochastic games provide a powerful generalization for the analysis of systems. Metrics and relations in stochastic games generalize simulation and bisimulation in transition systems and provide a robust mechanism to study approximate behavioral equivalences and refinement.

It has been shown that for systems with uncertainty, where non-deterministic choice is replaced with a probability distribution over the non-deterministic successors, exact behavioral equivalence and refinement is too strict and is not tolerant to errors in measurement, which may cause transition probabilities to vary ever so slightly. Metrics have been proposed [22, 54] as the quantitative generalization of exact equivalence and refinement in probabilistic systems. We propose to generalize this to the case of concurrent stochastic games. We show that many systems with uncertainty are suitable for modeling as games between autonomous agents or players. In the formal analysis of such game models, while many questions related to model checking and synthesis have been

addressed, the important case of simulation and bisimulation relations and their generalizations have not been addressed so far. In this thesis, we develop the theory of approximate behavioral equivalence and refinement to close this gap in the formal analysis of game models while showing the usefulness of such models to analyze systems with uncertainty.

1.2 Outline of the proposal

We propose two complementary lines of work. We first develop the theory of approximate behavioral equivalence and refinement generalized to two player concurrent stochastic games. We then describe applications that benefit by game models and argue that at the very least the kernels of the metrics we define, which are states that have a metric distance of zero, can be used for state space reduction of these models. We postulate that several new applications in emerging domains such as bio-informatics, where researchers are beginning to use probabilistic process models and approximate bisimilarity, lend themselves to game models, and will hence benefit by our results. Towards this end, in Section 2, we report on our work so far on metrics, their logical characterization, their computational issues and algorithms [17, 9]. We then describe an application for efficient resource management in multi-threaded C programs, that can be used to avoid deadlocks while ensuring all threads make progress [19]. In Section 3, we discuss work that remains to be done for this thesis and provide a timeline that roughly spans this academic year. We conclude in Section 4.

2 Results so far

In this section we review our results so far. We introduce metrics for stochastic games. We state that the metrics are logically characterized by the quantitative μ -calculus ($q\mu$); the metric distance between states provide a tight bound for the difference in valuation of all formulas expressed in $q\mu$. We state that the metrics are *reciprocal*; in that the distance between two states remains unchanged under a change in players and canonical; the metrics also provide a bound for the difference in values of discounted and long-run average games. We then describe algorithms and computational issues of the metrics and showcase an application of game models for the efficient management of resources in multi-threaded C programs. The work presented in this section is based on results published in [17, 9, 19]. We first introduce a set of definitions.

2.1 Definitions

Valuations. Let $[\theta_1, \theta_2] \subseteq \mathbb{R}$ be a fixed, non-singleton real interval. Given a set of states S , a *valuation over S* is a function $f : S \mapsto [\theta_1, \theta_2]$ associating with every state $s \in S$ a value $\theta_1 \leq f(s) \leq \theta_2$; we let \mathcal{F} be the set of all valuations. For $c \in [\theta_1, \theta_2]$, we denote by \mathbf{c} the constant valuation such that $\mathbf{c}(s) = c$ at all $s \in S$. We order valuations pointwise: for $f, g \in \mathcal{F}$, we write $f \leq g$ iff $f(s) \leq g(s)$ at all $s \in S$; we remark that \mathcal{F} , under \leq , forms a lattice. Given $a, b \in \mathbb{R}$, we write $a \sqcup b = \max\{a, b\}$, and $a \sqcap b = \min\{a, b\}$; we extend \sqcap, \sqcup to valuations by interpreting them in pointwise fashion.

Game structures. For a finite set A , let $\text{Dist}(A)$ denote the set of probability distributions over A . We say that $p \in \text{Dist}(A)$ is *deterministic* if there is $a \in A$ such that $p(a) = 1$. We assume a fixed, finite set \mathcal{V} of *observation variables*.

A (two-player, concurrent) *game structure* $G = \langle S, [\cdot], Moves, \Gamma_1, \Gamma_2, \delta \rangle$ consists of the following components [1, 13]: A finite set S of states. A variable interpretation $[\cdot] : \mathcal{V} \mapsto S \mapsto [\theta_1, \theta_2]$, which associates with each variable $v \in \mathcal{V}$ a valuation $[v]$. A finite set $Moves$ of moves. Two move assignments $\Gamma_1, \Gamma_2: S \mapsto 2^{Moves} \setminus \emptyset$. For $i \in \{1, 2\}$, the assignment Γ_i associates with each state $s \in S$ the nonempty set $\Gamma_i(s) \subseteq Moves$ of moves available to player i at state s . A probabilistic transition function $\delta: S \times Moves^2 \mapsto \text{Dist}(S)$, that gives the probability $\delta(s, a_1, a_2)(t)$ of a transition from s to t when player 1 plays move a_1 and player 2 plays move a_2 .

At every state $s \in S$, player 1 chooses a move $a_1 \in \Gamma_1(s)$, and simultaneously and independently player 2 chooses a move $a_2 \in \Gamma_2(s)$. The game then proceeds to the successor state $t \in S$ with probability $\delta(s, a_1, a_2)(t)$. We let $\text{Dest}(s, a_1, a_2) = \{t \in S \mid \delta(s, a_1, a_2)(t) > 0\}$. The *propositional distance* $p(s, t)$ between two states $s, t \in S$ is the maximum difference in the valuation of any variable:

$$p(s, t) = \max_{v \in \mathcal{V}} ([v](s) - [v](t)).$$

The kernel of the propositional distance induces an equivalence on states: for states s, t , we let $s \equiv t$ if $p(s, t) = 0$. In the following, unless otherwise noted, the definitions refer to a game structure with components $G = \langle S, [\cdot], Moves, \Gamma_1, \Gamma_2, \delta \rangle$. We indicate the opponent of a player $i \in \{1, 2\}$ by $\sim i = 3 - i$. We consider the following subclasses of game structures.

Turn-based game structures. A game structure G is *turn-based* if we can write $S = S_1 \cup S_2$ with $S_1 \cap S_2 = \emptyset$ where $s \in S_1$ implies $|\Gamma_2(s)| = 1$, and $s \in S_2$ implies $|\Gamma_1(s)| = 1$, and further, if there is a special variable $turn \in \mathcal{V}$, such that $[turn]s = \theta_1$ iff $s \in S_1$, and $[turn]s = \theta_2$ iff $s \in S_2$.

Markov decision processes. For $i \in \{1, 2\}$, we say that a structure is an i -MDP if $\forall s \in S$, $|\Gamma_{\sim i}(s)| = 1$. For MDPs, we omit the (single) move of the player without a choice of moves, and write $\delta(s, a)$ for the transition function.

Moves and strategies. A *mixed move* is a probability distribution over the moves available to a player at a state. We denote by $\mathcal{D}_i(s) \subseteq \text{Dist}(Moves)$ the set of mixed moves available to player $i \in \{1, 2\}$ at $s \in S$, where:

$$\mathcal{D}_i(s) = \{\mathcal{D} \in \text{Dist}(Moves) \mid \mathcal{D}(a) > 0 \text{ implies } a \in \Gamma_i(s)\}.$$

The moves in $Moves$ are called *pure moves*. We extend the transition function to mixed moves by defining, for $s \in S$ and $x_1 \in \mathcal{D}_1(s)$, $x_2 \in \mathcal{D}_2(s)$,

$$\delta(s, x_1, x_2)(t) = \sum_{a_1 \in \Gamma_1(s)} \sum_{a_2 \in \Gamma_2(s)} \delta(s, a_1, a_2)(t) \cdot x_1(a_1) \cdot x_2(a_2).$$

A *path* σ of G is an infinite sequence s_0, s_1, s_2, \dots of states in $s \in S$, such that for all $k \geq 0$, there are mixed moves $x_1^k \in \mathcal{D}_1(s_k)$ and $x_2^k \in \mathcal{D}_2(s_k)$ with $\delta(s_k, x_1^k, x_2^k)(s_{k+1}) > 0$. We write Σ for the set of all paths, and Σ_s the set of all paths starting from state s .

A *strategy* for player $i \in \{1, 2\}$ is a function $\pi_i : S^+ \mapsto \text{Dist}(Moves)$ that associates with every non-empty finite sequence $\sigma \in Q^+$ of states, representing the history of the game, a probability distribution $\pi_i(\sigma)$, which is used to select the next move of player i ; we require that all $\sigma \in S^*$ and states $s \in S$, if $\pi_i(\sigma s)(a) > 0$, then $a \in \Gamma_i(s)$. We write Π_i for the set of strategies for player i . Once the starting state s and the strategies π_1 and π_2 for the two players have been chosen, the game is reduced to an ordinary stochastic process, denoted $G_s^{\pi_1, \pi_2}$, which defines a probability distribution on the set Σ of paths. As usual, we can compute expectations $\mathbb{E}_s^{\pi_1, \pi_2}(\cdot)$ of measurable functions,

and probabilities $\Pr_s^{\pi_1, \pi_2}(\cdot)$ of events (measurable sets of paths) with respect to this process. For $k \geq 0$, we let $X_k : \Sigma \rightarrow S$ be the random variable denoting the k -th state along a path.

One-step expectations and predecessor operators. Given a valuation $f \in \mathcal{F}$, a state $s \in S$, and two mixed moves $x_1 \in \mathcal{D}_1(s)$ and $x_2 \in \mathcal{D}_2(s)$, we define the expectation of f from s under x_1, x_2 by $\mathbb{E}_s^{x_1, x_2}(f) = \sum_{t \in S} \delta(s, x_1, x_2)(t) f(t)$. For a game structure G , for $i \in \{1, 2\}$ we define the *valuation transformer* $\text{Pre}_i : \mathcal{F} \mapsto \mathcal{F}$ by, for all $f \in \mathcal{F}$ and $s \in S$,

$$\text{Pre}_i(f)(s) = \sup_{x_i \in \mathcal{D}_i(s)} \inf_{x_{\sim i} \in \mathcal{D}_{\sim i}(s)} \mathbb{E}_s^{x_1, x_2}(f).$$

Intuitively, $\text{Pre}_i(f)(s)$ is the maximal expectation player i can achieve of f after one step from s : this is the standard “one-day” or “next-stage” operator of the theory of repeated games [29].

2.2 Metrics for concurrent games

We develop a *metric* on states of a game structure that captures an approximate notion of equivalence: states close in the metric yield similar values to the players for any winning objective. Specifically, we develop a bisimulation metric $[\simeq_g] \in \mathcal{M}$ such that for any game structure G and states s, t of G , the following continuity property holds:

$$[\simeq_g](s, t) = \sup_{\varphi \in q\mu} |[\varphi](s) - [\varphi](t)|. \quad (1)$$

In particular, the kernel of the metric, that is, states at distance 0, are equivalent: each player can get exactly the same value from either state for any objective. The metrics are invariant under a change of player; we say they are *reciprocal*. Reciprocity is expected to hold since the underlying games we consider are determined —for any game, the value obtained by player 2 is one minus the value obtained by player 1— and yields canonical metrics on games.

Thus, our metrics will generalize equivalence and refinement relations that have been studied on MDPs and in the deterministic setting. Metrics of this type have already been developed for Markov decision processes (MDPs) [54, 23]. We first consider the case of MDPs and turn-based games.

