UNIVERSITY OF CALIFORNIA
SANTA CRUZ

GAME RELATIONS, METRICS AND REFINEMENTS

A dissertation submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

by

Vishwanath Raman

June 2010

The Dissertation of Vishwanath Raman
is approved:

Professor Luca de Alfaro, Chair

Professor Cormac Flanagan

Professor Rupak Majumdar

Tyrus Miller
Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures

List of Tables

Abstract

Game Relations, Metrics and Refinements

by

Vishwanath Raman

Game models for formal analysis have seen significant research effort over the last two decades. For the analysis of systems with non-deterministic behavior, games are a natural model of choice for studying both co-operative and competitive behaviors of the sources of non-determinism. In game models where the sources of non-determinism are treated adversarially, we have that the properties verified, or refinements synthesized, are correct against all possible realizations of non-deterministic behavior. In areas such as security protocols, where participants are rational and are primarily concerned with achieving their own objectives, and only secondarily concerned with violating the objectives of other participants, games are a natural model of participant behaviors. There is active ongoing research in both the theory and applications of games for verification, compositional reasoning and synthesis. In this dissertation, we first develop the theory of approximate behavioral equivalence and refinement in stochastic games and next explore games for synthesis in two different domains. The first in the automatic synthesis of fair non-repudiation protocols, a subclass of fair exchange protocols, used in e-commerce and the second in synthesizing resource managers that ensure progress, and hence lack of starvation, in multi-threaded C programs. Our results are derived from ideas in probabilistic systems, Markov decision processes and stochastic games.

To my dear daddy

Acknowledgments

I consider myself extremely fortunate to have been advised by Luca de Alfaro during the course of my graduate studies. Luca is one of the sharpest minds that I have ever interacted with. He always saw more than I did and set the highest standards for the quality of all publications that originated from the "Casa di Luca". I sincerely hope that his perspicuity and attention to detail will influence my future work and that I can make him proud wherever it is that my career path leads me.

I cannot thank Krishnendu Chatterjee enough for his untiring tutelage over the last two years of my studies and for hosting me as a visiting researcher at IST, Austria. I will never forget the following pearl of wisdom that Krishnendu shared with me: *A Mathematician, a Physicist and a Philosopher were staring at what appeared to be a black sheep on a meadow. The Philosopher declared "All sheep are black". The Physicist quickly followed with "That sheep is black". The Mathematician paused and said "The side of the sheep I see is black"*. This for me captures the essence of what it takes to be a Mathematician or for that matter a Computer Scientist. While I consider myself somewhere in between the Philosopher and the Computer Scientist, my profound thanks to Luca and Krish for having shown me what it takes to cross the chasm.

I would like to give special thanks to Rupak Majumdar for his kindness in agreeing to serve on my committee. I would also like to thank our postdocs Marco Faella, Maria Sorea and Cesar Sanchez, and my lab partners Pritam, Bo, Ian and Leandro, whose presence made my visits to campus, albeit short, extremely pleasant. Special thanks go to Pei-Hsin Ho and Freddy Mang who inspired me to seek a PhD mid-career. The results in this

thesis are based on collaborative papers; I thank my co-authors Krishnendu Chatterjee, Luca de Alfaro, Marco Faella, Rupak Majumdar, Cesar Sanchez and Mariëlle Stoelinga. I thank Cormac Flanagan for being the chair of my advancement committee and for always being there whenever I needed anything that required the attention of the Director of Graduate Studies. Last but not least, my thanks go to Tracie Tucker for the unbounded number of times she has helped me over the last five years.

I would like to thank my wife Nirada for her tremendous support and for being the almost-sole bread winner as I went back to school after over a decade in industry, my children Meghna and Tanvi for doing their best to play quietly as I busied myself with research, and my in-laws for their unwavering interest and enthusiasm for my graduate studies.

I cannot thank my parents enough for all their encouragement over the years; always sharing my aspirations for higher education. I dedicate this thesis to my father who passed away half way through my PhD and would have been immensely proud to see the day of my graduation. All that he wanted was for me to be doing that which made me happy.

The research presented in this thesis was supported in part by the NSF grants CCR-0132780 and CNS-0720884.

Chapter 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 (a) deterministic; the transition relation is a function that maps every state action pair to a unique successor, (b) non-deterministic; the transition relation maps every state action pair to one or more successors, or (c) probabilistic; where the transition relation is a function that maps every state action pair to a probability distribution over 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) checking 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. The behavior of a system, given an initial state, is defined as the 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 set of desired paths. The specification of these properties requires a logic, such as LTL or CTL [Pnu77, MP91, MP95], and their verification requires a *model checker* [CE81, QS81, CES83], which is a decision procedure that checks the validity of statements in the logic using labelled transition systems as structures.

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 [dAH01, dAS03]. This has lead to the development of *Interface Theories* where systems are modeled in terms of their interaction with an environment. This in turn has lead to the analysis of composability, which asks the question "Is there an environment which can instantiate the components of a system, satisfying a set of desired 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.

1.1 Game models

The formal models we study in this thesis are game models. We consider multi-player games played for an infinite number of rounds over finite state spaces. The game models we study are therefore finite state transition systems, where at each state of the system one or more players have a choice of moves that they may select both simultaneously and independently. In non-deterministic game models, given a state and a set of player moves, one for each player, the transition relation gives a set of possible successor states. In probabilistic game models, given a state and a set of player moves, one for each player, the transition relation is a function that returns a distribution over the successor states. These games, known variously as *stochastic games* [Sha53] or *concurrent games* [dAHK98, AHK02, dAM04], generalize many common structures in computer science, from transition systems, to Markov chains [KSK66] and Markov decision processes [Der70]. The most general game models we study are two-player *concurrent stochastic games*, where both players have a choice of moves at each state. If only one player has a choice of moves at each state, we get a *turn-based game*. If only one player has a choice of moves at all states, we get a *Markov decision process* also known as a labelled transition system. Given a game model, a player strategy is a mapping that given a sequence of states representing the history of the game, specifies the move that the player should choose at the last state of that sequence. Once the players fix their strategies, a stochastic game reduces to a probabilistic transition system also known as a *Markov chain*; an ordinary stochastic process. In this thesis we are concerned both in extending the theory of refinement relations and metrics to stochastic games and in the applications of game mod-

els to synthesize implementation refinements that are correct by construction. We defend the following thesis statement:

Games and refinements are central to the formal analysis of concurrent reactive systems and hence the study of refinement relations and metrics in games and the study of correct system realizations as refinements of game models can significantly help in the design of such systems.

1.2 Research Contribution

This thesis is a two part thesis. In the first part, we concern ourselves with extending the theory of equivalence and refinement relations from transition systems to stochastic games. In the second part, we show the effectiveness of game models for the realization of correct systems as component refinements, using two applications; one on synthesizing attack-free fair non-repudiation protocols in e-commerce and another on synthesizing resource managers that ensure progress for multi-threaded C programs. Parts of this work appear in [dAFMR05, dAMRS07, dAMRS08, CdAMR08, CR10].

1.2.1 Metrics and Relations in Stochastic Games

The refinement of 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; all paths originating

from bisimilar states will satisfy exactly the same set of properties. The usefulness of these relations therefore 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. The classical system relations are a basic tool in the study of *boolean* properties of systems, that is, the properties that yield a truth value. As an example, if a state s of a transition system can reach a set of target states R, written $s \models \Diamond R$ in temporal logic, and t can simulate s, then we can conclude $t \models \Diamond R$.

In transition systems, these relations classify states qualitatively; given two states, they are either in the relation or they are not. 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 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 [Koz83b], state equivalence is captured by bisimulation [Mil90]; for Markov decision processes, it is captured by probabilistic bisimulation [SL94]. This means that for quantitative properties, the metric pro-

vides 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 $\sup_{\varphi\in\Phi} |\varphi(s) - \varphi(t)|$.

System metrics play a fundamental role in the study of the quantitative behavior of systems. As an example, if a state s of a Markov chain can reach a set of target states R with probability 0.8, written $s \models \mathbb{P}_{\geq 0.8}\Diamond R$, and if the metric simulation distance from t to s is 0.3, then we can conclude $t \models \mathbb{P}_{\geq 0.5}\Diamond R$. The simulation relation is at the basis of the notions of system refinement and implementation, where qualitative properties are concerned. In analogous fashion, simulation metrics provide a notion of approximate refinement and implementation for quantitative properties.

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). Metrics for Markov decision processes have been studied in [DGJP99, vBW01a, vBW01b, DGJP03, DGJP02]. We extend the results on metrics for Markov decision processes by introducing metrics and equivalence relations for concurrent stochastic games, with respect to the class of properties Φ expressible in the quantitative μ-calculus ($q\mu$) [dAM04, MM04]. Our contributions are as follows:

1. We define two metrics for stochastic games which we call the *a priori* metric and the *a posteriori* metric. The definition of the *a posteriori* metric coincides with the metric

defined in [DGJP99] for Markov decision processes.

2. We show that for concurrent games the a priori and not the a posteriori metric is logically characterized by $q\mu$; the a priori metric bisimulation distance between two states s and t, denoted $[s \simeq_g t]$, has the property that $[s \simeq_g t] = \sup_{\varphi \in q\mu} |\varphi(s) - \varphi(t)|$, where $\varphi(s)$ and $\varphi(t)$ are the values φ assumes at s and t.

3. Moreover, the a priori metric is *reciprocal*, that is the metric distance remains unchanged under a change of players; a desired property of a game metric as games with omega-regular winning conditions are *determined*.

4. We show that the a priori and a posteriori metrics coincide for Markov decision processes and turn-based games and that they may not coincide for concurrent games. There are games where the a posteriori metric is strictly greater than the a priori metric, indicating that the former is too fine a metric, that may distinguish states that have identical valuations of all formulas in $q\mu$.

To each metric is associated a *kernel:* the kernel of a metric is the relation that relates the pairs of states that have distance 0; to each metric corresponds a metric kernel relation. The kernel of the simulation metric is *probabilistic simulation;* the kernel of the bisimulation metric is *probabilistic bisimulation* [SL94].

1.2.2 Discounted, Average and Total Rewards

We show that the a priori metrics are the canonical metrics for stochastic games, by showing that besides being reciprocal and being logically characterized by $q\mu$, they

also provide a bound for the difference in *long-run average* and *discounted average* properties across states of a system. These average rewards play a central role in the theory of stochastic games, and in its applications to optimal control and economics [Ber95, FV97]. Thus, the a priori metrics we introduce are useful both for system verification, and for performance evaluation, supporting our belief that they constitute the canonical metrics for the study of the similarity of states in a game. We point out that it is possible to define a discounted version $[\simeq_g]^\alpha$ of the a priori bisimulation metric; however, we show that this discounted metric does *not* provide a bound for the difference in discounted values. We define a *total metric*, which provides a bound for the difference in discounted values, average reward values and total reward values across states of a system and show that the kernel of all the metrics we develop in this thesis coincide.