2.2.1 MDPs and turn-based games

We consider the case of 1-MDPs; the case for 2-MDPs is symmetrical. Throughout this subsection, we fix a 1-MDP $\langle S, [\cdot], \text{Moves}, \Gamma_1, \Gamma_2, \delta \rangle$. Before we present the metric correspondent of probabilistic simulation, we first rephrase classical probabilistic (bi)simulation on MDPs [38, 33, 49, 50] as a fixpoint of a *relation transformer*. As a first step, we lift relations between states to relations between distributions. Given a relation $R \subseteq S \times S$ and two distributions $p, q \in \text{Dist}(S)$, we let $p \sqsubseteq_R q$ if there is a function $\Delta : S \times S \rightarrow [0, 1]$ such that:

- $\Delta(s, s') > 0$ implies $(s, s') \in R$;
- $p(s) = \sum_{s' \in S} \Delta(s, s')$ for any $s \in S$;
- $q(s') = \sum_{s \in S} \Delta(s, s')$ for any $s' \in S$.

To rephrase probabilistic simulation, we define the relation transformer $F : 2^{S \times S} \mapsto 2^{S \times S}$ as follows. For all relations $R \subseteq S \times S$ and $s, t \in S$, we let $(s, t) \in F(R)$ iff

$$s \equiv t \wedge \forall x_1 \in \mathcal{D}_1(s) . \exists y_1 \in \mathcal{D}_1(t) . \delta(s, x_1) \sqsubseteq_R \delta(t, y_1), \quad (2)$$

for all states $s, t \in S$. Probabilistic simulation is the greatest fixpoint of (2); probabilistic bisimulation is the greatest symmetrical fixpoint of (2).

To obtain a metric equivalent of probabilistic simulation, we lift the above fixpoint from relations (subsets of S^2) to metrics (maps $S^2 \mapsto \mathbb{R}$), defining a metric transformer $H_{post}^{1MDP} : \mathcal{M} \mapsto \mathcal{M}$. For all $d \in \mathcal{M}$, let $D(\delta(s, x_1), \delta(t, y_1))(d)$ be the *distribution distance* between $\delta(s, x_1)$ and $\delta(t, y_1)$ with respect to the metric d . For $s, t \in S$, we let

$$H_{post}^{1MDP}(d)(s, t) = p(s, t) \sqcup \sup_{x_1 \in \mathcal{D}_1(s)} \inf_{y_1 \in \mathcal{D}_1(t)} D(\delta(s, x_1), \delta(t, y_1))(d). \quad (3)$$

In this definition, the \forall and \exists of (2) have been replaced by \sup and \inf , respectively. The simulation metric is defined as the least fixpoint of (3) and the bisimulation metric is defined as the least symmetrical fixpoint of (3).

For a distance $d \in \mathcal{M}$ and two distributions $p, q \in \text{Dist}(S)$, the *distribution distance* $D(p, q)(d)$ is a measure of how much “work” we have to do to make p look like q , given that moving a unit of probability mass from $s \in S$ to $t \in S$ has cost $d(s, t)$. This distance is defined via the *trans-shipping problem* [54], as the minimum cost of shipping the distribution p into q , with edge costs d . Since the trans-shipping formulation is a linear program and since strong duality holds for linear programs, we use the dual form of the trans-shipping formulation and define our metric as follows,

$$H_{post}^{1MDP}(d)(s, t) = p(s, t) \sqcup \sup_{x_1 \in \mathcal{D}_1(s)} \inf_{y_1 \in \mathcal{D}_1(t)} \sup_{k \in C(d)} (\mathbb{E}_s^{x_1}(k) - \mathbb{E}_t^{y_1}(k)). \quad (4)$$

The constraint $C(d)$ on the valuation k , states that the value of k across states cannot differ by more than d . We call the metric transformer H_{post}^{1MDP} the *a posteriori* metric transformer: the valuation k in (4) is chosen *after* the moves x_1 and y_1 are chosen. We can define an *a priori* metric transformer, where k is chosen before x_1 and y_1 :

$$H_{prio}^{1MDP}(d)(s, t) = p(s, t) \sqcup \sup_{k \in C(d)} \sup_{x_1 \in \mathcal{D}_1(s)} \inf_{y_1 \in \mathcal{D}_1(t)} (\mathbb{E}_s^{x_1}(k) - \mathbb{E}_t^{y_1}(k)). \quad (5)$$

Intuitively, in the a priori transformer, first a valuation $k \in C(d)$ is chosen. State s chooses a move x_1 , trying to maximize the difference in expectations, and state t chooses a move y_1 , trying to minimize it. The distance between s and t is then equal to the difference in the resulting expectations of k .

Theorem 1 below states that for MDPs, a priori and a posteriori simulation metrics coincide.

Theorem 1 *For all MDPs, $H_{post}^{1MDP} = H_{prio}^{1MDP}$.*

2.2.2 Concurrent games

We now extend the simulation and bisimulation metrics from MDPs to general game structures. A posteriori metrics are defined via the metric transformer $H_{\sqsubseteq_1} : \mathcal{M} \mapsto \mathcal{M}$ as follows, for all $d \in \mathcal{M}$

and $s, t \in S$:

$$\begin{aligned} H_{\sqsubseteq_1}(d)(s, t) &= [s \equiv t] \sqcup \sup_{x_1 \in \mathcal{D}_1(s)} \inf_{y_1 \in \mathcal{D}_1(t)} \sup_{y_2 \in \mathcal{D}_2(t)} \inf_{x_2 \in \mathcal{D}_2(s)} D(\delta(s, x_1, x_2), \delta(t, y_1, y_2), d) \\ &= [s \equiv t] \sqcup \sup_{x_1 \in \mathcal{D}_1(s)} \inf_{y_1 \in \mathcal{D}_1(t)} \sup_{y_2 \in \mathcal{D}_2(t)} \inf_{x_2 \in \mathcal{D}_2(s)} \sup_{k \in C(d)} (\mathbb{E}_s^{x_1, x_2}(k) - \mathbb{E}_t^{y_1, y_2}(k)). \end{aligned} \quad (6)$$

A priori metrics are defined by bringing the \sup_k outside. Precisely, we define a metric transformer $H_{\preceq_1} : \mathcal{M} \mapsto \mathcal{M}$ as follows, for all $d \in \mathcal{M}$ and $s, t \in S$:

$$\begin{aligned} H_{\preceq_1}(d)(s, t) &= [s \equiv t] \sqcup \sup_{k \in C(d)} \sup_{x_1 \in \mathcal{D}_1(s)} \inf_{y_1 \in \mathcal{D}_1(t)} \sup_{y_2 \in \mathcal{D}_2(t)} \inf_{x_2 \in \mathcal{D}_2(s)} (\mathbb{E}_s^{x_1, x_2}(k) - \mathbb{E}_t^{y_1, y_2}(k)) \\ &= [s \equiv t] \sqcup \sup_{k \in C(d)} \left[\sup_{x_1 \in \mathcal{D}_1(s)} \inf_{x_2 \in \mathcal{D}_2(s)} \mathbb{E}_s^{x_1, x_2}(k) - \sup_{y_1 \in \mathcal{D}_1(t)} \inf_{y_2 \in \mathcal{D}_2(t)} \mathbb{E}_t^{y_1, y_2}(k) \right] \\ &= [s \equiv t] \sqcup \sup_{k \in C(d)} (\text{Pre}_1(k)(s) - \text{Pre}_1(k)(t)). \end{aligned} \quad (7)$$

Lemma 1 *The functions H_{\preceq_1} and H_{\sqsubseteq_1} are monotonic in the lattice of metrics (\mathcal{M}, \leq) .*

On the basis of this lemma, we can define the least fixpoints of H_{\preceq_1} and H_{\sqsubseteq_1} , which will yield our game simulation and bisimulation metrics.

Definition 1 *A priori metrics:*

- The *a priori simulation metric* $[\preceq_1]$ is the least fixpoint of H_{\preceq_1} .
- The *a priori bisimulation metric* $[\simeq_1]$ is the least symmetrical fixpoint of H_{\preceq_1} .

A posteriori metrics:

- The *a posteriori game simulation metric* $[\sqsubseteq_1]$ is the least fixpoint of H_{\sqsubseteq_1} .
- The *a posteriori game bisimulation metric* $[\cong_1]$ is the least symmetrical fixpoint of H_{\sqsubseteq_1} .

By exchanging the roles of the players, we define the metric transformers H_{\preceq_2} and H_{\sqsubseteq_2} , and the metrics $[\preceq_2]$, $[\simeq_2]$, $[\sqsubseteq_2]$, $[\cong_2]$.

We note that the a posteriori simulation metric $[\sqsubseteq_1]$ has been introduced in [15, 39]. We also note that the a posteriori bisimulation metric $[\cong_i]$ can be defined as the least fixpoint of $H_{\cong_i} : \mathcal{M} \mapsto \mathcal{M}$, defined for all $d \in \mathcal{M}$ and $i \in \{1, 2\}$ by

$$H_{\cong_1}(d) = H_{\sqsubseteq_1}(d) \sqcup \text{Opp}(H_{\sqsubseteq_1}(d)), \quad (8)$$

where $\text{Opp}(d) = \check{d}$ denotes the opposite of a metric d . Similarly, the a priori bisimulation metric $[\simeq_i]$ can be defined as the least fixpoint of $H_{\simeq_i} : \mathcal{M} \mapsto \mathcal{M}$, defined for all $d \in \mathcal{M}$ and $i \in \{1, 2\}$ by

$$H_{\simeq_1}(d) = H_{\preceq_1}(d) \sqcup \text{Opp}(H_{\preceq_1}(d)). \quad (9)$$

Lemma 2 *The operators H_{\preceq_1} and H_{\sqsubseteq_1} on the lattice (\mathcal{M}, \leq) are upper semi-continuous.*

The above lemma shows that the metrics of Definition 1 can be computed via Picard iteration.

A priori and a posteriori metrics are distinct. First, we state that a priori and a posteriori metrics are distinct in general: the a priori metric never exceeds the a posteriori one, and there are concurrent games where it is strictly smaller. Intuitively, this can be explained as follows. Simulation entails trying to simulate the expectation of a valuation k , as we see from (6), (7). It is easier to simulate a state s from a state t if the valuation is known in advance, as in a priori metrics (7), than if the valuation k is chosen after all the moves have been chosen, as in a posteriori metrics (6).

As a special case, we shall see that equality holds for turn-based game structures, in addition to MDPs as we have seen in the previous subsection.

Theorem 2 *The following assertions hold.*

1. For all game structures G , and for all states s, t of G , we have $[s \preceq_1 t] \leq [s \sqsubseteq_1 t]$.
2. There is a game structure G , and states s, t of G , such that $[s \preceq_1 t] = 0$ and $[s \sqsubseteq_1 t] > 0$.
3. For all turn-based game structures, we have $[\preceq_1] = [\sqsubseteq_1]$.

Reciprocity of a priori metric. The previous theorem establishes that the a priori and a posteriori metrics are in general distinct. We now state that it is the a priori metric, rather than the a posteriori one, that enjoys reciprocity, and that provides a (quantitative) logical characterization of $q\mu$.

Theorem 3 *The following assertions hold.*

1. For all game structures G , we have $[\preceq_1] = [\succeq_2]$, and $[\simeq_1] = [\simeq_2]$.
2. There is a concurrent game structure G , with states s and t , where $[\sqsubseteq_1] \neq [\sqsupseteq_2]$.
3. There is a concurrent game structure G , with states s and t , where $[\cong_1] \neq [\cong_2]$.

As a consequence of this theorem, we write $[\simeq_g]$ in place of $[\simeq_1] = [\simeq_2]$, to emphasize that the player 1 and player 2 versions of game equivalence metrics coincide.

2.2.3 Logical characterization

The logic $q\mu$ provides a logical characterization for the a priori metrics. We first state two lemmas that lead to the desired result. The proof of the lemmas use ideas from [39] and [23]. In the following, the first lemma proves that a priori metrics provide a bound for the difference in value of $q\mu$ -formulas.

Lemma 3 *The following assertions hold for all game structures.*

1. For all $\varphi \in q\mu_1^+$, and for all $s, t \in S$, we have $\llbracket \varphi \rrbracket(s) - \llbracket \varphi \rrbracket(t) \leq [s \preceq_1 t]$.
2. For all $\varphi \in q\mu$, and for all $s, t \in S$, we have $|\llbracket \varphi \rrbracket(s) - \llbracket \varphi \rrbracket(t)| \leq [s \simeq_g t]$.

The second lemma states that the $q\mu$ formulas can attain the distance computed by the simulation metric.

Lemma 4 *The following assertions hold for all game structures G , and for all states s, t of G .*

$$[s \preceq_1 t] \leq \sup_{\varphi \in q\mu_1^+} (\llbracket \varphi \rrbracket(s) - \llbracket \varphi \rrbracket(t))$$

$$[s \simeq_g t] \leq \sup_{\varphi \in q\mu} |\llbracket \varphi \rrbracket(s) - \llbracket \varphi \rrbracket(t)|$$

From these two lemmas, we can conclude that $\llbracket q\mu \rrbracket$ provides a logical characterization for the a priori metrics, as stated by the next theorem.

Theorem 4 *The following assertions hold for all game structures G , and for all states s, t of G :*

$$[s \preceq_1 t] = \sup_{\varphi \in q\mu_1^+} (\llbracket \varphi \rrbracket(s) - \llbracket \varphi \rrbracket(t)) \quad [s \simeq_g t] = \sup_{\varphi \in q\mu} |\llbracket \varphi \rrbracket(s) - \llbracket \varphi \rrbracket(t)|$$

We note that, due to Theorem 2, an analogous result does not hold for the a posteriori metrics. Together with the lack of reciprocity of the a posteriori metrics, this is a strong indication that the a priori metrics, and not the a posteriori ones, are the “natural” metrics on concurrent games.

2.2.4 The Kernel

The kernel of the metric $[\simeq_g]$ defines an equivalence relation \simeq_g on the states of a game structure: $s \simeq_g t$ iff $[s \simeq_g t] = 0$. We call this the *game bisimulation* relation. Notice that by the reciprocity property of \simeq_g , the game bisimulation relation is canonical: $\simeq_1 = \simeq_2 = \simeq_g$. Similarly, we define the *game simulation* preorder $s \preceq_1 t$ as the kernel of the directed metric $[\preceq_1]$, that is, $s \preceq_1 t$ iff $[s \preceq_1 t] = 0$. Alternatively, it is possible to define \preceq_1 and \simeq_g directly. Given a relation $R \subseteq S \times S$, let $B(R) \subseteq \mathcal{F}$ consist of all valuations $k \in \mathcal{F}$ such that, for all $s, t \in S$, if sRt then $k(s) \leq k(t)$. We have the following result.

Theorem 5 *Given a game structure G , the relation \preceq_1 (resp. \simeq_1) can be characterized as the largest (resp. largest symmetrical) relation R such that, for all states s, t with sRt , we have $s \equiv t$ and*

$$\forall k \in B(R). \forall x_1 \in \mathcal{D}_1(s). \exists y_1 \in \mathcal{D}_1(t). \forall y_2 \in \mathcal{D}_2(t). \exists x_2 \in \mathcal{D}_2(s). (\mathbb{E}_t^{y_1, y_2}(k) \geq \mathbb{E}_s^{x_1, x_2}(k)) . \quad (10)$$

We note that the above theorem allows the computation of \simeq_g via a partition-refinement scheme. From the logical characterization theorem, we obtain the following corollary.

Corollary 1 *For any game structure G and states s, t of G , we have $s \simeq_g t$ iff $\llbracket \varphi \rrbracket(s) = \llbracket \varphi \rrbracket(t)$ holds for every $\varphi \in q\mu$ and $s \preceq_1 t$ iff $\llbracket \varphi \rrbracket(s) \leq \llbracket \varphi \rrbracket(t)$ holds for every $\varphi \in q\mu_1^+$.*

2.2.5 Discounted and average reward games

From (4) it follows that the game bisimulation metric provides a tight bound for the difference in values of quantitative μ -calculus formulas. In this section, we state that the game bisimulation metric also provides a bound for the difference in average and discounted value of games. This lends further support for the game bisimulation metric, and its kernel, the game bisimulation relation, being the canonical game metrics and relations.

Discounted payoff games. Let π_1 and π_2 be strategies of player 1 and player 2 respectively. Let $\alpha \in [0, 1)$ be a discount factor. The α -discounted payoff $v_1(s, \pi_1, \pi_2)$ for player 1 at a state s for a variable $r \in \mathcal{V}$ and the strategies π_1 and π_2 is defined as

$$v_1^\alpha(s, \pi_1, \pi_2) = (1 - \alpha) \cdot \sum_{n=0}^{\infty} \alpha^n \cdot \mathbb{E}_s^{\pi_1, \pi_2}([r](X_n)) . \quad (11)$$

The discounted payoff for player 2 is defined by $v_2^\alpha(s, \pi_1, \pi_2) = -v_1^\alpha(s, \pi_1, \pi_2)$. Thus, player 1 wins (and player 2 loses) the “discounted sum” of the valuations of r along the path, where the discount factor weighs future rewards with the discount α . Given a state $s \in S$, we are interested in finding the maximal payoff $v_i^\alpha(s)$ that player i can ensure against all opponent strategies, when the game starts from state $s \in S$. This maximal payoff is given by: $w_i^\alpha(s) = \sup_{\pi_i \in \Pi_i} \inf_{\pi_{\sim i} \in \Pi_{\sim i}} v_i(s, \pi_1, \pi_2)$. These values can be computed as the limit of the sequence of α -discounted, n -step rewards, for $n \rightarrow \infty$. For $i \in \{1, 2\}$, we define a sequence of valuations $w_i^\alpha(0)(s)$, $w_i^\alpha(1)(s)$, $w_i^\alpha(2)(s)$, \dots as follows: for all $s \in S$ and $n \geq 0$:

$$w_1^\alpha(n+1)(s) = (1 - \alpha) \cdot [r](s) + \alpha \cdot \text{Pre}_1(w_i^\alpha(n))(s) . \quad (12)$$

where the initial valuation $w_i^\alpha(0)$ is arbitrary. Shapley proved that $w_i^\alpha = \lim_{n \rightarrow \infty} w_i^\alpha(n)$ [51].

Average payoff games. Let π_1 and π_2 be strategies of player 1 and player 2 respectively. The *average* payoff $v_1(s, \pi_1, \pi_2)$ for player 1 at a state s for a variable $r \in \mathcal{V}$ and the strategies π_1 and π_2 is defined as

$$v_1(s, \pi_1, \pi_2) = \lim_{n \rightarrow \infty} \inf \frac{1}{n} \sum_{k=0}^{n-1} \mathbb{E}_s^{\pi_1, \pi_2}([r](X_k)) . \quad (13)$$

The reward for player 2 is obtained by replacing $[r]$ with $-[r]$ in (13). A game structure G with average payoff is called an average reward game. The *average value* of the game G at s for player $i \in \{1, 2\}$ is defined by $w_i(s) = \sup_{\pi_i \in \Pi_i} \inf_{\pi_{\sim i} \in \Pi_{\sim i}} v_i(s, \pi_1, \pi_2)$.

Mertens and Neyman established the determinacy of average games, and showed that the limit of the discounted value of a game as all the discount factors tend to 1 is the same as the average value of the game: for all $s \in S$ and $i \in \{1, 2\}$, we have $\lim_{\alpha \rightarrow 1} w_i^\alpha(s) = w_i(s)$ [43]. It is easy to show that the average value of a game is a valuation.

Metrics for discounted and average payoffs. We state that the game simulation metric $[\preceq_1]$ provides a bound for discounted and long-run rewards. In the following we consider player 1 rewards (the case for player 2 is identical). Our first result is that the difference in discounted rewards is bound by the metric.

Theorem 6 *For all α -discounted rewards w^α , we have that $w^\alpha(s) - w^\alpha(t) \leq [s \preceq_1 t]$ and $|w^\alpha(s) - w^\alpha(t)| \leq [s \simeq_g t]$.*

Using the fact that the limit of the discounted reward, for the discount factor that approaches 1, is equal to the average reward, we obtain that the metrics provide a bound for the difference of average values as well.

Theorem 7 *For all game structures G and states s and t , we have $w(s) - w(t) \leq [s \preceq_1 t]$ and $|w(s) - w(t)| \leq [s \simeq_g t]$.*

2.2.6 Related work

An equivalence relation for deterministic games that are either turn-based, or where the players are constrained to playing pure moves, has been introduced in [2] and called *alternating bisimulation*. Relations and metrics for the general case of concurrent games have so far proved elusive, with some previous attempts at their definition by a subset of the authors following a subtly flawed approach [15, 39]. The cause of the difficulty goes to the heart of the definition of bisimulation. In the definition of bisimulation for transition systems, for every pair s, t of bisimilar states, we require that if s can go to a state s' , then t should be able to go to t' , such that s' and t' are again bisimilar (we also ask that s, t have an equivalent predicate valuation). This definition has been extended to Markov decision processes by requiring that for every mixed move from s , there is a mixed move from t , such that the moves induce probability distributions over successor states that are equivalent modulo the underlying bisimulation [49, 48]. Unfortunately, the generalization of this appealing definition to games fails. It turns out, that requiring players to be able to replicate probability distributions over successors (modulo the underlying equivalence) leads to an equivalence that is too fine, and that may fail to relate states at which the same quantitative μ -calculus formulas hold. In our work we show that phrasing the definition in terms of distributions over successor states is the wrong approach for games; rather, the definition should be phrased in terms of expectations of certain metric-bounded quantities. Our starting point is a closer look at the definition of metrics for Markov decision processes. We observe that we can manipulate the definition of metrics given in [55], obtaining an alternative form, which we call the *a priori* form, in contrast with the original form of [55], which we call the *a posteriori* form.

2.3 Algorithms for the metrics in turn-based games and MDPs

In this section, we present algorithms for computing the metric and its kernel for turn-based games and MDPs. We first present a polynomial time algorithm to compute the operator $H_{\preceq_i}(d)$ that gives the *exact* one-step distance between two states, for $i \in \{1, 2\}$. We then present a PSPACE algorithm to decide whether the limit distance between two states s and t (i.e., $[s \preceq_1 t]$) is at most a rational value r . Our algorithm matches the best known bound known for the special class of Markov chains [53]. Finally, we present improved algorithms for the important case of the kernel of the metrics. For the bisimulation kernel our algorithm is significantly more efficient, when compared to previous algorithms.