1.2.3 Algorithms for Game Metrics

We next investigate algorithms for the computation of the metrics. The metrics can be computed in iterative fashion, following the inductive way in which they are defined. A metric d can be computed as the limit of a monotonically increasing sequence of approximations d_0, d_1, d_2, ..., where $d_0(s,t)$ is the difference in value that variables can take at states s and t. For $k \geq 0$, d_{k+1} is obtained from d_k via $d_{k+1} = H(d_k)$, where the operator H depends on the metric (bisimulation, or simulation), and on the type of system. Our main results are as follows:

1. *Metrics for turn-based games and MDPs.* We show that for turn-based games and MDPs, the one-step metric operator H for both bisimulation and simulation can

be computed in polynomial time, via a reduction to linear programming (LP). The key step in obtaining our polynomial-time algorithm consists in transforming the original sup-inf *non-linear* optimization problem into a quadratic-size inf *linear* optimization problem that can be solved via LP. We then present PSPACE algorithms for both the decision problem of the metric distance between two states and for the problem of computing the approximate metric distance between two states. Our algorithms match the complexity of the best known algorithms for the sub-class of Markov chains [vBSW08].

2. *Metrics for concurrent games.* For concurrent games, our algorithms for the H operator rely on decision procedures for the theory of real closed fields, leading to an EXPTIME procedure with time-complexity $\mathcal{O}(|G|^{\mathcal{O}(|G|^5)})$.

3. *Hardness of metric computation in concurrent games.* We show that computing the exact metric distance of states in concurrent games is at least as hard as computing the value of concurrent reachability games [EY06, dAHK07], which is known to be at least as hard as solving the square-root-sum problem in computational geometry [GGJ76]. These two latter problems are known to lie in PSPACE, and have resisted many attempts to show that they are in NP.

4. *Kernel of the metrics.* We present polynomial time algorithms to compute the simulation and bisimulation kernel of the metrics for turn-based games and MDPs. Our algorithm for the bisimulation kernel of the metric runs in time $\mathcal{O}(n^4)$ (assuming a constant number of moves) as compared to the previous known $\mathcal{O}(n^9 \cdot \log(n))$ algorithm of [ZH07] for MDPs, where n is the size of the state space. For concurrent games the

simulation and the bisimulation kernel can be computed in time $\mathcal{O}(|G|^{\mathcal{O}(|G|^3)})$, where $|G|$ is the size of a game.

For turn-based games and MDPs, the algorithms for probabilistic simulation and bisimulation can be obtained from the LP algorithms that yield the metrics. For probabilistic simulation, the algorithm we obtain coincides with the algorithm previously published in [ZH07]. The algorithm requires the solution of feasibility-LP problems with a number of variables and inequalities that is quadratic in the size of the system. For probabilistic bisimulation, we are able to improve on this result by providing an algorithm that requires the solution of feasibility-LP problems that have linearly many variables and constraints. Precisely, as for ordinary bisimulation, the kernel is computed via iterative refinement of a partition of the state space [Mil90]. Given two states that belong to the same partition, to decide whether the states need to be split in the next partition-refinement step, we present an algorithm that requires the solution of a feasibility-LP problem with a number of variables equal to the number of moves available at the states, and number of constraints linear in the number of equivalence classes. Overall, our algorithm for bisimulation runs in time $\mathcal{O}(n^4)$ (assuming a constant number of moves), considerably improving the $\mathcal{O}(n^9 \cdot \log(n))$ algorithm of [ZH07] for MDPs, and providing for the first time a polynomial algorithm for turn-based games.

1.2.4 Games for Synthesis

In the second part of the thesis, we explore applications of stochastic games. We consider the synthesis of systems from a set of specifications that formally capture desired

behaviors. *Weak co-synthesis*, or *co-operative synthesis*, considers two processes A and B with objectives φ_A and φ_B, and asks whether there exist *refinements*, A' of process A and B' of process B, such that the closed system $[\![A' \parallel B']\!]$ satisfies both φ_A and φ_B. *Classical co-synthesis*, or *strictly competitive synthesis*, considers two processes, a proponent process A with an objective φ and an opponent process B, and asks whether there exists a refinement A' of A, such that the closed system $[\![A' \parallel B]\!]$ satisfies φ against all possible behaviors of B. *Assume-guarantee synthesis*, introduced in [CH07], considers two processes A and B, with objectives φ_A and φ_B and asks whether there exists refinements A' of A and B' of B, such that $[\![A' \parallel B]\!]$ satisfies $\varphi_B \Rightarrow \varphi_A$, $[\![A \parallel B']\!]$ satisfies $\varphi_A \Rightarrow \varphi_B$ and $[\![A' \parallel B']\!]$ satisfies $\varphi_A \wedge \varphi_B$. We show the usefulness of game models for synthesis through two applications. The first application is the automatic synthesis of attack-free fair non-repudiation protocols, a subclass of fair exchange protocols, used in e-commerce. We consider weak, classical and assume-guarantee synthesis and show that classical co-synthesis fails, weak co-synthesis generates solutions that are not attack-free and are hence unacceptable, while assume-guarantee synthesis succeeds. The second application is the automatic synthesis of resource managers for multi-threaded C programs that manage system resources to ensure all threads make progress, thus providing deadlock freedom.

1.2.5 Synthesis of Fair Exchange Protocols

The traditional paper-based contract signing mechanism involves two participants with an intent to sign a piece of contractual text, that is typically in front of them. In this case, either both of them agree and sign the contract or they do not. The mechanism is "fair" to both participants in that it does not afford either participant an unfair

"advantage" over the other. In digital contract signing, ubiquitous in the internet era, an *originator* sends her intent to sign a contractual text to a *recipient*. Over the course of a set of messages, they then proceed to exchange their actual signatures on the contract. In this case, it is in general difficult to ensure fairness as one of the two participants gains an advantage over the other, during the course of the exchange. If the participants do not trust each other, then neither wants to sign the contract first and the one that signs it first may never get a reciprocal signature from the other participant. Moreover, as these contracts are typically signed over asynchronous networks, the communication channels may provide no guarantees on message delivery. The same situation arises in other related areas, such as fair exchange and certified email.

Many protocols have been designed to facilitate the exchange of digital signatures. The earliest exchange protocols were probabilistic. Participants transmit successive bits of information, under the expectation that both participants have similar computation power, to detect dishonest behavior and stop participating in the protocol. These protocols are impractical as the number of messages exchanged may be very large, and both participants having similar computation power may not be realistic. Even and Yacobi [EY80] first showed that no deterministic contract signing protocol can be realized without the involvement of a third party arbitrator who is trusted by all participants. This was formalized as an impossibility result in [PG99], where the authors show that fair exchange is impossible without a *trusted third party (TTP)* for non-repudiation protocols. A simple protocol with a TTP has a TTP collect all signatures and then distribute them to the participants. But this is inefficient as it involves an online TTP to facilitate every exchange, easily

creating a bottleneck at the site of the TTP. This has lead to the development of *optimistic protocols*, where two participants exchange their signatures without involving a TTP, calling upon the TTP to adjudicate only when one of the two participants is dishonest. These protocols are called *fair non-repudiation protocols* with *offline* TTP.

A *fair non-repudiation* protocol is therefore a contract signing protocol, falling under the category of fair exchange protocols, that ensures that at the end of the exchange of signatures over a network, neither participant can deny having participated in the protocol. A non-repudiation protocol, upon successful termination, provides each participant evidence of commitment to a contract that cannot be repudiated by the other participant. A *non-repudiation of origin (NRO)* provides the recipient in an exchange, the ability to present to an adjudicator, evidence of the senders commitment to a contract. A *non-repudiation of receipt (NRR)* provides the sender in an exchange, the ability to present to an adjudicator, evidence of the recipient's commitment to a contract. An exchange protocol should satisfy the following informal requirements [MGK02, GJM99]:

1. *Fairness.* The communication channels quality being fixed, at the end of the exchange protocol run, either all involved parties obtain their expected items or none (even a part) of the information to be exchanged with respect to the missing items is received.

2. *Abuse-freeness.* It is impossible for a single entity at any point in the protocol to be able to prove to an outside party that she has the power to terminate (abort) or successfully complete the protocol.

3. *Timeliness.* The communication channels quality being fixed, the parties always have the ability to reach, in a finite amount of time, a point in the protocol where they can

stop the protocol while preserving fairness.

Some of the existing protocols in this category are the Zhou-Gollmann (ZG) protocol [ZG97], the Asokan-Shoup-Waidner (ASW) protocol [ASW98], the Garay-Jakobsson-MacKenzie (GJM) protocol [GJM99] and the Kremer-Markowitch (KM) protocol [MK01]. Non-Repudiation protocols are difficult to design in general [ZDB00, SM02, MGK02, KMZ02, KR03] and much literature covers the design and verification of these protocols. While some of the literature covers the discovery of vulnerabilities in these protocols based on the content of the exchanged messages, others have tried to find attacks based on the sequences of messages that can be exchanged, based on the rules of the protocols. However, there is no work that focuses on automatically obtaining correct solutions of these subtle and hard to design protocols.

We study the problem of automatically deriving correct fair non-repudiation protocols, that prevent malicious participants from gaining an unfair advantage, by modeling the problem as an automated synthesis problem. To our knowledge this is the first application of controller synthesis to security protocols. Our contributions are as follows:

1. We formally introduce the objectives of the participants and the trusted third party as path formulas in linear-time temporal logic (LTL) and prove that the satisfaction of the objectives imply the fairness and abuse-freeness properties of the protocols. The timeliness property is also satisfied easily.

2. We show that classical (strictly competitive) co-synthesis and weak (co-operative) co-synthesis fail, whereas assume-guarantee synthesis succeeds.

3. We show that all solutions in the set P_{AGS} of assume-guarantee solutions are *attack-*

free, i.e., any solution in P_{AGS} prevents malicious participants from gaining an unfair advantage.

4. We show that the ASW certified mail protocol is not in P_{AGS}, due to known vulnerabilities that could have been automatically discovered. The GJM protocol is also not in P_{AGS} as it compromises our objective for the TTP, while providing fairness and abuse-freeness to the agents. The KM protocol is in P_{AGS} and it follows that it could have been automatically generated by formalizing the problem of protocol design as a synthesis problem.

5. The ASW, GJM and the KM protocol are not symmetric as they do not allow the recipient to abort the protocol. From our analysis of the refinements in P_{AGS} we construct a *new* and *symmetric* fair non-repudiation protocol that provides not just the originator but also the recipient in an exchange, the ability to abort the protocol. Given that the TTP does not change its behavior, we show that the symmetric protocol is attack-free.

6. Our results provide a game-theoretic justification of the need for a trusted third party. This gives an alternative justification of the impossibility results of [EY80, PG99].

It was shown in [CH07] that the solutions of assume-guarantee synthesis can be obtained through the solution of secure equilibria in graph games, and applying the results of [CH07], given our objectives, we show that for fair non-repudiation protocols, the solutions can be obtained in quadratic time.