2.3.1 Algorithms for the metrics

For turn-based games and hence MDPs, only one player has a choice of moves at a given state. We consider two player 1 states. A similar analysis applies to player 2 states. We remark that the distance between states in S_i and S_{\sim_i} is always $\theta_2 - \theta_1$ due to the existence of the variable *turn*. For a metric $d \in \mathcal{M}$, and states $s, t \in S_1$, computing $H_{\preceq_1}(d)(s, t)$, given that $p(s, t)$ is trivially computed by its definition, entails evaluating the expression, $\sup_{k \in C(d)} \sup_{x \in \mathcal{D}_1(s)} \inf_{y \in \mathcal{D}_1(t)} (\mathbb{E}_s^x(k) - \mathbb{E}_t^y(k))$. By expanding the expectations, we get the following form,

$$\sup_{k \in C(d)} \sup_{x \in \mathcal{D}_1(s)} \inf_{y \in \mathcal{D}_1(t)} \left(\sum_{u \in S} \sum_{a \in \Gamma_1(s)} \delta(s, a)(u) \cdot x(a) \cdot k(u) - \sum_{v \in S} \sum_{b \in \Gamma_1(t)} \delta(t, b)(v) \cdot y(b) \cdot k(v) \right). \quad (14)$$

We observe that the one-step distance as defined in (14) is a *sup-inf non-linear (quadratic)* optimization problem. Using the following lemma we transform (14) to an *inf linear* optimization problem, which we solve by linear programming.

Lemma 5 *For all turn-based game structures G , for all player i states s and t , given a metric $d \in \mathcal{M}$, the following equality holds,*

$$\sup_{k \in C(d)} \sup_{x \in \mathcal{D}_i(s)} \inf_{y \in \mathcal{D}_i(t)} (\mathbb{E}_s^x(k) - \mathbb{E}_t^y(k)) = \sup_{a \in \Gamma_i(s)} \inf_{y \in \mathcal{D}_i(t)} \sup_{k \in C(d)} (\mathbb{E}_s^a(k) - \mathbb{E}_t^y(k)).$$

Therefore, given $d \in \mathcal{M}$, we can write the one-step distance between states s and t as follows,

$$\text{OneStep}(s, t, d) = \sup_{a \in \Gamma_1(s)} \inf_{y \in \mathcal{D}_1(t)} \sup_{k \in C(d)} (\mathbb{E}_s^a(k) - \mathbb{E}_t^y(k)). \quad (15)$$

Hence we compute for all $a \in \Gamma_1(s)$, the expression $\text{OneStep}(s, t, d, a) = \inf_{y \in \mathcal{D}_1(t)} \sup_{k \in C(d)} (\mathbb{E}_s^a(k) - \mathbb{E}_t^y(k))$, and then choose the maximum, i.e., $\max_{a \in \Gamma_1(s)} \text{OneStep}(s, t, d, a)$. We now present a lemma that helps reduce the above inf sup optimization problem to a linear program. We first present a few notations. We denote by λ the set of variables $\lambda_{u,v}$, for $u, v \in S$. Given $d \in \mathcal{M}$, $a \in \Gamma_1(s)$, and a distribution $y \in \mathcal{D}_1(t)$, we write $\lambda \in \Phi(d, a, y)$ if the following linear constraints are satisfied:

- (1) for all $v \in S$: $\sum_{u \in S} \lambda_{u,v} = \delta(s, a)(v)$; (2) for all $u \in S$: $\sum_{v \in S} \lambda_{u,v} = \sum_{b \in \Gamma_1(t)} y(b) \cdot \delta(t, b)(u)$;
(3) for all $u, v \in S$: $\lambda_{u,v} \geq 0$.

Lemma 6 *For all turn-based games and MDPs, for all $d \in \mathcal{M}$, and for all $s, t \in S$, the following assertion holds:*

$$\sup_{a \in \Gamma_1(s)} \inf_{y \in \mathcal{D}_1(t)} \sup_{k \in C(d)} (\mathbb{E}_s^a(k) - \mathbb{E}_t^y(k)) = \sup_{a \in \Gamma_1(s)} \inf_{y \in \mathcal{D}_1(t)} \inf_{\lambda \in \Phi(d, a, y)} \left(\sum_{u, v \in S} d(u, v) \cdot \lambda_{u,v} \right).$$

Using the above result we obtain the following LP for $\text{OneStep}(s, t, d, a)$ over the variables: (a) $\{\lambda_{u,v}\}_{u,v \in S}$, and (b) y_b for $b \in \Gamma_1(t)$:

$$\text{Minimize } \sum_{u, v \in S} d(u, v) \cdot \lambda_{u,v} \quad \text{subject to} \quad (16)$$

- (1) for all $v \in S$: $\sum_{u \in S} \lambda_{u,v} = \delta(s, a)(v)$; (2) for all $u \in S$: $\sum_{v \in S} \lambda_{u,v} = \sum_{b \in \Gamma_1(t)} y_b \cdot \delta(t, b)(u)$;
(3) for all $u, v \in S$: $\lambda_{u,v} \geq 0$; (4) for all $b \in \Gamma_1(t)$: $y_b \geq 0$; (5) $\sum_{b \in \Gamma_1(t)} y_b = 1$.

Theorem 8 *For all turn-based games and MDPs, given $d \in \mathcal{M}$, for all states $s, t \in S$, we can compute $H_{\leq 1}(d)(s, t)$ in polynomial time by the LP (16).*

Iteration of $\text{OneStep}(s, t, d)$ converges to the exact distance, however, in general, there are no known bounds for the rate of convergence. We now present a decision procedure to check whether the exact distance between two states is at most a rational value r . We first show how to express

the predicate $d = \text{OneStep}(s, t, d)$, for a given $d \in \mathcal{M}$. We observe that since H_{\preceq_1} is non-decreasing, it follows that $\text{OneStep}(s, t, d) \geq d$. It follows that the equality $d = \text{OneStep}(s, t, d)$ holds iff all the linear inequalities of LP (16) are satisfied, and $d(s, t) = \sum_{u, v \in S} d(u, v) \cdot \lambda_{u, v}$ holds. It then follows that $d = \text{OneStep}(s, t, d)$ can be written as a predicate in the theory of real closed fields. Given a rational r , two states s and t , we present an existential theory of reals formula to decide whether $[s \preceq_1 t] \leq r$. Since $[s \preceq_1 t]$ is the least fixed point of H_{\preceq_1} , we define a formula $\Phi(r)$ that is true iff $[s \preceq_1 t] \leq r$, as follows:

$$\exists d \in \mathcal{M}. [(\text{OneStep}(s, t, d) = d) \wedge (d(s, t) \leq r)] .$$

If the formula $\Phi(r)$ is true, then there exists a fixpoint that is bounded by r , which implies that the least fixpoint is bounded by r . Conversely, if the least fixpoint is bounded by r , then the least fixpoint is a witness d for $\Phi(r)$ being true. Since the existential theory of reals is decidable in PSPACE [7], we have the following result.

Theorem 9 (Decision complexity for exact distance). *For all turn-based games and MDPs, given a rational r , and two states s and t , whether $[s \preceq_1 t] \leq r$ can be decided in PSPACE.*

Given a rational $\epsilon > 0$, using binary search and $\mathcal{O}(\log(\frac{\theta_2 - \theta_1}{\epsilon}))$ many calls to check the formula $\Phi(r)$, we can obtain an interval $[l, u]$ with $u - l \leq \epsilon$ such that $[s \preceq_1 t]$ lies in the interval $[l, u]$.

Corollary 2 (Approximation for exact distance). *For all turn-based games and MDPs, given a rational ϵ , and two states s and t , an interval $[l, u]$ with $u - l \leq \epsilon$ such that $[s \preceq_1 t] \in [l, u]$ can be computed in PSPACE.*

2.3.2 Algorithms for the metric kernels in turn-based games and MDPs

The kernel of the simulation metric \preceq_1 can be computed as the limit of the series $\preceq_1^0, \preceq_1^1, \preceq_1^2, \dots$, of relations. For all $s, t \in S$, we have $(s, t) \in \preceq_1^0$ iff $s \equiv t$. For all $n \geq 0$, we have $(s, t) \in \preceq_1^{n+1}$ iff $\text{OneStep}(s, t, 1_{\preceq_1^n}) = 0$. Checking the condition $\text{OneStep}(s, t, 1_{\preceq_1^n}) = 0$, corresponds to solving an LP feasibility problem for every $a \in \Gamma_1(s)$, as it suffices to replace the minimization goal $\gamma = \sum_{u, v \in S} 1_{\preceq_1^n}(u, v) \cdot \lambda_{u, v}$ with the constraint $\gamma = 0$ in the LP (16).

For the bisimulation kernel we again use partition refinement and get an algorithm that is more efficient than the one proposed in [57]. The refinement step is as follows: given a partition, two states s and t belong to the same refined partition iff every pure move from s induces a probability distribution on equivalence classes that can be matched by mixed moves from t , and vice versa. Precisely, we compute a sequence $\mathcal{Q}^0, \mathcal{Q}^1, \mathcal{Q}^2, \dots$, of partitions. Two states s, t belong to the same class of \mathcal{Q}^0 iff they have the same variable valuation (i.e., iff $s \equiv t$). For $n \geq 0$, since by the definition of the bisimulation metric given in (9), $[s \simeq_g t] = 0$ iff $[s \preceq_1 t] = 0$ and $[t \preceq_1 s] = 0$, two states s, t belonging to a particular class in \mathcal{Q}^n remain in the same class in \mathcal{Q}^{n+1} iff both (s, t) and (t, s) satisfy the set of feasibility LP problems $\text{OneStepBis}(s, t, \mathcal{Q}^n)$ below:

$\text{OneStepBis}(s, t, \mathcal{Q})$ consists of one feasibility LP problem for each $a \in \Gamma(s)$. The problem for $a \in \Gamma(s)$ has set of variables $\{x_b \mid b \in \Gamma(t)\}$, and set of constraints:

$$\begin{aligned} (1) \text{ for all } b \in \Gamma(t) : x_b &\geq 0, & (2) \sum_{b \in \Gamma(t)} x_b &= 1, \\ (3) \text{ for all } V \in \mathcal{Q} : \sum_{b \in \Gamma(t)} \sum_{u \in V} x_b \cdot \delta(t, b)(u) &\geq \sum_{u \in V} \delta(s, a)(u) . \end{aligned}$$

Complexity. The number of partition refinement steps required for the computation of both the simulation and the bisimulation kernel is bounded by $O(|S|^2)$ for turn-based games and MDPs, where S is the set of states. At every refinement step, at most $O(|S|^2)$ state pairs are considered, and for each state pair (s, t) at most $|\Gamma(s)|$ LP feasibility problems need to be solved. Let us denote by $\text{LPF}(n, m)$ the complexity of solving the feasibility of m linear inequalities over n variables. We obtain the following result.