Security protocols often contain subtle errors that sometimes take years to uncover. A case in point being the Needham-Schroeder public-key protocol that was believed

secure for years before Lowe found a flaw in the protocol [Low95]. Considerable effort is therefore spent in verifying these protocols formally, from the use of special logics such as BAN [BAN90], a logic for the analysis of information exchange protocols, to the use of model checking and theorem proving. The state of the art formal analysis of fair exchange protocols uses model checking to verify a set of properties that these protocols should satisfy, specified in a suitable temporal logic. The work of Shmatikov and Mitchell [SM02] uses the finite state tool Murφ to model the participants in a protocol, together with an intruder model, to check a set of safety properties by state space exploration. They expose a number of vulnerabilities that may lead to replay attacks in both the ASW protocol and the GJM protocol. Zhou et al., show the use of belief logics to verify non-repudiation protocols [ZG98]. The works of Kremer et al., [KfR02, KMZ02, KR03, CKS06] are the first that use game theoretic models and the logic ATL to formally specify fairness, abuse-freeness and timeliness properties and verify them using the tool MOCHA [AHM$^+$98]. They show that protocols can be naturally modeled as games, using ATL with Alternating Transition Systems as structures, to discharge security properties. However these works focus on verification and not synthesis of protocols. Armando et al., [ACC07] use a set-rewriting formalism with LTL, to verify the ASW protocol and report a new attack on the protocol. Louridas in [Lou00] provides several insightful guidelines for the design of non-repudiation protocols.

The work of [AETCR08] uses multi-player games to obtain correct solutions of multi-party rational exchange protocols in the emerging area of rational cryptography. They eliminate the TTP by making the assumption that all agents act rationally to max-

imize their payoffs. These protocols do not provide fairness, but do ensure that rational parties would have no reason to deviate from the protocol. Their technique whittles down the space of all possible rational exchange protocols to those protocols that satisfy feasibility (the possibility of an exchange) and rationality (participating entities all gain at least their individual minimum price or utility for their participation). They use simulated annealing as their search technique. Our technique is very different from this and all previous works, as we use controller synthesis methods to construct protocols that are correct by construction. The finite state models are typically small, so that the application of synthesis techniques as we propose is both appealing and realizable in practice. Moreover, the co-synthesis formulation also enables the automatic discovery of subtle errors in these protocols.

1.2.6 Synthesis of Resource Managers

The second application we present is the automatic synthesis of resource managers in the context of scheduling multi-threaded C programs. Embedded and reactive software is often implemented as a set of communicating and interacting threads. The threads most commonly rely on primitives such as mutexes and counting semaphores to coordinate their interaction, to ensure the atomic execution of critical code regions, and to ensure that shared data structures are correctly accessed. These mutexes and semaphores (which we collectively term *resources*) are managed independently of the application code. In this thesis, we propose the automated construction of *code-aware* managers for resources. Such managers use their knowledge of the thread structure and resource usage to manage resources in an efficient and deadlock-free fashion.

The simplest resource managers, found in the implementation of just about any thread library, use the most liberal of policies: grant a resource whenever it is available. The liberality of this policy creates the possibility of deadlocks: the classical example is when thread 1 requests (and is granted) a mutex A, and thread 2 requests (and is granted) a mutex B. If the next requests are for mutex B from thread 1, and for mutex A from thread 2, deadlock ensues. Writing software that is deadlock-free under such a simple resource management policy is a difficult and error-prone task [SBN+97, EA03]. Monotonic locking [PS85] ensures deadlock freedom, at the price of imposing additional bookkeeping on the programmer. Monotonic locking also cannot be extended to counting semaphores, where there is no notion of a particular thread "holding" a resource. Priority ceiling uses information on the set of locks used by each thread to guarantee deadlock freedom [But04]. Like monotonic locking, however, priority ceiling cannot cope with counting semaphores. Furthermore, when all threads have the same priority and need to get a fair share of CPU time, priority ceiling is a most restrictive policy: it allows at most one thread to hold mutexes at any given time. Other algorithms, such as the banker's algorithm [PS85], rely on a manual analysis of the resources needed for given tasks, and again do not cover code with semaphores.

We present an automatic static technique to synthesize code-aware resource managers for multi-threaded embedded applications that guarantee deadlock freedom while managing resources in a liberal and efficient way. Rather than synthesizing the whole scheduler, we focus on the *resource policy*, i.e., the part of the scheduler responsible for granting resources, depending on the underlying OS scheduler to resolve the remaining

scheduling choices. Our formulation does not require special programmer annotations or code structures, nor any change in programming style. Hence, it is directly applicable to existing bodies of code. We model each thread as an interface automaton and take the product of these automata to yield a game model of a system of threads. We formulate the scheduling problem as a game between the manager and the threads, where the goal for the manager is to avoid deadlocks while ensuring that all threads make progress. The manager is the proponent process and the non-determinism in the thread behavior and in the behavior of the scheduler is the opponent process. We identify two sources of non-determinism: (a) inter-thread non-determinism that is present at each conditional branch statement, where either one of the two branches can be taken, and (b) intra-thread non-determinism that is caused by the OS scheduler choosing amongst the set of threads that are ready and can make progress. A winning strategy in this game provides a refinement of the resource manager that guarantees progress for all threads at run time as long as the inter-thread and intra-thread non-determinism can be resolved in a fair fashion.

For our synthesis formulation we consider the objective of progress under fairness assumptions on inter-thread and intra-thread non-determinism. We provide efficient algorithms that compute winning strategies from the source code in quadratic time, while accounting for scheduler and thread fairness. We then take a closer look at the interaction between the resource manager and the underlying operating system scheduler, and we show how the standard strategy obtained by solving the game can be made more efficient in a real-world resource manager. We show how the strategies can be represented compactly using BDDs, and we discuss how to implement the resource manager so that it is

compact in terms of code size as well as efficient to execute at runtime.

In closely related work, [KY03, KNY03] 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 [Dev77, Min82, BK90, HC92, ECM95, IMA02], 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 [vBCZ+03], 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.

1.3 Organization

In the first part of this thesis, Part I, we focus on the theory of approximate behavioral equivalence and refinement in stochastic games. We introduce background material in Chapter 2, which can be skipped by those familiar with the area of relations in transition systems, logics and competitive Markov decision processes. In Chapter 3, we introduce a

set of definitions that apply to the material covered in the remainder of Part I. We present our quantitative generalizations of the classical simulation and bisimulation relations for stochastic games in Chapter 4. In Chapter 5 we relate metrics with various reward criteria in stochastic games. We conclude Part I with Chapter 6 where we present algorithms to compute the metrics and relations. In the second part of this thesis, Part II, we show two applications of games for synthesis. The first, covered in Chapter 7 is an application of games for the automatic synthesis of fair non-repudiation protocols as participant refinements. The second, covered in Chapter 8 addresses the problem of ensuring progress, and hence lack of starvation, in multi-threaded C programs. We conclude in Chapter 9 where we discuss future directions for work presented in this thesis.

1.3.1 Dependencies

The background material in Chapter 2 may be skipped by readers familiar with transition systems, simulation and bisimulation in transition systems and their logical characterizations. Chapter 7 and Chapter 8, in the second part of this thesis are self-contained and do not require understanding of any of the prior chapters; all that is assumed is familiarity with game models. The material presented in Appendix A has proofs of some of the results stated but not proved in Chapter 7 and may be skipped for a first reading of this thesis. The contents of Chapter 5 and Chapter 6 require understanding of the material presented in Chapter 3 and Chapter 4. The material in Chapter 5 extends game metrics to the case of discounted values, average reward values and total reward values in stochastic games and may be skipped by a reader interested in metrics and relations from a verification standpoint. We have the following self-contained paths in this thesis:

– Chapter 3 → Chapter 4,

– Chapter 3 → Chapter 4 → Chapter 5,

– Chapter 3 → Chapter 4 → Chapter 6,

– Chapter 3 → Chapter 4 → Chapter 5 → Chapter 6,

– Chapter 7, and

– Chapter 8.

Chapter 2

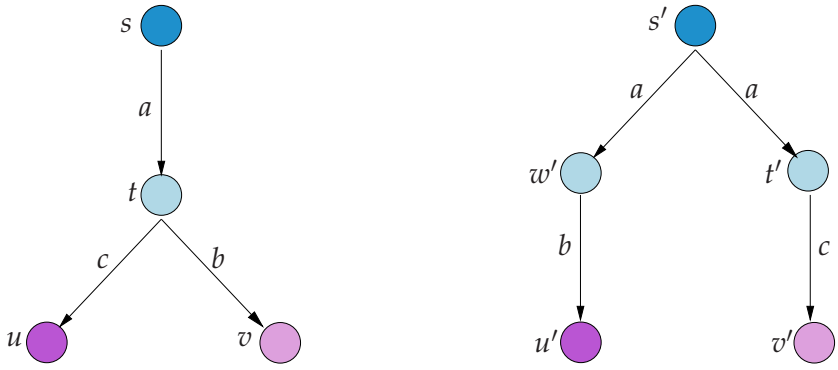
Background

In this chapter we introduce transition systems, logics, simulation and bisimulation relations and their logical characterizations. This chapter can be skipped by readers familiar with the mathematical foundations of formal analysis.

2.1 Labelled Transition Systems

A labelled transition system (LTS) is a formal model of a finite state process that interacts with its environment. A process has an initial state and transitions to a successor state based on the input it receives from its environment. Formally, an LTS is a tuple $(S, s_0, \mathcal{A}, \delta)$, where S is a finite set of states, s_0 is the initial state, $\mathcal{A}$ is a finite set of labels or actions and $\delta : S \times \mathcal{A} \mapsto 2^S \setminus \emptyset$ is a non-deterministic transition function, which given a state and an action returns a non-empty set of possible successor states. If the transition function always returns a singleton set, then the LTS is deterministic.

Example 1 Consider the labelled transition systems in Figure 2.1. In the system shown



(a) LTS 1 (b) LTS 2

Figure 2.1: Labelled Transition Systems.

in Figure 2.1(a), if the process receives input a when it is at state s, then it transitions to state t. Similarly, if the process is in state t and receives input c, then it transitions to state u. This is an example of a deterministic LTS. The LTS in Figure 2.1(b) is non-deterministic, in that it can transition from state s' to either state w' or t' when it sees input a from the environment. ■

2.1.1 Simulation and Bisimulation

One of the most important questions one can ask in concurrency theory is whether two processes are *equivalent*, in that they have the same set of behaviors. There are two notions of equivalence. One is trace containment, where two processes are considered equivalent if they accept the same language. The two processes modeled as labelled transition systems in Figure 2.1 are trace equivalent; they accept the strings ac and ab. A second notion of equivalence is that of bisimulation, introduced by Milner [Mil80], which is a stronger notion of equivalence than trace containment. Two processes are considered

equivalent or *bisimilar* if at all points in their interaction with an environment, for every labelled transition taken by one process the other can take a matching transition and vice versa. While the labelled transition systems in Figure 2.1 are trace equivalent, they are not bisimilar. This is because, after accepting input a from the environment, the process modeled by LTS 1 can accept both b and c, whereas the process modeled by LTS 2 can accept either b or c depending on the manner in which the non-deterministic transition from state s' was resolved. Bisimulation is therefore a state equivalence property and is hence a local property. It is a relation where states that are bisimilar are lumped into equivalence classes such that the sets of immediate transitions that can be generated from all states in an equivalence class coincide.

In formal verification, if we model the specification and the implementation as labelled transition systems, then the bisimulation relation can be used to check if the implementation has exactly the same set of behaviors as the specification and vice versa. Notice that this may be too strong a notion in that we explicitly disallow (a) liberal implementations that may produce a superset of the behaviors in the specification or (b) implementations that *refine* a specification in that all the behaviors of an implementation are also behaviors of the specification but not vice versa. We could therefore drop the requirement for the specification and implementation to be bisimilar, instead seeking to verify that the implementation is a *refinement* of the specification. This leads to a preorder called a *simulation relation*. Besides checking if an implementation refines a specification, the simulation relation can be used to check if one implementation simulates or *is better than* another in that from a user's point of view it allows more behaviors than the other. In the example of

Figure 2.1, the LTS in Figure 2.1(a) simulates the one in Figure 2.1(b) but not the other way around. In other words, LTS 2 *refines* LTS 1 but LTS 1 *does not refine* LTS 2.

Let $(S, s_0, \mathcal{A}, \delta)$ and $(S', s_0', \mathcal{A}', \delta')$ be two labelled transition systems. We can define the simulation relation R using $\forall$ to represent *for all* and $\exists$ to represent *there exists* as follows: A relation $R \subseteq S \times S'$ is a simulation relation if $(s, s') \in R$, also written as $s \preceq s'$, implies that $\forall a \in \mathcal{A}$, if $t \in \delta(s, a)$ then $\exists t' \in S'$ such that $t' \in \delta'(s', a)$ and $t \preceq t'$.

We now present another definition of simulation, where we relax the requirement that labelled transitions should have identical labels. We do this by introducing a set P of predicates that are boolean valued functions, mapping every state in an LTS to either *true* or *false*, based on whether a predicate holds in a state or not. Let $\Gamma : S \mapsto 2^{\mathcal{A}} \setminus \emptyset$ be a function that given a state, returns the set of all enabled actions in that state. For the set of predicates P, we say two states $s, t \in S$ have a *predicate distance* of 0, written $s \equiv t$, iff for all predicates $p \in P$, $p(s) = p(t)$; that is all predicates have identical values at the two states. We can then define the simulation relation as follows: A relation $R \subseteq S \times S'$ is a simulation relation if $s \preceq s'$ implies that,

1. $s \equiv s'$ and

2. $\forall a \in \Gamma(s) \,.\, \exists b \in \Gamma(s') \,.\, \delta(s, a) \preceq \delta(s', b)$.

In this case, what we really want is that the set of states visited in the two systems should have identical predicate values. Since predicates are at the heart of formal specifications, this is a more appealing definition. Going back to our example in Figure 2.1, taking the predicate distance to be 0 for states that have the same color, we see that $s' \preceq s$ but $s \not\preceq s'$; the LTS in Figure 2.1(a) simulates the one in Figure 2.1(b) but not the other way around.

The states s and s' are not bisimilar. For all deterministic labelled transition systems, $s \preceq s'$ and $s' \preceq s$ implies s and s' are bisimilar but this does not hold for non-deterministic systems. We write $s \simeq t$ if the states s and t are bisimilar.

2.1.2 Game Interpretation of Simulation

The simulation relation can be interpreted as game between a protagonist and an antagonist on the graphs that represent labelled transition systems. Let L_I and L_S stand for the labelled transition systems corresponding to an implementation and a specification respectively. Let s and t be the initial states of L_I and L_S. For the implementation to refine the specification, we require that $s \preceq t$. To check if this is the case, we define a game between a protagonist and an antagonist as follows: From s, the antagonist picks an action leading to a state s'. From t, the protagonist picks an action leading to a state t' such that $s' \equiv t'$; the destination states agree in the values of all predicates. The game then proceeds to the next round played from states s' and t'. If the game proceeds forever, then the protagonist wins as she succeeds in matching every move of the antagonist. If the protagonist fails to match a move of the antagonist at any point in the game, then the antagonist wins; the implementation does not refine the specification.

While simulation in both directions implies bisimulation for deterministic systems, there are non-deterministic systems where this is not the case. To show bisimulation in such systems, we need a variant of Ehrenfeucht-Fraïssé games where the antagonist picks an action from either state at each round of the game, while the protagonist should select a move from the state not picked by the antagonist. Similar to the case of simulation, if the game proceeds forever, given that the antagonist chooses states from either L_I or L_S

at each round of the game, then $s \simeq t$ and the two systems are bisimilar.

2.2 Alternating Transition Systems

Reductionism was introduced by Descartes and dates back to the times of Democritus. It is at the heart of understanding complex processes in terms of their simpler constituents and has influenced research in Science and Mathematics for over two millennia. A reductionist approach to the design of systems has lead to the design of complex systems by using simpler well understood components or agents. In this world of *component based design*, a labelled transition system is not as effective a formal model as we would like, because it only deals with *closed systems*; the system is modeled together with its environment. Labelled transition systems thus assume that system behaviors are produced by a set of *co-operating* agents. When we assume that agents co-operate, we place artificial constraints on some of them, including the environment, to ensure the system has expected behaviors. This suggests that we need a richer formal model to reason about properties of sets of agents without constraining the other agents or the environment. At the very least, we want to model *open* systems consisting of a component and its environment as separate agents so that we can check properties of the component against all possible behaviors of the environment. Generalizing this, we want to consider systems as a collection of agents (including the environment) and check properties of arbitrary subsets of agents against all possible and hence unconstrained behaviors of the others. Alternating transition systems provide precisely this ability in formal analysis [AHK97].

Formally, an alternating transition system (ATS), restricted to two agents is a tu-

ple, $(S, s_0, Moves, \Gamma_1, \Gamma_2, \delta)$, where S is a finite set of states, s_0 is the initial state, *Moves* is a finite set of moves available to the agents, $\Gamma_i : S \mapsto Moves \setminus \emptyset$ for $i \in \{1, 2\}$ is a move assignment that given a state returns a set of moves available to agent i at that state and $\delta : S \times Moves \times Moves \mapsto 2^S \setminus \emptyset$ is a transition function, that given a state and a pair of moves chosen by the agents, returns a subset of possible successor states. The agents select their moves simultaneously and independently. If only one of the two agents has a choice of moves at each state, then we get a turn-based synchronous ATS. We now present an example from [AHK97] for turn-based synchronous alternating transition systems.

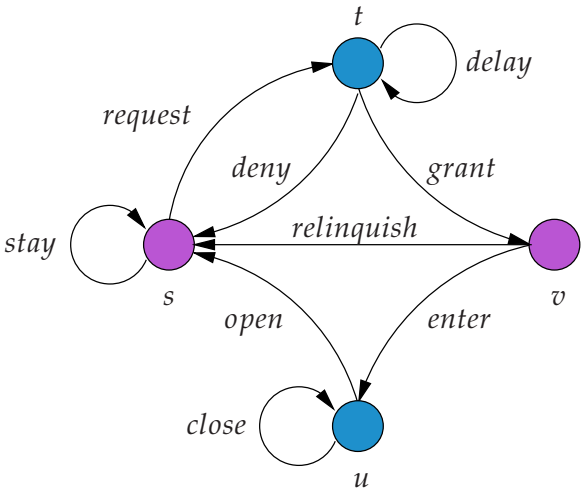


Figure 2.2: Alternating Transition System that models a train entering a railroad crossing.

Example 2 In Figure 2.2 is an alternating transition system that models a railroad crossing. It is a turn-based synchronous process. The purple states are the train states and the blue states are those of the controller. In state s, a train state, the gate is open and the train has two possible moves; stay outside the gate or request permission to enter the gate. In state t, a controller state, the controller has a choice of three moves; either grant the train

permission to enter the gate, delay making a decision or deny the train permission to enter the gate. In state v, a train state, the train can either choose to enter the gate or relinquish its request to enter the gate. Finally, in state u, a controller state, the controller can either keep the gate closed or open it for other trains. ■

Unlike in the case of labelled transition systems where both the train and the controller are assumed to co-operate, in alternating transition systems we reason about properties that the controller can satisfy against all possible behaviors of the train. We can therefore think of this as a zero-sum game played between the controller and the train, where the train is not expected to be angelic and may in fact be demonic.

2.2.1 Alternating Simulation

We recall that in labelled transition systems, since we interpret all agents as co-operating, we defined simulation in terms of the ability to match all transitions from a simulated state with at least one transition from the simulating state. In alternating simulation [AHKV98], since we interpret the agents as being antagonistic, we extend the definition of simulation as follows: Let $(S, s_0, \mathit{Moves}, \Gamma_1, \Gamma_2, \delta)$ and $(S', s'_0, \mathit{Moves}, \Gamma'_1, \Gamma'_2, \delta')$ be two alternating transition systems. A relation $R \subseteq S \times S'$ is an alternating simulation relation if $s \preceq s'$ implies that,

1. $s \equiv s'$ and

2. $\forall a \in \Gamma_1(s) \,.\, \exists b \in \Gamma_1(s') \,.\, \forall c \in \Gamma_2(s') \,.\, \exists d \in \Gamma_2(s) \,.\, \delta(s, a, d) \preceq \delta(s', b, c)$.

Notice that agent 1 picks moves first at both states before agent 2 does. The definition therefore applies to agent 1 alternating simulation; the case for agent 2 alternating simu-

lation is defined symmetrically and the two versions may not yield the same simulation relation.

2.2.2 Game Interpretation of Alternating Simulation

The intuition behind alternating simulation is captured by the following game theoretic interpretation. Let $(S, s_0, \mathit{Moves}, \Gamma_1, \Gamma_2, \delta)$ and $(S', s'_0, \mathit{Moves}, \Gamma'_1, \Gamma'_2, \delta')$ be two alternating transition systems. Consider a two-player game whose positions are pairs $\langle s, s' \rangle \in S \times S'$ of states. The initial position of the game is $\langle s_0, s'_0 \rangle$. The game is played between an antagonist and a protagonist and proceeds in rounds. Each round involves the following steps, assuming that the current position is $\langle s, s' \rangle$,

- The antagonist chooses an agent 1 move a at state s,
- The protagonist chooses an agent 1 move b at state s',
- The antagonist chooses an agent 2 move c at state s',
- The protagonist chooses an agent 2 move d at state s such that $\delta(s, a, d) \equiv \delta(s', b, c)$.

If the game proceeds ad infinitum, then the antagonist loses and hence the protagonist wins. Otherwise, the game reaches a state where the protagonist is unable to pick a move at state s as required, leading to her loss. Since the antagonist is assumed to be adversarial, we require that, at both states, for all moves that can be selected by the antagonist, there exist a move for the protagonist, such that the destination states are in the alternating simulation relation.