Theorem 10 *For all turn-based games and MDPs G , the following assertions hold:*

1. *the simulation kernel can be computed in $\mathcal{O}(n^4 \cdot m \cdot \text{LPF}(n^2 + m, n^2 + 2n + m + 2))$ time;*
2. *the bisimulation kernel can be computed in $\mathcal{O}(n^4 \cdot m \cdot \text{LPF}(m, n + m + 1))$ time;*

where $n = |S|$ is the size of the state space, and $m = \max_{s \in S} |\Gamma(s)|$.

2.3.3 Algorithms for the concurrent game metric

In this section we first state that the computation of the metric distance is at least as hard as the computation of optimal values in concurrent reachability games. The exact complexity of the latter is open, but it is known to be at least as hard as the square-root sum problem, which is in PSPACE but whose inclusion in NP is a long-standing open problem [26, 30].

We will use the following terms in the result. A *proposition* is a boolean observation variable, and we say a state is labeled by a proposition q iff q is true at s . A state t is *absorbing* in a concurrent game, if both players have only one action available at t , and the next state of t is always t (it is a state with a self-loop). For a proposition q , let $\diamond q$ denote the set of paths that visit a state labeled by q at least once. In concurrent reachability games, the objective is $\diamond q$, for a proposition q , and without loss of generality all states labeled by q are absorbing states.

Theorem 11 *Consider a concurrent game structure G , with a single proposition q , such that all states labeled by q are absorbing states. We can construct in linear-time a concurrent game structure G' , with one additional state t' , such that for all $s \in S$, we have $[s \preceq_1 t'] = \sup_{\pi_1 \in \Pi_1} \inf_{\pi_2 \in \Pi_2} \Pr_s^{\pi_1, \pi_2}(\diamond q)$.*

We state that the metric for concurrent games can be computed using a decision procedure for the theory of real closed fields. Due to the reduction from concurrent reachability games, shown in Theorem 11, it is unlikely that we have an algorithm in NP for the metric distance between states. We therefore construct statements in the theory of real closed fields, firstly to decide whether $[s \preceq_1 t] \leq r$, for a rational r , so that we can approximate the metric distance between states s and t , and secondly to decide if $[s \preceq_1 t] = 0$ in order to compute the kernel of the game simulation and bisimulation metrics. We refer the reader to [9] for our detailed construction of statements in the theory of real closed fields. We present the following theorems on the complexity of computing the metrics. We define the size of a game G as: $|G| = |S| + |T|$, where $|T| = \sum_{s, t \in S} \sum_{a, b \in \text{Moves}} |\delta(s, a, b)(t)|$. Using the complexity of deciding a formula in the theory of real closed fields [5], we get the following result.

Theorem 12 (Decision complexity for exact distance). *For all concurrent games G , given a rational r , and two states s and t , whether $[s \preceq_1 t] \leq r$ can be decided in time $\mathcal{O}(|G|^{\mathcal{O}(|G|^5)})$.*

Corollary 3 (Approximation for exact distance). *For all concurrent games G , given a rational ϵ , and two states s and t , an interval $[l, u]$ with $u - l \leq \epsilon$ such that $[s \preceq_1 t] \in [l, u]$ can be computed in time $\mathcal{O}(\log(\frac{\theta_2 - \theta_1}{\epsilon}) \cdot |G|^{\mathcal{O}(|G|^5)})$.*

2.3.4 Computing the concurrent metric kernels

Similar to the case of turn-based games and MDPs, the kernel of the simulation metric \preceq_1 for concurrent games can be computed as the limit of the series $\preceq_1^0, \preceq_1^1, \preceq_1^2, \dots$, of relations. For all $s, t \in S$, we have $(s, t) \in \preceq_1^0$ iff $s \equiv t$. For all $n \geq 0$, we have $(s, t) \in \preceq_1^{n+1}$ iff the $n + 1$ step metric distance is 0. We again employ a decision procedure for the theory of real closed fields. We now present the complexity of computing the relations.

Theorem 13 *For all concurrent games G , states s and t , whether $s \preceq_1 t$ can be decided in $\mathcal{O}(|G|^{\mathcal{O}(|G|^3)})$ time, and whether $s \simeq_g t$ can be decided in $\mathcal{O}(|G|^{\mathcal{O}(|G|^3)})$ time.*

2.3.5 Related work

Our problem differs from the one previously considered for MDPs in [3]: there, the names of moves (called “labels”) must be preserved by simulation and bisimulation, so that a move from a state has at most one candidate simulator move at another state. Our problem for MDPs is closer to the one considered in [57], where labels must be preserved, but where a label can be associated with multiple probability distributions (moves). Our algorithms match the complexity of the best known algorithms for the sub-class of Markov chains [53]. The only other algorithms for computing the bisimulation kernel for Markov decision processes were first presented in [49], but that approach requires computing the convex hulls of the induced probability distributions over all actions from each of two states and then checking that the hulls have matching vertices. Our approach is by far simpler and has better complexity than other existing algorithms for this important class of games.

2.4 Applications

2.4.1 Resource management in multi-threaded applications

In this subsection, we describe an application of game models towards showing the value of game models in solving practical problems. The problem we consider is that of synthesizing resource managers for multi-threaded C programs. Multi-threaded programs coordinate their interactions through synchronization primitives like mutexes and semaphores, which are managed by an OS-provided resource manager. Our resource managers use knowledge of the structure and resource usage of the threads to guarantee deadlock freedom and progress while managing resources in an efficient way. We developed a tool called Cynthesis, where we automatically generate *code-aware* resource managers. The algorithms compute managers as winning strategies in certain infinite games, and produce a compact code description of these strategies.

To motivate the need for such resource managers, consider the snippets of C code shown in Figure 1. Thread 1 and Thread 2 can lead to a deadlock under a standard, most liberal scheduler. On the other hand, if we can distinguish between the requests for mutex a occurring in the **then** branch versus the **else** branch, then by denying mutex b to Thread 2 when Thread 1 is holding mutex a while it is in the **then** branch, we prevent a deadlock in this case.

In order to describe our results we present the following definitions.

Resources. A *resource* is a non-sharable, reusable quantity. For our purposes, a resource x is an integer-valued variable together with a set of actions $\{w_x!, g_x!, r_x!\}$ on x . These actions correspond to communications between the threads that manipulate the resource and the resource manager, and have the following meaning:

```

1  void inf_1(void)
2  {
3    while (1) {
4      if (exp) {
5        mutex_lock(a);           20 void inf_2(void)
6        mutex_lock(b);           21
7        // critical section     22 while (1) {
8        mutex_unlock(b);        23   mutex_lock(b);
9        mutex_unlock(a);        24   mutex_lock(a);
10     } else {                   25   // critical section
11       mutex_lock(a);           26   mutex_unlock(a);
12       mutex_lock(c);           27   mutex_unlock(b);
13       // critical section     28   }
14       mutex_unlock(c);        29   }
15       mutex_unlock(a);
16     }
17   }
18 }

```

Thread 1

Thread 2

Figure 1: A simple application with two threads

- $w_x!$: a thread requests the resource x (“want x ”).
- $g_x?$: the resource manager grants the resource x to a thread (“get x ”).
- $r_x!$: the thread releases the resource x (“release x ”).

Given a set R of resources, the set of actions on R is $Acts[R] = \{w_x!, g_x?, r_x! | x \in R\} \cup \{\epsilon\}$. The output actions over R correspond to communication from the thread to the resource manager and are given by $Acts^O[R] = \{w_x!, r_x! | x \in R\} \cup \{\epsilon\}$. The input actions over R given by $Acts^I[R] = \{g_x? | x \in R\}$, correspond to communication from the resource manager to the thread.

Thread Interfaces. We model the behavior of threads by *thread interfaces*. Thread interfaces model only the resource manipulation aspect of threads, and abstract out all data manipulation.

A thread interface $I = (R, S, E, s^{init}, \lambda)$ consists of a set R of resources, a finite control-flow graph (S, E) with $E \subseteq S \times S$, an initial state $s^{init} \in S$, and an action label $\lambda : E \rightarrow Acts[R] \setminus \{\epsilon\}$ mapping each edge to a resource action, such that

- each $w_x!$ edge leads to a state whose only outgoing edge is labeled with $g_x?$;
- each $g_x?$ edge starts from a state whose incoming edges are all labeled with $w_x!$.

Intuitively, the conditions on a thread interface guarantee that a “want” action is immediately followed by the corresponding “get” action; moreover, a “get” action has no siblings. We say that a state s is *final* if it has no successors. For $s \in S$, let $Isucc(s) = \{t \in S \mid (s, t) \in E \wedge \lambda(s, t) \in Acts^I[R]\}$ be the set of input successors of s , and let $Osucc(s) = \{t \in S \mid (s, t) \in E \wedge \lambda(s, t) \in$

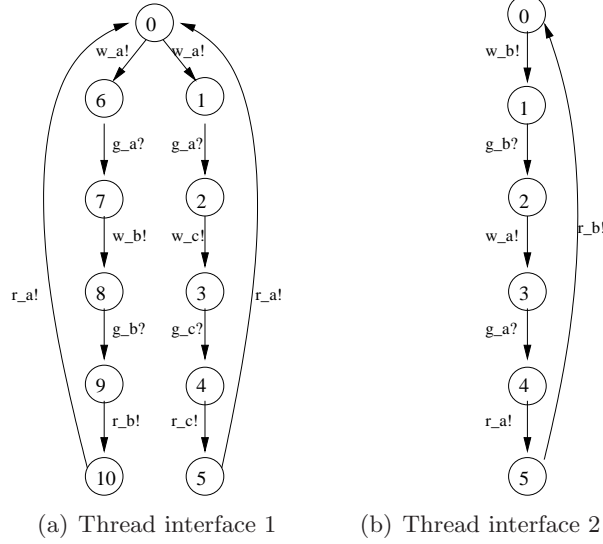


Figure 2: The thread interfaces corresponding to the code in Figure 1.

$Acts^O[R]$ be the set of output successors of s . We carry subscripts over to components, so that an interface I_i will consist of $(R_i, S_i, E_i, s_i^{init}, \lambda_i)$; similarly, we carry subscripts to $Isucc$ and $Osucc$.

Our tool extracts the resource interfaces of Figure 2 from the code in Figure 1. ■

Systems. Given a set R of resources, a *resource valuation* is a function $v : R \mapsto \mathbb{N}$ mapping each resource to a natural number value. For a valuation v and $x \in R$, we denote by $v[x := k]$ the valuation that agrees with the valuation v for all variables except x which is assigned the value $k \in \mathbb{N}$.

A system is a tuple $\mathcal{I} = (R, v^0, (I_1, \dots, I_n))$, consisting of a set R of resources, a mapping $v^0 : R \mapsto \mathbb{N}$ assigning an initial value to each resource, and of $n > 0$ thread interfaces I_1, \dots, I_n . We require that $R_i \subseteq R$, for $1 \leq i \leq n$, and that if $x \in R$ is a mutex, $v^0(x) = 1$.

Given a system, we can define its semantics using a *joint interface*, obtained by constructing the product of the interfaces, annotated with the values of the resources at the states. The joint interface models the execution of a multi-threaded system on a single processor.

Given a system $\mathcal{I} = (R, v^0, (I_1, \dots, I_n))$, its joint interface is a tuple $M_{\mathcal{I}} = (R, S, E, s^{init}, \lambda, \theta)$, where R is as in \mathcal{I} , and:

- $S = (\prod_i S_i) \times (R \mapsto \mathbb{N})$;
- $s^{init} = (s_1^{init}, \dots, s_n^{init}, v^0)$;
- $E \subseteq S \times S$, and $\lambda : E \mapsto Acts[R]$, $\theta : E \mapsto \{0, \dots, n\}$ are defined as follows. Let $s = (s_1, \dots, s_n, v) \in S$; we have $(s, t) \in E$, $\lambda(s, t) = \alpha$, and $\theta(s, t) = i$ iff there is $s'_i \in S_i$ such that $(s_i, s'_i) \in E_i$, $\lambda_i(s_i, s'_i) = \alpha$, and for $t = (s_1, \dots, s_{i-1}, s'_i, s_{i+1}, \dots, s_n, v')$ we have:

[resource grant] if $\alpha = g_x?$, then $v(x) > 0$ and $v' = v[x := v(x) - 1]$;

[resource request] if $\alpha = w_x!$, then $v' = v$; and

[resource release] if $\alpha = r_x!$, then $v' = v[x := v(x) + 1]$; further, if x is a mutex, then $v(x) = 0$.

2.4.2 The Scheduling Game

In this section, unless otherwise noted, we consider a fixed system $\mathcal{I} = (R, \nu^0, (I_1, \dots, I_n))$, which gives rise to a joint interface $M_{\mathcal{I}} = (R, S, E, s^{\text{init}}, \lambda, \theta)$.

A joint interface evolves by the interaction between three entities: the threads, the resource manager, and the scheduler. From a given state, if there are any outgoing edges labeled by input actions, the resource manager can choose to follow one of them: this corresponds to granting a resource to a thread. Once the input edge has been followed (and the resource granted), the resource manager still retains control at the destination state. From a given state, if there are any edges labeled by output actions that leave the state, the resource manager can also elicit to return control to the threads. At this point, which output action occurs next depends on two factors. The underlying operating-system scheduler, using its own policy (such as time-sharing with round robin), selects which of the ready threads execute on the CPU. In addition, each thread has its own internal nondeterminism, which determines which output action the thread generates next. Thus, we identify three types of nondeterminism in the joint interface.

1. *Resource manager nondeterminism*, due to the resource manager choosing an input edge, or choosing to wait for an output action.
2. *Inter-thread nondeterminism*, due to the operating-system scheduler resolving thread interleaving.
3. *Intra-thread nondeterminism*, which determines which of several possible output actions a thread will do.

Resource manager The goal of the resource manager is to ensure that all threads progress, unless they terminate. In order to define the goal, we introduce the following predicates over edges of $M_{\mathcal{I}}$: for $1 \leq i \leq n$, the predicate $progress_i$ is true over an edge $(s, t) \in E$ if $\theta(s, t) = i$, and the predicate $final_i$ is true over an edge $(s, t) \in E$ if the thread i is in a final state in s . Using temporal logic notation, and considering that $final_i$ is equivalent to $\Box final_i$, the goal can be written as a *generalized Büchi* condition over the edges:

$$\phi_{\mathcal{I}}^{goal} = \bigwedge_{i=1}^n \Box \Diamond (progress_i \vee final_i).$$

Our aim is to synthesize a resource manager that satisfies this goal. In order to model accurately the resource manager synthesis problem, we make the following fairness assumptions over the other two types of nondeterminism.

Inter-thread nondeterminism We assume that the underlying operating system scheduler is fair: more precisely, we assume that, if a thread is infinitely often ready to execute, it will make progress infinitely often. We introduce a predicate $ready_i$, for $1 \leq i \leq n$, which is true over an edge $(s, t) \in E$ iff (i) (s, t) is labeled with an output action, and (ii) there is $(s, t') \in E$ with $\theta(s, t') = i$. Intuitively, (i) means that the resource manager decided to let the scheduler schedule some thread, and (ii) means that thread i was among the threads that could have generated the next output. With this notation, the fairness assumption on the scheduler is:

$$\phi_{\mathcal{I}}^{inter} = \bigwedge_{i=1}^n (\Box \Diamond ready_i \Rightarrow \Box \Diamond progress_i).$$

Intra-thread nondeterminism Assuming that intra-thread nondeterminism is resolved in an arbitrary way may easily lead to declaring the manager synthesis problem to be infeasible. In fact, whenever a thread can execute a loop while holding a resource, the arbitrary resolution of intra-thread nondeterminism introduces the possibility that the loop never terminates. In practice, a reasonable assumption is that intra-thread nondeterminism is resolved in a (strongly) fair fashion: if each choice is presented infinitely often, each choice outcome will follow infinitely often. Such fairness entails loop termination.¹ For all threads $1 \leq i \leq n$, all $u, v \in S_i$, and all $(s, t) \in E$, we introduce the predicates $from_i^u(s, t) \stackrel{\text{def}}{=} (loc_i(s) = u)$ and $take_i^{u,v}(s, t) \stackrel{\text{def}}{=} ((loc_i(s) = u) \wedge (loc_i(t) = v))$. The fairness assumption for intra-thread nondeterminism can then be written as

$$\phi_{\mathcal{I}}^{intra} = \bigwedge_{i=1}^n \bigwedge_{u \in S_i} \bigwedge_{v \in Osucc_i(u)} (\Box \Diamond from_i^u \Rightarrow \Box \Diamond take_i^{u,v}).$$

2.4.3 Stochastic Games

We base the synthesis of the resource manager on *stochastic games*. We use probabilities both to approximate the above types of nondeterminism, and to be able to generate manager strategies that are memoryless, but that may require randomization [8]. We denote by $Uniform(A)$ the probability distribution that associates probability $1/|A|$ to every element of a set A . We say that a state $s \in S$ is *winning* if there is $\pi_1 \in \Pi_1$ such that, for all $\pi_2 \in \Pi_2$, we have $\Pr_s^{\pi_1, \pi_2}(\phi) = 1$. As we use randomized strategies, winning with probability 1 is the natural notion of winning. We denote by $Win(G)$ the set of winning states. A *winning strategy* is a strategy that wins from all winning states, that is, a strategy $\pi_1 \in \Pi_1$ such that, for all $s \in Win(G)$ and all $\pi_2 \in \Pi_2$, we have $\Pr_s^{\pi_1, \pi_2}(\phi) = 1$.

Since our aim is to derive strategies that resolve resource manager nondeterminism, we formulate the resource manager synthesis problem as a game played on the joint interface by the resource manager against a team consisting of the threads and the scheduler. Again, unless otherwise noted, we refer to a system $\mathcal{I} = (R, \nu^0, (I_1, \dots, I_n))$ which gives rise to a joint interface $M_{\mathcal{I}} = (R, S, E, s^{\text{init}}, \lambda, \theta)$.

The *two-player game* corresponding to a system \mathcal{I} consists of a tuple $G^2 = (S, Moves, \Gamma_1, \Gamma_2, Dest, \phi^2)$, where $Moves = S \cup \{\perp\}$ and $\phi^2 = (\phi_{\mathcal{I}}^{inter} \wedge \phi_{\mathcal{I}}^{intra}) \Rightarrow \phi_{\mathcal{I}}^{goal}$. The sets of moves for player 1 (representing the resource manager) and player 2 (representing the inter and intra-thread nondeterminism) are as follows, for all $s \in S$:

- If $Osucc(s) \neq \emptyset$, then $\Gamma_1(s) = Isucc(s) \cup \{\perp\}$ and $\Gamma_2(s) = Osucc(s)$.
- If $Osucc(s) = \emptyset$, then $\Gamma_1(s) = Isucc(s)$ and $\Gamma_2(s) = \{\perp\}$.

The destination function is given by the following rules, where $*$ represents a wild-card, and $s \in S$:

- For $t \in Isucc(s)$, we have $Dest(s, t, *) = \delta(t)$;
- for $t \in Osucc(s)$, we have $Dest(s, \perp, t) = \delta(t)$.

The manager synthesis problem can thus be phrased as the problem of finding a winning strategy

¹Recall that our goal is to schedule *correct* software, rather than to perform software verification.

in G^2 . We say that the system is *schedulable* if $s^{\text{init}} \in \text{Win}(G^2)$. One can see that this goal is *upward-closed*, so that memoryless, but randomized, strategies suffice to win the game [8].

The best known algorithms to compute a winning strategy in G^2 take time exponential in the winning condition, and in our case, the size of the winning condition is proportional to the sum of the sizes (numbers of states) of all thread interfaces in \mathcal{I} [52]. Thus, this approach leads to an inefficient algorithm. Instead, we state that we can exploit the special structure of the joint interface and solve the synthesis problem in a more efficient way, consisting of two steps. We consider two simplified versions of G^2 :

1. A game $G^{2.5}$, resulting from resolving all intra-thread nondeterminism in G^2 in a purely randomized fashion.
2. An MDP $G^{1.5}$, resulting from resolving both the intra-thread and the inter-thread nondeterminism in G^2 in a purely randomized fashion.

We can construct in quadratic time in $|G^2|$ a winning strategy for the MDP $G^{1.5}$ which is also a winning strategy of the game $G^{2.5}$. This winning strategy, under many cases of practical importance, is also a winning strategy for the original game G^2 . In all cases, it is possible to check efficiently whether the strategy for game $G^{2.5}$ works also for G^2 — and in our experience, this has been always the case in the examples we have studied so far.

Given the game $G^2 = (S, \text{Moves}, \Gamma_1, \Gamma_2, \text{Dest}, \phi^2)$, the games $G^{2.5} = (S, \text{Moves}', \Gamma_1, \Gamma_2', \text{Dest}', \phi^{2.5})$ and $G^{1.5} = (S, \text{Moves}, \Gamma_1, \Gamma_2'', \text{Dest}'', \phi^{1.5})$ are obtained as follows. We have $\text{Moves}' = \text{Moves} \cup \{1, \dots, n\}$, $\phi^{2.5} = \phi_{\mathcal{I}}^{\text{inter}} \Rightarrow \phi_{\mathcal{I}}^{\text{goal}}$, and $\phi^{1.5} = \phi_{\mathcal{I}}^{\text{goal}}$. The functions Γ_2', Dest' and $\Gamma_2'', \text{Dest}''$ coincide with Γ_2, Dest , except that:

- For all $s \in S$ such that $|\text{Osucc}(s)| > 1$, we let $\Gamma_2'(s) = \{i \mid \exists t \in \Gamma_2(s). \theta(s, t) = i\}$, and for $i \in \Gamma_2'(s)$, we let $\text{Dest}'(s, \perp, i) = \text{Uniform}(\{t \in \Gamma_2(s) \mid \theta(s, t) = i\})$.
- For all $s \in S$, we let $\Gamma_2'' = \{\perp\}$, and we let $\text{Dest}''(s, \perp, \perp) = \text{Uniform}(\text{Osucc}(s))$.

First, we state how to construct the most liberal winning strategy for game $G^{1.5}$; informally, this is the strategy that, among the winning ones, plays with positive probability the largest possible sets of moves.