2.3 Logics

In our definitions of simulation and bisimulation relations we stated that two states that are in a relation should have identical predicate values. We also stated that states in a relation, such as the bisimulation relation, generate exactly the same set of behaviors. Recall that we defined the behavior of a finite state system as a set of paths where each path is a sequence of states. Since we have introduced and formalized two very successful formal models together with simulation and bisimulation relations in those models, we now address the question of formalizing behavior. What does it mean mathematically to say that two states generate the same set of behaviors? The answer lies in logics. Logic has a been studied as a discipline of philosophy since the times of Aristotle. A subfield of logic is symbolic logic where symbolic abstractions are used to study formal inference. Mathematical logic, an extension of symbolic logic, finds its origins in the works of George Boole and Augustus de Morgan that lead to the framework to study the foundations of mathematics. The connection between mathematical logic and computer science owes its origins to Hilbert, who in 1928 posed the Entscheidungsproblem, which asked for a procedure that could decide whether or not a formal mathematical statement was true or false.

A logic in its simplest form consists of a set U, possibly infinite, of *symbols* that include connectives $\rightarrow, \vee, \wedge, \neg, \leftrightarrow, (,)$ and a finite set of *generating* functions $\mathcal{F} = \{f_1, f_2, \ldots, f_n\}$, where $1 \leq i \leq n$ is the *arity* or number of arguments taken by function $f_i : U^i \mapsto U$. The functions in $\mathcal{F}$ generate *expressions* from the set of symbols. A *well formed formula* or *wff* is a grammatically correct expression; one that can be generated from the

symbols and the functions in $\mathcal{F}$. The set of all wffs so generated forms an *inductive set*. If U is the universal set of everything, including expressions, a set $C \subseteq U$ is an *inductive* set if, given an initial set $B \subseteq C$ and a set of generating functions $\mathcal{F}$, the following conditions hold:

- $B \subseteq C$, and
- C is closed under the functions in $\mathcal{F}$.

An inductive set is said to be *freely generated* if the ranges of the functions in $\mathcal{F}$ and the set B are pairwise disjoint. For example, the set of natural numbers $\mathbb{N}$, with $B = 0$ and the successor function $S : \mathbb{N} \mapsto \mathbb{N}$, where $S(n) = n + 1$ for all $n \in \mathbb{N}$ is a freely generated inductive set. We can define *recursive functions* on a freely generated inductive set $C \subseteq U$ using the *recursion theorem*. It states that given C and $\mathcal{F}_C$, a set of generating functions for C, a set V and $\mathcal{F}_V$, a set of generating functions for V, and a function $h : B \mapsto V$, there exists a unique extension h' of h such that h' is a *homomorphism* from C (with operations $\mathcal{F}_C$) to V (with operations $\mathcal{F}_V$). The concepts of induction and recursion are fundamental to the study of logic and computability. For instance, given a set of boolean expressions S, extending a truth assignment over the variables in the expressions to a truth assignment over the expressions in S relies on the recursion theorem.

A set of expressions Σ is *decidable* if there is a procedure, that given an expression ϵ, will decide whether or not $\epsilon \in \Sigma$. A set of expressions Σ is *recursively enumerable*, or *RE*, if there is a procedure that given an expression ϵ will answer "yes" if $\epsilon \in \Sigma$; we can enumerate the set of all expressions in an inductive set and check if each new expression we form is the same as ϵ or not. If $\epsilon \notin \Sigma$, then the procedure may not terminate; it is

therefore said to be *semi-decidable*. An important result in computability theory is that if a set of expressions Σ and its complement (with respect to the set of all expressions) are recursively enumerable, then the set Σ is decidable (Kleene's theorem). Therefore, if a recursively enumerable set is closed under complementation, in other words it is RE and co-RE, it is decidable. These concepts have played a critical role in various decidability results, notably the decidability of one of the most expressive decidable logics: the monadic second order logic of n successors or SnS for short.

While mathematical logic deals with infinite structures where results such as the *compactness theorem* and *Gödel's completeness theorem* hold, the models that are presented to computers are necessarily finite. This has lead to the development of *finite model theory* and *descriptive complexity*, which is the study of computational complexity of a problem in terms of the complexity of the logical languages used to describe the problem. There are deep links between logic and computability. For instance, the result of Fagin states that the class NP is precisely the set of languages expressible in the existential fragment of second-order logic. As a substantial part of this thesis deals with metrics and the logics that characterize them (Logical Characterizations of Simulation and Bisimulation presented later in this chapter), and in formalizing desired behaviors for synthesis, we now present a brief introduction to logics. We begin with first order logic, cover temporal logics and conclude with the μ-calculus and the Hennessy-Milner logic.

2.3.1 First-order Logic

First order logic is expressive enough to capture descriptive set theory, from which most mathematical theories can be derived. First order logic consists of infinitely

many distinct symbols, arranged as follows:

1. *Logical symbols*
   - Parentheses: (,),
   - Sentential connectives: $\Rightarrow, \neg$,
   - Variables $V = \{v_1, v_2, \ldots\}$,
2. *Parameters*
   - Quantifier $\mathcal{Q}$: $\forall$,
   - For each positive integer n, a possibly empty set of n-ary predicates P_n,
   - Constants or 0-ary functions,
   - For each positive integer n, a possibly empty set of n-ary functions f_n.

$\mathcal{F}_f$ is a set of term building functions, one for each function f_n. The *terms* in first order logic are then the set of all expressions built from variables and constants by zero or more applications of the functions in $\mathcal{F}_f$; they are defined inductively. The *atomic formulas* are expressions obtained by applying predicates to terms. The well formed formulas are constructed inductively from atomic formulas and sentential connectives and parentheses using the following set of formula generating functions, $\mathcal{E}_\neg(\alpha) = (\neg\alpha)$, $\mathcal{E}_\rightarrow(\alpha, \beta) = (\alpha \rightarrow \beta)$ and $\mathcal{Q}_a(\alpha) = \forall a.\alpha$ for expressions α and β. A wff is *closed* if it has no free variables, that is all variables are bound to quantifiers. Closed wffs are called *sentences* as opposed to formulas which have free variables. A *structure* $\mathfrak{A}$ for first-order logic is a function that assigns to the quantifier $\forall$ a universe U, to each n-ary predicate P_n a relation $P_n^{\mathfrak{A}} \subseteq U^n$ of n tuples,

to each constant symbol an element of U and to each n-ary function f_n an n-ary operation $f_n^{\mathfrak{A}} : U^n \mapsto U$. An *interpretation* $\xi : V \mapsto U$, is an assignment from variables to U; an interpretation therefore assigns values to unbound variables in a formula. A structure $\mathfrak{A}$ is a *model* of a wff φ iff the translation of φ under $\mathfrak{A}$ over all interpretations of the unbound variables in φ is true, written $\models_{\mathfrak{A}} \varphi$. If φ is true under all structures, then φ is considered *valid* in first-order logic. We say that a set of wffs Γ *logically implies* a wff φ, written $\Gamma \models \varphi$, if in every structure, every interpretation ξ that satisfies every member of Γ also satisfies φ with the same interpretation ξ; in other words every model of every member of Γ is also a model of φ. For a complete treatment of the material presented in this subsection we refer the reader to [End01].

2.3.2 LTL, CTL and ATL

In the remainder of this section, we confine ourselves to finite structures, such as the LTS and ATS that we introduced earlier. Since LTS and ATS are effective models of non-terminating systems, such models have a finite set of states generating a set of infinite paths that constitute the behavior of the system. A single path is then a particular computation of the system and is simply a sequence of visited states. The predicates in these models are subsets of states where they evaluate to true. This implies that the predicates may change in truth value along a path. Therefore, to describe the behavior of a system modeled as an LTS or an ATS, we require a logic that can describe paths in terms of the predicates that may change in value along a given path. Since paths evolve over time, the logics used to describe them are called *temporal logics* [MP91, MP95, AHK97]. The logics LTL, CTL and ATL are particular temporal logics with varying expressive power. These logics are also

called *modal logics* as they have modalities that qualify the truth value of formulas.

The logic LTL, or *Linear Temporal Logic*, is a temporal logic that is used to describe sets of paths. LTL consists of a finite set of predicates, logical connectives such as $\Rightarrow$ and $\neg$ and modalities $\bigcirc$, $\Box$, $\Diamond$, $\mathcal{U}$ and $\mathcal{R}$ that are defined below. Taking Φ as the set of all well-formed formulas in LTL, for wffs $\varphi_1, \varphi_2 \in \Phi$, the modalities in LTL have the following meaning:

1. $\bigcirc\varphi_1$ is satisfied by a path if φ_1 holds in the next state of the path.

2. $\Box\varphi_1$ is satisfied by a path if φ_1 holds in all states of the path; this property is referred to as a *safety property*.

3. $\Diamond\varphi_1$ is satisfied by a path if φ_1 holds in at least one state of the path; this property is referred to as a *reachability property*.

4. $\varphi_1 \mathcal{U} \varphi_2$ is satisfied by a path if φ_1 holds at all states of the path at least *until* φ_2 holds.

5. $\varphi_1 \mathcal{R} \varphi_2$ is satisfied by a path if φ_2 holds at all states until and including the state where φ_1 holds.

An LTL formula is satisfied by a path iff it is satisfied at the first state of the path. There are no explicit quantifiers in LTL with all formulas being implicitly universally quantified.

Example 3 Consider two states s, t in an LTS with a transition from state s to t with state t being absorbing; the system transitions from s to t and then remains in t forever. If a predicate p holds in t and not in s, then for all paths of the LTS, $\Box p$ is not satisfied whereas $\Diamond p$ is satisfied. ■

The logic LTL can be used to express many interesting properties of transition systems. For example a *liveness property*, expressed as $\Box\Diamond win$, for a predicate *win*, is satisfied if in all paths, *win* holds infinitely often. Similarly a *fairness property*, expressed as $\Box\Diamond request \Rightarrow \Box\Diamond grant$, for predicates *request* and *grant*, is satisfied if in all paths where *request* is true infinitely often, we have *grant* is also true infinitely often. A safety property can be used to express that something good always happens or something bad never happens. A reachability property can be used to express that something good happens eventually. A liveness property can be used to express that something good happens infinitely often. A fairness property can be used to express that, if a *request* for a resource is true infinitely often, then so is a *grant* of that request; the request could be for a mutex in a multi-threaded program for example, in which case satisfaction of the property implies that for every such request the mutex is granted eventually.

If we consider the set of all computation paths of an LTS, then we can represent them as a tree, that is rooted at a distinguished starting state and contains all computation paths that start at that state. The logic CTL, or *Computation Tree Logic*, also called a *branching-time temporal logic*, describes properties of computation trees of an LTS. It is a state logic, in that all formulas are state formulas, as opposed to LTL which is a path logic. Similar to LTL, it consists of a set of predicates, logical connectives and path modalities. In addition, it includes the path quantifiers $\forall$ for universal quantification and $\exists$ for existential quantification. All quantified path formulas are state formulas and all modalities must be immediately preceded by a path quantifier. CTL then consists of all state formulas that can be generated from the atomic propositions, logical connectives, path modalities and

quantifiers. For example, given a wff φ, satisfaction of the CTL formula $\forall\Diamond\varphi$ requires that a state satisfying φ be visited in all computation paths. Similarly, satisfaction of the CTL formula $\exists\Diamond\varphi$ requires that there exists a path that visits a state satisfying φ.