A memoryless strategy $\pi \in \Pi_1$ gives rise to a graph (S, E_π) , where $E_\pi = \{(s, t) \mid \pi(s)(t) > 0 \text{ or } \pi(s)(\perp) > 0 \text{ and } \lambda(s, t) \in \text{Acts}^O[R]\}$. A *maximal end component* (MEC) of $G^{1.5}$ is a maximal subgraph (C, F) of (S, E) such that: there is a memoryless strategy π such that C is a closed (no outgoing edge) and strongly connected component of (S, E_π) , and such that $F = \{(s, t) \in E_\pi \mid s \in C\}$ [12]. We say that thread k is *finished* in a state s if $\text{loc}_k(s)$ is final in I_k . Notice that if a thread k is finished at some state of a MEC, it is finished at all states of the MEC. We say that a MEC (C, F) is *fair* iff, for every thread $1 \leq k \leq n$, either k is finished in C , or there is $(s, t) \in F$ with $\theta(s, t) = k$. Let W be the union of all sets of states belonging to fair end components. It can be shown that a state is winning in $G^{1.5}$ iff it can reach W with probability 1 [8]; we denote by $\text{Win}(G^{1.5})$ the set of winning states of $G^{1.5}$. By the results of [12, 13], this set can be computed in time quadratic in $|G^{1.5}|$.

The *most liberal winning strategy* π^* for $G^{1.5}$ is the strategy that selects uniformly at random among moves of player 1 that lead only to winning states. Precisely, for $s \in \text{Win}(G^{1.5})$, we let $\pi^*(s) = \text{Uniform}(\{m \in \Gamma_1(s) \mid \forall t \in S. (\text{Dest}''(s, m, \perp)(t) > 0 \Rightarrow t \in \text{Win}(G^{1.5}))\})$. π^* is arbitrarily defined on states $s \in S \setminus \text{Win}(G^{1.5})$.

Theorem 14 *The strategy π^* is winning in $G^{1.5}$, and can be computed in time $O(|G^{1.5}|^2)$.*

2.4.4 Properties

Theorem 15 *The strategy π^* is winning in game $G^{2.5}$, and $\text{Win}(G^{1.5}) = \text{Win}(G^{2.5})$.*

The previous result, which depends in a crucial way on the structural properties of $G^{2.5}$ (it is certainly not valid for an arbitrary two-person game), enables us to compute in quadratic time a winning strategy for game $G^{2.5}$. We now state how to use this result for $G^{2.5}$ also for our original problem G^2 .

Our first result concerns systems where all resources are mutexes (called *mutex-only systems*), and where the threads satisfy the *periodically mutex-free* (PMF) assumption. Informally, this assumption states that, if the intra-thread nondeterminism is resolved in a fair fashion, then the thread is infinitely often not holding any mutex. In practice, threads in mutex-only systems invariably satisfy the PMF assumption. To make this precise, consider a fixed thread interface $I_i = (R_i, S_i, E_i, s_i^{\text{init}}, \lambda_i)$, for $1 \leq i \leq n$. A *path* in I_i is a path in the graph (S_i, E_i) . We say that an infinite path is *fair* iff it satisfies $\bigwedge_{u \in S_i} \bigwedge_{v \in \text{Osucc}_i(u)} \square \diamond \text{from}_i^u \Rightarrow \square \diamond \text{take}_i^{u,v}$. Moreover, for a finite path σ and a resource $x \in R$, let $\text{decr}(x, \sigma) = |\{(s, t) \in \sigma \mid \lambda_i(s, t) = g_x?\}|$, $\text{incr}(x, \sigma) = |\{(s, t) \in \sigma \mid \lambda_i(s, t) = r_x!\}|$, and $\text{balance}(x, \sigma) = \text{incr}(x, \sigma) - \text{decr}(x, \sigma)$. We say that I_i is *mutex-correct* if for all finite traces σ and all mutexes $x \in R_i$, it holds $\text{balance}(x, \sigma) \in \{-1, 0\}$.

We say that a thread is *periodically mutex free* (PMF) if it only uses mutexes, it is mutex-correct, and in all fair paths σ , there exist infinitely many prefixes σ' of σ that satisfy $\text{balance}(x, \sigma') = 0$ for all mutexes x . For mutex-only systems consisting of threads satisfying the PMF assumption (called, for short, *PMF systems*), the strategy π^* is winning also in G^2 . Hence, for PMF systems we can derive resource managers in time quadratic in $|G^2|$.

Theorem 16 *For PMF systems, π^* is winning in game G^2 , and $\text{Win}(G^{1.5}) = \text{Win}(G^2)$.*

Our next result, useful for threads that may use semaphores, enables us to establish whether the strategy π^* is winning also for G^2 .

Theorem 17 *We can check in time $O(|G^2|^2 \cdot n \cdot \sum_{i=1}^n |E_i|)$ whether the strategy π^* is winning in G^2 .*

In our experience, the strategy π^* is almost invariably winning in G^2 ; indeed, the only counterexamples we have been able to construct are based on threads with fairly special structure, where inter-thread communication can be used to synchronize the usage of resources by threads in particular ways. Therefore, we claim that in most cases, we can construct a resource manager strategy in time quadratic in $|G^2|$.

2.4.5 Related work

In closely related work, [36, 35] study the synthesis of code-aware managers for Java. The focus is deadlock avoidance, and as mentioned earlier, the question of progress (absence of starvation) is not addressed. The problem of deadlock prevention has been extensively studied in at least three different fields: databases, operating systems, and flexible manufacturing systems. In the latter field [25, 45, 4, 31, 27, 32], it is assumed that a Petri Net model is constructed by hand. Also,

most of these works deal with processes that are terminating and/or deterministic. In contrast, our approach and tool rely on the automated analysis of software, and we deal in detail with the issues arising from code abstraction and interaction with operating-system schedulers. Further, the use of randomization to generate efficient schedulers has not been studied. Static compiler techniques have been used in high performance thread packages to improve response time through better scheduling [56], however, the problem of resource interaction and deadlock has not been studied. Finally, deadlock detection and prevention methods from transactional databases do not apply in our setting, since our applications do not have transactional semantics and rollback.

3 Towards defense of the thesis

We now review our work in the light of the thesis stated in Section 1.1, and discuss work that remains to be done to defend that thesis. We also provide a rough timeline.

3.1 Discussion on results so far and remaining work

The thesis statement naturally leads to two lines of work. On the theoretical side, we explore an important aspect in the formal analysis of systems, namely the quantitative generalization of simulation and bisimulation relations to two-player stochastic games. The metrics we introduced in Section 2.2 help in the determination of how close is one state is to another with respect to quantitative properties. This can be used to compare multiple models to understand how close they are to each other by the distance between their initial states, leading to a theory of approximate behavioral equivalence and refinement. The metric distances can be used to augment existing techniques for abstraction refinement, which is a proof technique that aims to prove properties by culling states under the argument that a property proved under more liberal system behaviors holds under the restricted actual system behavior. The kernels of the metrics, which are the simulation and bisimulation relations help in lumping equivalent states together and in substituting states based on the simulation relation. The lumping of states has consequences in state space reduction; a recurrent problem in formal analysis.

An interesting application we have seen of metrics in the context of probabilistic systems is in determining how close two models gleaned from *Reaction Networks*, that model intra-cellular interactions are to each other; the closer the initial states are of two models, the more similar is the expected predictive power of the models. As we have described in Section 2.4, systems that are non-deterministic can be effectively modeled as two player games; a system can be thought of as consisting of interactions between various agents. Game models provide a level of unprecedented generality in analyzing system behavior in the presence of adversarial agent intent. We believe most systems modeled as Markov chains or probabilistic processes can be generalized to two player stochastic games, giving the modeler the power of reasoning under adversarial agent behaviors.

On the theoretical side, we would like to extend our metrics with continuous time. We would do this by defining a *rate function* that maps every state to a transition rate, that quantifies how quickly in the future the system will transition out of each state. The moves of players induce transition probabilities with the rate functions inducing an independent transition rate for reaching the successor states. We would like to explore generalizing our metrics first to Continuous Time Markov Chains (CTMCs) and then to two player stochastic games and study such aspects as logics that characterize them and algorithms that can be used to compute them.

A second line of work that is implied by our thesis statement is to show the usefulness of game models by developing applications that solve real-world problems. In Section 2.4, we described one such application, where we synthesized resource managers from winning strategies of suitably constructed games for multi-threaded C programs. To further the application side of our thesis, we propose to work on the problem of *coverage* in multi-threaded C programs. Multi-threaded applications are very hard to test and verify. There are bugs that remain hidden, manifesting under overlapping thread interactions that are hard to test in general through directed testing. We are working on a project called DIRECT, which addresses this testing challenge. We propose to use and extend the infrastructure we developed for Cynthesis, our tool that synthesizes resource managers and described in Section 2.4, to address the problem of coverage. Specifically, we propose to use the model of multiple interacting threads that we built for Cynthesis, to determine interesting local coverage policies, that by one or both of the following mechanisms produce as many overlaps of thread critical sections as possible,

Inject delays at *control points*, which are points during program execution, when control is transferred to the Operating System.

Deny resources at control points, to ensure that threads are selectively put to sleep with the goal of increasing coverage.

We note that a model of interacting threads is the joint interface that we defined in Section 2.4. But the size of such interfaces can be prohibitive even for a small number of threads; a system of five threads with five control points generates 625 states in the joint interface. We therefore propose to study small systems and devise algorithms that provide strategies that are *local* in that the information needed to decide on a course of action at runtime will be based on the current state and perhaps a short history of states seen. This will obviate the need to extract joint interfaces and hence make our approach scale to larger systems, since in such systems we cannot hope to have prior knowledge of the state space in terms of an explicit graph.

Towards our work on coverage, we have completed work on analyzing change in C code. We monitor code fragments that changed between two versions of a program, tracking sections that moved, were deleted or introduced in order to preserve program events across two versions of a program. We call this change impact analysis. We augment change impact analysis with sensitivity analysis, wherein we inject either random, fixed or proportional delays at various control points to produce thread interactions not seen before. The goal of DIRECT is then to automate the selection of delays so that we can increase coverage of a given user program under a given test harness.

While following up on these lines of work, we plan to group related results and publish some of them in archival journals. Finally, we plan to survey the state of the art in the areas of refinement and equivalence in the area of probabilistic systems and present a coherent story of our contributions to the field. We propose to do this while bringing together applications of game models to showcase the usefulness of such models and the need, as a consequence, to fully develop those aspects of the theoretical underpinnings of game models for formal analysis that haven't been done before.

3.2 Timeline

We propose the following tentative timeline that roughly spans this year.

Winter 2009 Expect to finish the algorithmic work on DIRECT. Specifically, study the joint interfaces of a system of interacting threads and define algorithms that can be used to generate

local strategies, which make for efficient runtime exploration of thread interleaving.

Spring 2009 Expect to finish our implementation on DIRECT and work on extending our results on metrics to the case of continuous time.

Summer 2009 We expect to complete a survey of the state of the art on metrics and finish writing our dissertation.

Fall 2010 Expect to defend our thesis.

4 Conclusion

We conclude by summarizing what we expect to be the main contributions of this thesis, and by speculating on its likely impact.

Based on this thesis, we expect to be able to argue that game models are effective for the formal analysis of systems. On the application side, our contributions will lie in part on adapting existing techniques developed for game models. The rest of our contribution will be in motivating, developing and applying new techniques for such problems as scheduling and coverage in the area of multi-threaded C programs. We expect that this work will be useful for engineers to develop tools for the verification, synthesis and testing of systems in the realm of multi-threaded applications, which are becoming increasingly prevalent. We expect that our work on metrics will make it possible for researchers in emerging areas such as bioinformatics to experiment with game models for reaction networks and other models in computational systems biology, using our extension to the theory of approximate behavioral equivalence and refinement.