While LTL and CTL are logics that can be used to describe properties of labelled transition systems, the logic ATL, or *Alternating-Time Temporal Logic*, can be used to describe properties of alternating transition systems. We recall that an LTS is a model for closed systems, whereas an ATS is a model for open systems, such as a set of components interacting with an unknown, possibly adversarial environment. Given a set Σ of components including the environment, for a set $A \subseteq \Sigma$, with φ being a set of computations expressed in LTL, given a state s of the ATS, consider a game between a protagonist and an antagonist that starts in state s. At each round, the protagonist resolves the choices controlled by the components in A (*internal non-determinism*) and the antagonist resolves the choices controlled by components in $\Sigma \setminus A$ (*external non-determinism*). If the resulting infinite path is in the set of computations φ, then the protagonist wins, otherwise the antagonist wins. If the protagonist has a *winning strategy* such that she can ensure the computation path is in φ against all possible ways of resolving external non-determinism, we say that the path satisfies the ATL formula $\langle\langle A\rangle\rangle\varphi$ at state s. The path quantifier $\langle\langle A\rangle\rangle$ is therefore the set of all computation paths that can be forced by the components in A against all possible behaviors of the components in $\Sigma \setminus A$. While CTL provides either universal or existential path quantifiers that assume co-operative behaviors, ATL generalizes the path quantifiers by subsets of components. The existential path quantifier in CTL is generalized to $\langle\langle\Sigma\rangle\rangle$ and the universal path quantifier is generalized to $\langle\langle\emptyset\rangle\rangle$. Therefore

the universal path quantifier exactly corresponds to a single component $\langle\langle\emptyset\rangle\rangle$ and the existential path quantifier corresponds to a single component $\langle\langle\{sys\}\rangle\rangle$. For example, taking $\Sigma = \{a, b, c\}$, and the starting state as s, the ATL formula $\langle\langle a\rangle\rangle\Diamond\varphi$ is satisfied, if component a has a strategy against components b and c such that all paths starting at s satisfy φ. Similarly, $\neg\langle\langle b, c\rangle\rangle\Box\varphi$ is satisfied if the coalition of b and c do not have a strategy against a to ensure all computation paths satisfy φ. The logic ATL is better suited for compositional reasoning than CTL. If a component a of a system satisfies the CTL formula $\exists\Diamond\varphi$, we cannot conclude that any composite system that includes a will also satisfy $\exists\Diamond\varphi$. But if the ATL formula $\langle\langle a\rangle\rangle\Diamond\varphi$ is satisfied by a, then any composite system that includes a satisfies the CTL formula $\exists\Diamond\varphi$.

2.3.3 μ-calculus

To facilitate introducing the μ-calculus, we first present some preliminaries [AN01]. A set $\mathcal{L}$ and an order relation $\leq$ forms a lattice $(\mathcal{L}, \leq)$, if for every pair of elements $a, b \in \mathcal{L}$, there exists a *least upper bound* written as $a \vee b$ and a *greatest lower bound* written as $a \wedge b$ such that $a \vee b \in \mathcal{L}$ and $a \wedge b \in \mathcal{L}$. In general, for a subset $X \subseteq \mathcal{L}$, we define $\bigvee X = x^* \in \mathcal{L} \mid \forall y \in X \,.\, \forall z \in \mathcal{L} \,.\, ((y \leq x^*) \text{ and } (y \leq z \Rightarrow x^* \leq z))$. Similarly for a subset $X \subseteq \mathcal{L}$, we define $\bigwedge X = x_* \in \mathcal{L} \mid \forall y \in X \,.\, \forall z \in \mathcal{L} \,.\, ((y \geq x_*) \text{ and } (y \geq z \Rightarrow x_* \geq z))$. A lattice $(\mathcal{L}, \leq)$ is a *complete lattice* iff for all subsets $X \subseteq \mathcal{L}$, $\bigvee X \in \mathcal{L}$ and $\bigwedge X \in \mathcal{L}$. Consider a function $f : \mathcal{L} \mapsto \mathcal{L}$. The function is monotonically increasing iff for all $a, b \in \mathcal{L}$ such that $a \leq b$, we have $f(a) \leq f(b)$. Similarly a function $g : \mathcal{L} \mapsto \mathcal{L}$ is monotonically decreasing iff for all $a, b \in \mathcal{L}$ such that $a \geq b$, we have $f(a) \geq f(b)$. We have the following theorem due to Bronisław Knaster and Alfred Tarski.

Theorem 1 (Knaster-Tarski) *All monotonic functions on a complete lattice have a fixed point.*

As a direct consequence of the theorem, we have, for a monotonically increasing function $f : \mathcal{L} \mapsto \mathcal{L}$ over a complete lattice $(\mathcal{L}, \leq)$, there exists a least fixpoint $a \in \mathcal{L}$ such that $f(a) = a$. Similarly for a monotonically decreasing function $g : \mathcal{L} \mapsto \mathcal{L}$ over a complete lattice $(\mathcal{L}, \leq)$, there exists a greatest fixpoint $b \in \mathcal{L}$ such that $g(b) = b$. We remark that, given a set $\mathcal{L}$, the partially ordered set $(2^{\mathcal{L}}, \subseteq)$ of power sets of $\mathcal{L}$, ordered by subset inclusion, forms a complete lattice. Therefore, any monotonically defined function $f : 2^{\mathcal{L}} \mapsto 2^{\mathcal{L}}$ has a fixed point. These are some of the most fundamental results at the heart of formal analysis. We now have all the ingredients to present the μ-calculus. The μ-calculus [Koz83b, SE89] is defined as follows:

1. A set P of propositions,
2. A set V of variables,
3. Logical connectives $\neg$, $\vee$ and $\wedge$,
4. $\langle A \rangle p$ and $[A]p$, where $A \subseteq \Sigma$ is a subset of program letters (components) and p is any formula and
5. $\mu X.f(X)$ and $\nu X.f(X)$, where $f(X)$ is any formula that is syntactically monotone in the variable X, i.e., all occurrences of X in $f(X)$ fall under an even number of negations.

The operators μ and ν are the least and greatest fixpoint operators respectively. The sentences in the logic are the set of closed formulas; formulas where all variables are bound

to either a least or a greatest fixpoint operator. Further, the sentences of the μ-calculus are interpreted over non-deterministic LTS. Let S be the states of an LTS. Since all propositions and variables are subsets of states, the underlying algebraic structure is that of a complete lattice $(2^S, \subseteq)$. Given that all formulas in the scope of a fixpoint operator are monotone, by Theorem 1, the fixpoints always exist. The modal μ-calculus is a decidable logic that can be used to specify all ω-regular properties. The ω-regular languages, which are regular languages of infinite words, are the most general class of languages that subsume the set of all temporal languages we have presented so far. The μ-calculus therefore subsumes all temporal logics.

Example 4 Consider an LTS, with set of states S and a set $R \subseteq S$ of desired states that we would like to reach from an initial state $s_0 \in S$. Let pre : $2^S \mapsto 2^S$ be a state set transformer, that given a set of states $T \subseteq S$, returns the set of states in S that can reach T after one transition. It is easy to verify that pre is non-decreasing and hence monotone. If we take $X = R$, the set of desired reachability states, then $X = \mu X.\text{pre}(X)$ is precisely the set of states from which we can reach a state in R. If $s_0 \in X$, we conclude that there exists a path from s_0 to reach a state in R.

2.3.4 Hennessy-Milner Logic

To facilitate the presentation of logical characterizations of simulation and bisimulation we present the following logic due to Hennessy and Milner [HM85].

$$\varphi \ ::= T \mid \neg\varphi \mid \bigwedge_{i \in \mathbb{N}} \varphi \mid \langle a \rangle \varphi \ .$$

The formula T is satisfied at every state of an LTS. The modal formula $\langle a \rangle \varphi$ is satisfied by a state $s \in S$, if there exists an a labelled transition from s to a state $t \in S$, such that φ is satisfied at t.

2.3.5 Logical Characterizations of Simulation and Bisimulation

The usefulness of a simulation or a bisimulation relation stems from its logical characterization. For two bisimilar states s and t, all formulas in a suitable logic, given a variable interpretation, should have identical truth values. For LTS, the bisimulation relation is characterized by the Hennessy-Milner Logic. Given an LTS with set of states S, if $s \simeq t$, then for all $\varphi \in HML$, $s \models \varphi$ implies $t \models \varphi$ and vice versa. Given an LTS with set of states S, if $s \preceq t$, then for all $\varphi \in HML$, $s \models \varphi$ implies $t \models \varphi$. Therefore, the logic HML logically characterizes the simulation and bisimulation relations in labelled transition systems.

From the result of [AHKV98], the logic ATL logically characterizes alternating simulation. Given an ATS with set of states S, if $s \preceq t$, then for all $\varphi \in ATL$, $s \models \varphi$ implies $t \models \varphi$.

2.4 Probabilistic Systems and Games

A probabilistic labelled transition system (PLTS) is a formal model of a finite state process that interacts with its environment. Formally, a PLTS is a tuple $(S, s_0, \mathcal{A}, \delta)$, where S is a finite set of states, s_0 is the initial state, $\mathcal{A}$ is a finite set of labels or actions and $\delta : S \times \mathcal{A} \mapsto \text{Dist}(S)$ is a probabilistic transition function, that given a state s and an action

a returns a probability distribution $\delta(s, a)$ over S. We can think of a probabilistic system as a single player game, where the player is the environment, or the *controller*, and the system is probabilistic; for every action a chosen by the controller at a state $s \in S$, the system transitions to a state $t \in S$ such that $\delta(s, a)(t) > 0$. The system in this case can be thought of as a second player, but one who has a single action at every state of the system. We remark that, if the controller fixes her strategy such that whenever the system is in a state s, she always chooses the same action $a \in \mathcal{A}$ at s, then the PLTS reduces to a Markov Chain which is an ordinary stochastic process. A generalization of probabilistic systems is then a two-player turn-based game structure, where the set of states S is partitioned into controller states and system states. At every controller state, the environment chooses an action and at every system state, the system chooses an action. The transition function returns a distribution over the set of states S for every state and player action. A concurrent game structure generalizes a turn-based game structure where both players have a choice of moves at each state. Given a state and a pair of moves, chosen simultaneously and independently by the two players, the transition function returns a distribution over the set of states S. We define game structures formally in the next section of this thesis. We conclude this chapter by illustrating that in concurrent games, it may be necessary for a player to pick lotteries over her moves at a state to guarantee positive transition probabilities. We show through the following example, called *matching pennies*, that by using deterministic strategies, player 1 cannot achieve her objective with positive probability.

Example 5 (Matching Pennies) Consider the concurrent game structure in Figure 2.3. Both players have moves $\{a, b\}$ at state s. If both players pick matching moves, then the

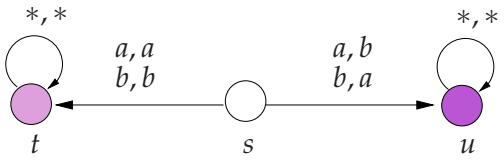


Figure 2.3: The need for randomized strategies.

destination state is t, otherwise it is u. The transitions are deterministic. If player 1 wants to reach state t from state s with positive probability, then by using pure strategies, she can never ensure that the game transitions to state t; the transition probability she can guarantee is 0. This is because, for every player 1 pure move at s, there exists a move for player 2 such that the destination state is u and not t; player 1 picks a and player 2 picks b or player 1 picks b and player 2 picks a. If player 1 randomizes and picks either a or b with equal probability, then she can guarantee a transition to state t with probability $\frac{1}{2}$. This example illustrates that lotteries over pure moves, also called mixed moves, may be necessary to ensure positive transition probabilities in concurrent games. ■

Part I

Game Relations and Metrics

Chapter 3

Definitions

We will develop metrics for game structures over a set S of states. We start with some preliminary definitions.

For a finite set A, let $\mathrm{Dist}(A) = \{p : A \mapsto [0,1] \mid \sum_{a \in A} p(a) = 1\}$ denote the set of probability distributions over A. We say that $p \in \mathrm{Dist}(A)$ is *deterministic* if there is $a \in A$ such that $p(a) = 1$.

For a set S, a *valuation over* S is a function $f : S \mapsto [0,1]$ associating with every element $s \in S$ a value $0 \leq f(s) \leq 1$; we let $\mathcal{F}$ be the set of all valuations. For $c \in [0,1]$, 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 complete lattice.

Given $a, b \in \mathbb{R}$, we write $a \sqcup b = \max\{a, b\}$, and $a \sqcap b = \min\{a, b\}$; we also let $a \oplus b = \min\{1, \max\{0, a + b\}\}$ and $a \ominus b = \max\{0, \min\{1, a - b\}\}$. We extend $\sqcap, \sqcup, +, -, \oplus, \ominus$ to valuations by interpreting them in pointwise fashion.

A *directed metric* is a function $d : S^2 \mapsto \mathbb{R}_{\geq 0}$ which satisfies $d(s,s) = 0$ and $d(s,t) \leq d(s,u) + d(u,t)$ for all $s,t,u \in S$. We denote by $\mathcal{M} \subseteq S^2 \mapsto \mathbb{R}$ the space of all metrics; this space, ordered pointwise, forms a lattice which we indicate with $(\mathcal{M}, \leq)$. Given a metric $d \in \mathcal{M}$, we denote by $\breve{d}$ its *opposite* version, defined by $\breve{d}(s,t) = d(t,s)$ for all $s,t \in S$; we say that d is symmetrical if $d = \breve{d}$.

3.1 Game Structures

We assume a fixed, finite set $\mathcal{V}$ of *observation variables*. [1] A (two-player, concurrent) *game structure* $G = \langle S, [\cdot], \mathit{Moves}, \Gamma_1, \Gamma_2, \delta \rangle$ consists of the following components [AHK02, dAHK98]:

- A finite set S of states.
- A variable interpretation $[\cdot] : \mathcal{V} \times S \mapsto [0,1]$, which associates with each variable $v \in \mathcal{V}$ a valuation $[v]$.
- A finite set *Moves* of moves.
- Two move assignments Γ_1, Γ_2: $S \mapsto 2^{\mathit{Moves}} \setminus \emptyset$. For $i \in \{1,2\}$, the assignment Γ_i associates with each state $s \in S$ the nonempty set $\Gamma_i(s) \subseteq \mathit{Moves}$ of moves available to player i at state s.
- A probabilistic transition function δ: $S \times \mathit{Moves} \times \mathit{Moves} \mapsto \mathrm{Dist}(S)$, 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

[1] The only reason why we assume a fixed set of variables, rather than specifying one set for each game structure, is notational convenience: later, when introducing logics for games, it will be simple to ensure that the logic formulas refer to the same set of variables as the games on which they are evaluated.

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 denote by $\tau(s, a_1, a_2) = \{t \in S \mid \delta(s, a_1, a_2)(t) > 0\}$ the set of *destination states* when actions a_1, a_2 are chosen at s. The *propositional distance* $p(s, t)$ between two states $s, t \in S$ is the maximum difference in the valuation of any variable and is defined as follows:

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

The kernel of the propositional distance naturally 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 $G = \langle S, [\cdot], \mathit{Moves}, \Gamma_1, \Gamma_2, \delta \rangle$. For player $i \in \{1, 2\}$, we write $\sim i = 3 - i$ for the opponent. We also consider the following subclasses of game structures.

- *Turn-based game structures.* A game structure G is *turn-based* if we can write S as the disjoint union of two sets: the set S_1 of *player 1 states*, and the set S_2 of *player 2 states*, such that $s \in S_1$ implies $|\Gamma_2(s)| = 1$, and $s \in S_2$ implies $|\Gamma_1(s)| = 1$, and further, there is a special variable $\mathit{turn} \in \mathcal{V}$, such that $[\mathit{turn}](s) = 1$ iff $s \in S_1$, and $[\mathit{turn}](s) = 0$ iff $s \in S_2$: thus, the variable *turn* indicates whose turn it is to play at a state.

- *Markov decision processes.* A game structure G is a *Markov decision process* (MDP) [Der70] if only one of the two players has a choice of moves. 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 (sin-

gle) move of the player without a choice of moves, and write $\delta(s, a)$ for the transition function.

– *Deterministic game structures.* A game structure G is *deterministic* if, for all $s \in S$, $a_1 \in \mathit{Moves}$, and $a_2 \in \mathit{Moves}$, there exists a $t \in S$ such that $\delta(s, a_1, a_2)(t) = 1$; we denote such t by $\tau(s, a_1, a_2)$. We sometimes call *probabilistic* a general game structure, to emphasize the fact that it is not necessarily deterministic.

Note that MDPs can be seen as turn-based games by setting $[\mathit{turn}] = \mathbf{1}$ for 1-MDPs and $[\mathit{turn}] = \mathbf{0}$ for 2-MDPs.

Pure and mixed moves. A *mixed move* is a probability distribution over the moves available to a player at a state. We denote by $\mathcal{D}_i(s) = \mathrm{Dist}(\Gamma_i(s))$ the set of mixed moves available to player $i \in \{1, 2\}$ at $s \in S$. The moves in *Moves* are called *pure moves*, in contrast to mixed moves. We extend the transition function to mixed moves. For $s \in S$ and $x_1 \in \mathcal{D}_1(s)$, $x_2 \in \mathcal{D}_2(s)$, we write $\delta(s, x_1, x_2)$ for the next-state probability distribution induced by the mixed moves x_1 and x_2, defined for all $t \in S$ by

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

In the following, we sometimes restrict the moves of the players to pure moves. We identify a pure move $a \in \Gamma_i(s)$ available to player $i \in \{1, 2\}$ at a state s with a deterministic distribution that plays a with probability 1.

The deterministic setting. The *deterministic setting* is obtained by considering deterministic game structures, with players restricted to playing pure moves.

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:

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

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 $s_0, s_1, s_2, ...$ 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 $\sigma \in S^+$ 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 for all $\sigma \in S^*$ and states $s \in S$, if $\pi_i(\sigma s)(a) > 0$, then $a \in \Gamma_i(s)$. A strategy is a *stationary strategy*, also known as a *memoryless strategy*, if $\pi_i(\sigma s) = \pi_i(s)$ for all $s \in S$; the move selected by player i at every state of the game is independent of history. 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. We denote by $\Pr_s^{\pi_1,\pi_2}(\cdot)$ the probability of a measurable event (sets of paths) with respect to this process, and denote by $\mathbb{E}_s^{\pi_1,\pi_2}(\cdot)$ the associated expectation operator. For $k \geq 0$, we let $X_k : \Sigma \to S$ 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:

$$\mathbb{E}_s^{x_1,x_2}(f) = \sum_{t \in S} \delta(s, x_1, x_2)(t)\, f(t) \ .$$

For a game structure G, for $i \in \{1, 2\}$ we define the *valuation transformer* $\mathrm{Pre}_i : \mathcal{F} \mapsto \mathcal{F}$ by, for all $f \in \mathcal{F}$ and $s \in S$,

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

Intuitively, $\mathrm{Pre}_i(f)(s)$ is the maximal expectation player i can achieve of f after one step from s: this is the classical "one-day" or "next-stage" operator of the theory of repeated games [FV97]. We also define a *deterministic* version of this operator, in which players are forced to play pure moves:

$$\mathrm{Pre}_i^{\Gamma}(f)(s) = \max_{x_i \in \Gamma_i(s)} \min_{x_{\sim i} \in \Gamma_{\sim i}(s)} \mathbb{E}_s^{x_1,x_2}(f) \ .$$

3.2 Quantitative μ-calculus

We consider the set of properties expressed by the *quantitative μ-calculus* ($q\mu$). As discussed in [Koz83a, dAM04, MM04], a large set of properties can be encoded in $q\mu$, spanning from basic properties such as maximal reachability and safety probability, to the maximal probability of satisfying a general ω-regular specification.

Syntax. The syntax of quantitative μ-calculus is defined with respect to the set of observation variables $\mathcal{V}$ as well as a set *MVars* of *calculus variables*, which are distinct from the

observation variables in $\mathcal{V}$. The syntax is given as follows:

$$\varphi ::= c \mid v \mid V \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \varphi \oplus c \mid \varphi \ominus c \mid \mathrm{pre}_1(\varphi) \mid \mathrm{pre}_2(\varphi) \mid \mu V.\, \varphi \mid \nu V.\, \varphi$$

for constants $c \in [0,1]$, observation variables $v \in \mathcal{V}$, and calculus variables $V \in \mathit{MVars}$. In the formulas $\mu V.\, \varphi$ and $\nu V.\, \varphi$, we furthermore require that all occurrences of the bound variable V in φ occur in the scope of an even number of occurrences of the complement operator $\neg$. A formula φ is *closed* if every calculus variable V in φ occurs in the scope of a quantifier μV or νV. From now on, with abuse of notation, we denote by $q\mu$ the set of closed formulas of $q\mu$. A formula is a *player i formula,* for $i \in \{1,2\}$, if φ does not contain the $\mathrm{pre}_{\sim i}$ operator; we denote with $q\mu_i$ the syntactic subset of $q\mu$ consisting only of closed player i formulas. A formula is in *positive form* if the negation appears only in front of observation variables, i.e., in the context $\neg v$; we denote with $q\mu^+$ and $q\mu_i^+$ the subsets of $q\mu$ and $q\mu_i$ consisting only of positive formulas.

We remark that the fixpoint operators μ and ν will not be needed to achieve our results on the logical characterization of game relations. They have been included in the calculus because they allow the expression of many interesting properties, such as safety, reachability, and in general, ω-regular properties. The operators $\oplus$ and $\ominus$, on the other hand, are necessary for our results.