References

- [1] R. Alur, T.A. Henzinger, and O. Kupferman. Alternating time temporal logic. *J. ACM*, 49:672–713, 2002.
- [2] R. Alur, T.A. Henzinger, O. Kupferman, and M.Y. Vardi. Alternating refinement relations. In *CONCUR 98: Concurrency Theory. 9th Int. Conf.*, volume 1466 of *Lect. Notes in Comp. Sci.*, pages 163–178. Springer-Verlag, 1998.
- [3] C. Baier. Polynomial time algorithms for testing probabilistic bisimulation and simulation. In *CAV*, pages 50–61, 1996.
- [4] Zbigniew A. Banaszak and Bruce H. Krogh. Deadlock avoidance in flexible manufacturing systems with concurrently competing process flows. *IEEE Trans. Rob. Autom.*, 6(6):724–734, December 1990.
- [5] S. Basu. New results on quantifier elimination over real closed fields and applications to constraint databases. *J. ACM*, 46(4):537–555, 1999.
- [6] Marina Biberstein, Eitan Farchi, and Shmuel Ur. Choosing among alternative pasts. In *IPDPS*, page 289, 2003.

- [7] J. F. Canny. Some algebraic and geometric computations in PSPACE. In *STOC*, pages 460–467, 1988.
- [8] K. Chatterjee, L. de Alfaro, , and T.A. Henzinger. Trading memory for randomness. In *In QEST 04: Proceedings of the First International Conference on Quantitative Evaluation of Systems*, pages 206–217. IEEE Computer Society Press, 2004.
- [9] Krishnendu Chatterjee, Luca de Alfaro, Rupak Majumdar, and Vishwanath Raman. Algorithms for game metrics. In *FSTTCS*, 2008.
- [10] E.M. Clarke and E.A. Emerson. Synthesis of synchronization skeletons for branching time temporal logic. In Dexter Kozen, editor, *Logic of Programs: Workshop*, volume 131 of *Lect. Notes in Comp. Sci.*, pages 52–71. Springer-Verlag, 1981.
- [11] E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finite state concurrent systems using temporal logic. In *Proc. 10th ACM Symp. Princ. of Prog. Lang.*, 1983.
- [12] L. de Alfaro. *Formal Verification of Probabilistic Systems*. PhD thesis, Stanford University, 1997. Technical Report STAN-CS-TR-98-1601.
- [13] L. de Alfaro, T. A. Henzinger, and O. Kupferman. Concurrent reachability games. In *FOCS*, pages 564–575, 1998.
- [14] L. de Alfaro and T.A. Henzinger. Interface theories for component-based design. In *EMSOFT 01: 1st Intl. Workshop on Embedded Software*, volume 2211 of *Lect. Notes in Comp. Sci.*, pages 148–165. Springer-Verlag, 2001.
- [15] L. de Alfaro, T.A. Henzinger, and R. Majumdar. Discounting the future in systems theory. In *Proc. 30th Int. Colloq. Aut. Lang. Prog.*, volume 2719 of *Lect. Notes in Comp. Sci.*, pages 1022–1037. Springer-Verlag, 2003.
- [16] L. de Alfaro and R. Majumdar. Quantitative solution of omega-regular games. *Journal of Computer and System Sciences*, 68:374–397, 2004.
- [17] L. de Alfaro, R. Majumdar, V. Raman, and M. Stoelinga. Game relations and metrics. In *LICS*, pages 99–108, 2007.
- [18] L. de Alfaro and M. Stoelinga. Interfaces: A game-theoretic framework to reason about open systems. In *FOCLASA 03: Proceedings of the 2nd International Workshop on Foundations of Coordination Languages and Software Architectures*, volume 97, pages 3–23, 2003.
- [19] Luca de Alfaro, Vishwanath Raman, Marco Faella, and Rupak Majumdar. Code aware resource management. In *EMSOFT*, pages 191–202, 2005.
- [20] C. Derman. *Finite State Markovian Decision Processes*. Academic Press, 1970.
- [21] J. Desharnais, V. Gupta, R. Jagadeesan, and P. Panangaden. Metrics for labelled markov systems. In *CONCUR 99: Concurrency Theory. 10th Int. Conf.*, volume 1664 of *Lect. Notes in Comp. Sci.*, pages 258–273. Springer, 1999.

- [22] J. Desharnais, V. Gupta, R. Jagadeesan, and P. Panangaden. Metrics for labelled markov systems. In *CONCUR 99: Concurrency Theory. 10th Int. Conf.*, volume 1664 of *Lect. Notes in Comp. Sci.*, pages 258–273. Springer, 1999. Full version.
- [23] J. Desharnais, V. Gupta, R. Jagadeesan, and P. Panangaden. Approximating labelled markov processes. *Information and Computation*, 2002.
- [24] J. Desharnais, V. Gupta, R. Jagadeesan, and P. Panangaden. The metric analogue of weak bisimulation for probabilistic processes. In *Proc. 17th IEEE Symp. Logic in Comp. Sci.*, pages 413–422, 2002.
- [25] R. Devillers. Game interpretation of the deadlock avoidance problem. *Commun. ACM*, 20(10):741–745, 1977.
- [26] K. Etessami and M. Yannakakis. On the complexity of Nash equilibria and other fixed points (extended abstract). In *FOCS*, pages 113–123, 2007.
- [27] J. Ezpeleta, J. M. Colom, and J. Martnez. A petri net based deadlock prevention policy for flexible manufacturing systems. *IEEE Transactions on Robotics and Automation.*, N, 2, 11:173–184, 1995.
- [28] Eitan Farchi, Yarden Nir, and Shmuel Ur. Concurrent bug patterns and how to test them. In *IPDPS*, page 286, 2003.
- [29] J. Filar and K. Vrieze. *Competitive Markov Decision Processes*. Springer-Verlag, 1997.
- [30] M. R. Garey, R. L. Graham, and D. S. Johnson. Some NP-complete geometric problems. In *STOC '76: Proceedings of the eighth annual ACM symposium on Theory of computing*, pages 10–22, New York, NY, USA, 1976. ACM.
- [31] F. S. Hsieh and S. C. Chang. Deadlock avoidance controller synthesis for flexible manufacturing systems. *Proc. of 3rd Int. Conf. on Comp. Integrated Manufacturing*, pages 252–261, 1992.
- [32] M.V. Iordache, J. Moody, and P.J. Antsaklis. Synthesis of deadlock prevention supervisors using petri nets. *IEEE Trans. on Robotics and Automation*, 18:59–68, Feb 2002.
- [33] C.-C. Jou and S.A. Smolka. Equivalences, congruences and complete axiomatizations for probabilistic processes. In *CONCUR 90: Concurrency Theory. 1st Int. Conf.*, volume 458 of *Lect. Notes in Comp. Sci.*, pages 367–383. Springer-Verlag, 1990.
- [34] J.G. Kemeny, J.L. Snell, and A.W. Knapp. *Denumerable Markov Chains*. D. Van Nostrand Company, 1966.
- [35] C. Kloukinas, C. Nakhli, and S. Yovine. A methodology and tool support for generating scheduled native code for real-time java applications. In *EMSOFT 03: 3rd Intl. Workshop on Embedded Software*, volume 2855 of *Lect. Notes in Comp. Sci.*, pages 274–289. Springer-Verlag, 2003.
- [36] C. Kloukinas and S. Yovine. Synthesis of safe, qos extendible, application specific schedulers for heterogeneous real-time systems. In *ECRTS*, 2003.

- [37] D. Kozen. Results on the propositional μ -calculus. *Theoretical Computer Science*, 27(3):333–354, 1983.
- [38] K.G. Larsen and A. Skou. Compositional verification of probabilistic processes. In W.R. Cleaveland, editor, *CONCUR 92: Concurrency Theory. 3rd Int. Conf.*, volume 630 of *Lect. Notes in Comp. Sci.* Springer-Verlag, 1992.
- [39] R. Majumdar. *Symbolic algorithms for verification and control*. PhD thesis, University of California, Berkeley, 2003.
- [40] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer-Verlag, New York, 1991.
- [41] Z. Manna and A. Pnueli. *Temporal Verification of Reactive Systems: Safety*. Springer-Verlag, New York, 1995.
- [42] A. McIver and C. Morgan. *Abstraction, Refinement, and Proof for Probabilistic Systems*. Monographs in Computer Science. Springer-Verlag, 2004.
- [43] J.F. Mertens and A. Neyman. Stochastic games. *International Journal of Game Theory*, 10:53–66, 1981.
- [44] R. Milner. Operational and algebraic semantics of concurrent processes. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 1202–1242. Elsevier Science Publishers, 1990.
- [45] Toshimi Minoura. Deadlock avoidance revisited. *J. ACM*, 29(4):1023–1048, 1982.
- [46] A. Pnueli. The temporal logic of programs. In *Proc. 18th IEEE Symp. Found. of Comp. Sci.*, pages 46–57. IEEE Computer Society Press, 1977.
- [47] J.P. Queille and J. Sifakis. Specification and verification of concurrent systems in Cesar. In *Proc. 5th International Symposium on Programming*, volume 137 of *Lect. Notes in Comp. Sci.*, pages 337–351. Springer-Verlag, 1981.
- [48] R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, MIT, 1995. Technical Report MIT/LCS/TR-676.
- [49] R. Segala and N.A. Lynch. Probabilistic simulations for probabilistic processes. In *CONCUR 94: Concurrency Theory. 5th Int. Conf.*, volume 836 of *Lect. Notes in Comp. Sci.*, pages 481–496. Springer-Verlag, 1994.
- [50] R. Segala and N.A. Lynch. Probabilistic simulations for probabilistic processes. *Nordic Journal of Computing*, 2(2):250–273, 1995.
- [51] L.S. Shapley. Stochastic games. *Proc. Nat. Acad. Sci. USA*, 39:1095–1100, 1953.
- [52] W. Thomas. On the synthesis of strategies in infinite games. In *Proc. of 12th Annual Symp. on Theor. Asp. of Comp. Sci.*, volume 900 of *Lect. Notes in Comp. Sci.*, pages 1–13. Springer-Verlag, 1995.

- [53] F. van Breugel, B. Sharma, and J. Worrell. Approximating a behavioural pseudometric without discount for probabilistic systems. *CoRR*, abs/0803.3796, 2008.
- [54] F. van Breugel and J. Worrell. An algorithm for quantitative verification of probabilistic transition systems. In *CONCUR*, pages 336–350, 2001.
- [55] F. van Breugel and J. Worrell. Towards quantitative verification of probabilistic transition systems. In *ICALP*, pages 421–432, 2001.
- [56] J.R. von Behren, J. Condit, F. Zhou, G.C. Necula, and E.A. Brewer. Capriccio: scalable threads for internet services. In *SOSP 03: Symposium on Operating Systems Principles*, pages 268–281. ACM, 2003.
- [57] L. Zhang and H. Hermanns. Deciding simulations on probabilistic automata. In *ATVA*, pages 207–222, 2007.