Semantics. A variable valuation ξ: $\mathit{MVars} \mapsto \mathcal{F}$ is a function that maps every variable $V \in \mathit{MVars}$ to a valuation in $\mathcal{F}$. We write $\xi[V \mapsto f]$ for the valuation that agrees with ξ on all variables, except that V is mapped to f. Given a game structure G and a variable valuation ξ, every formula φ of the quantitative μ-calculus defines a valuation $[\![\varphi]\!]^G_\xi \in \mathcal{F}$

(the superscript G is omitted if the game structure is clear from the context):

$$[\![c]\!]_\xi = \mathbf{c}$$

$$[\![v]\!]_\xi = [v]$$

$$[\![V]\!]_\xi = \xi(V)$$

$$[\![\neg\varphi]\!]_\xi = \mathbf{1} - [\![\varphi]\!]_\xi$$

$$[\![\varphi\{{}^{\oplus}_{\ominus}\}c]\!]_\xi = [\![\varphi]\!]_\xi\{{}^{\oplus}_{\ominus}\}\mathbf{c}$$

$$[\![\varphi_1\,\{{}^{\vee}_{\wedge}\}\,\varphi_2]\!]_\xi = [\![\varphi_1]\!]_\xi\,\{{}^{\sqcup}_{\sqcap}\}\,[\![\varphi_2]\!]_\xi$$

$$[\![\mathrm{pre}_i(\varphi)]\!]_\xi = \mathrm{Pre}_i([\![\varphi]\!]_\xi)$$

$$[\![\{{}^{\mu}_{\nu}\}V.\,\varphi]\!]_\xi = \{{}^{\inf}_{\sup}\}\{f \in \mathcal{F} \mid f = [\![\varphi]\!]_{\xi[V\mapsto f]}\}$$

where $i \in \{1,2\}$. The existence of the fixpoints is guaranteed by the monotonicity and continuity of all operators and can be computed by Picard iteration [dAM04]. If φ is closed, $[\![\varphi]\!]_\xi$ is independent of ξ, and we write simply $[\![\varphi]\!]$.

We also define a *deterministic* semantics $[\![\cdot]\!]^\Gamma$ for $q\mu$, in which players can select only pure moves in the operators pre_1, pre_2. $[\![\cdot]\!]^\Gamma$ is defined as $[\![\cdot]\!]$, except for the clause

$$[\![\mathrm{pre}_i(\varphi)]\!]^\Gamma_\xi = \mathrm{Pre}^\Gamma_i([\![\varphi]\!]^\Gamma_\xi)\ .$$

Example 6 Given a set $T \subseteq S$, the *characteristic valuation* $\mathbf{T}$ of T is defined by $\mathbf{T}(s) = 1$ if $s \in T$, and $\mathbf{T}(s) = 0$ otherwise. With this notation, the maximal probability with which player $i \in \{1,2\}$ can ensure eventually reaching $T \subseteq S$ is given by $[\![\mu V.(\mathbf{T} \vee \mathrm{pre}_i(V))]\!]$, and the maximal probability with which player i can guarantee staying in T forever is given by $[\![\nu V.(\mathbf{T} \wedge \mathrm{pre}_i(V))]\!]$ (see, e.g., [dAM04]). The first property is called a *reachability* property, the second a *safety* property.

3.3 Discounted Quantitative μ-calculus

A *discounted* version of the μ-calculus was introduced in [dAHM03]; we call this $d\mu$. We define it here for completeness. The discounted calculus $d\mu$ is derived from $q\mu$ by multiplying every application of the pre operator by a discount factor in the interval $[0, 1]$. If the discount factor of a pre operator is less than 1, then each additional application of the operator in a fixpoint iteration carries less weight.

Syntax. The syntax of the discounted quantitative μ-calculus is defined with respect to the set of observation variables $\mathcal{V}$ as well as a set *MVars* of *calculus variables,* which are distinct from the observation variables in $\mathcal{V}$. The syntax is given as follows:

$$\begin{aligned}\varphi \ ::= \ & c \mid v \mid V \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \varphi \oplus c \mid \varphi \ominus c \mid \\ & \alpha \cdot \mathrm{pre}_1(\varphi) \mid \alpha \cdot \mathrm{pre}_2(\varphi) \mid (1-\alpha) + \alpha \cdot \mathrm{pre}_1(\varphi) \mid (1-\alpha) + \alpha \cdot \mathrm{pre}_2(\varphi) \mid \\ & \mu V.\, \varphi \mid \nu V.\, \varphi\end{aligned}$$

for constants $c \in [0, 1]$, observation variables $v \in \mathcal{V}$, calculus variables $V \in \mathit{MVars}$, and parameters α from some fixed set Λ.

Semantics. A variable valuation $\xi : \mathit{MVars} \mapsto \mathcal{F}$ is a function that maps every variable $V \in \mathit{MVars}$ to a valuation in $\mathcal{F}$. We write $\xi[V \mapsto f]$ for the valuation that agrees with ξ on all variables, except that V is mapped to f. A parameter valuation $\mathcal{P} : \Lambda \mapsto [0, 1]$ is a function that maps every parameter $\alpha \in \Lambda$ to a real-valued discount factor in the interval $[0, 1]$. Given a real $r \in [0, 1]$, the parameter valuation $\mathcal{P}$ is *r-bounded* if $\mathcal{P}(\alpha) \leq r$ for all parameters $\alpha \in \Lambda$. Given a game structure G, a variable valuation ξ, and a parameter valuation $\mathcal{P}$, every formula φ of the discounted quantitative μ-calculus defines a valuation

$[\![\varphi]\!]^G_{\xi,\mathcal{P}} \in \mathcal{F}$ (the superscript G is omitted if the game structure is clear from the context):

$$[\![c]\!]_{\xi,\mathcal{P}} = \mathbf{c}$$

$$[\![v]\!]_{\xi,\mathcal{P}} = [v]$$

$$[\![V]\!]_{\xi,\mathcal{P}} = \xi(V)$$

$$[\![\neg\varphi]\!]_{\xi,\mathcal{P}} = \mathbf{1} - [\![\varphi]\!]_{\xi,\mathcal{P}}$$

$$[\![\varphi \{^{\oplus}_{\ominus}\} c]\!]_{\xi,\mathcal{P}} = [\![\varphi]\!]_{\xi,\mathcal{P}} \{^{\oplus}_{\ominus}\} \mathbf{c}$$

$$[\![\varphi_1 \{^{\vee}_{\wedge}\} \varphi_2]\!]_{\xi,\mathcal{P}} = [\![\varphi_1]\!]_{\xi,\mathcal{P}} \{^{\sqcup}_{\sqcap}\} [\![\varphi_2]\!]_{\xi,\mathcal{P}}$$

$$[\![\alpha \cdot \mathrm{pre}_i(\varphi)]\!]_{\xi,\mathcal{P}} = \mathcal{P}(\alpha) \cdot \mathrm{Pre}_i([\![\varphi]\!]_{\xi,\mathcal{P}})$$

$$[\![(1-\alpha) + \alpha \cdot \mathrm{pre}_i(\varphi)]\!]_{\xi,\mathcal{P}} = (1 - \mathcal{P}(\alpha)) + \mathcal{P}(\alpha) \cdot \mathrm{Pre}_i([\![\varphi]\!]_{\xi,\mathcal{P}})$$

$$[\![\{^{\mu}_{\nu}\} V.\, \varphi]\!]_{\xi,\mathcal{P}} = \{^{\inf}_{\sup}\} \{f \in \mathcal{F} \mid f = [\![\varphi]\!]_{\xi[V \mapsto f],\mathcal{P}}\}$$

where $i \in \{1, 2\}$. The existence of the fixpoints is guaranteed by the monotonicity and continuity of all operators and can be computed by Picard iteration [dAM04]. The region $[\![\varphi]\!]_{\xi,\mathcal{P}}$ is in general not boolean even if the game structure is turn-based deterministic, because the discount factors introduce real numbers. The discounted μ-calculus is closed under negation: if we define the negation of a formula φ inductively using $\neg(\alpha \cdot \mathrm{pre}_1(\varphi')) = (1-\alpha) + \alpha \cdot \mathrm{pre}_2(\neg\varphi')$ and $\neg((1-\alpha) + \alpha \cdot \mathrm{pre}_1(\varphi')) = \alpha \cdot \mathrm{pre}_2(\neg\varphi')$, then $[\![\neg\varphi]\!]_{\xi,\mathcal{P}} = \mathbf{1} - [\![\varphi]\!]_{\xi,\mathcal{P}}$. This generalizes the duality $\mathbf{1} - \mathrm{pre}_1(f) = \mathrm{pre}_2(\mathbf{1} - f)$ of the undiscounted pre operators.

Chapter 4

Relations and Metrics

In this chapter we present metrics and relations in stochastic games and their logical characterization. We are interested in developing a *metric* on states of a game structure that captures an approximate notion of equivalence: states close in the metric should yield similar values to the players for any winning objective. Specifically, we are interested in defining 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:

$$[\simeq_g](s,t) = \sup_{\varphi \in q\mu} |[\![\varphi]\!](s) - [\![\varphi]\!](t)| \,. \tag{4.1}$$

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. Notice that in defining the metric independent of a player, we are expecting our metrics to be *reciprocal*, that is, invariant under a change of player. 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. To underline the connection between classical equivalences and the metrics we develop, we write $[s \simeq_g t]$ for $[\simeq_g](s,t)$, so that the desired property of the bisimulation metric can be stated as

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

Metrics of this type have already been developed for Markov decision processes (MDPs) [vBW01a, DGJP03]. Our construction of metrics for games starts from an analysis of these constructions.

4.1 Metrics for MDPs

We consider the case of 1-MDPs; the case for 2-MDPs is symmetrical. Throughout this subsection, we fix a 1-MDP $\langle S, [\cdot], \mathit{Moves}, \Gamma_1, \Gamma_2, \delta \rangle$. Before we present the metric correspondent of probabilistic simulation, we first rephrase classical probabilistic (bi)simulation on MDPs [LS92, JS90, SL94, SL95] 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 $p, q \in \mathrm{Dist}(S)$, we let $p \sqsubseteq_R q$ if there is a function $\Delta : S \times S \to [0,1]$ 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

$$s \equiv t \ \wedge \ \forall x_1 \in \mathcal{D}_1(s) \,.\, \exists y_1 \in \mathcal{D}_1(t) \,.\, \delta(s,x_1) \sqsubseteq_R \delta(t,y_1), \tag{4.2}$$

for all states $s, t \in S$. Probabilistic simulation is the greatest fixpoint of (4.2); probabilistic bisimulation is the greatest symmetrical fixpoint of (4.2).

To obtain a metric equivalent of probabilistic simulation, we lift the above fixpoint from relations (subsets of S^2) to metrics (maps $S^2 \mapsto \mathbb{R}$). We lift (4.2) to metrics, defining a metric transformer $H^{1MDP}_{post} : \mathcal{M} \mapsto \mathcal{M}$. 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. We will show later how to define such a distribution distance. For $s, t \in S$, we let

$$H^{1MDP}_{post}(d)(s,t) = p(s,t) \sqcup \sup_{x_1 \in \mathcal{D}_1(s)} \inf_{y_1 \in \mathcal{D}_1(t)} D(\delta(s,x_1), \delta(t,y_1))(d) \,. \tag{4.3}$$

In this definition, the $\forall$ and $\exists$ of (4.2) have been replaced by sup and inf, respectively. Since equivalent states should have distance 0, the simulation metric in MDPs is defined as the *least* (rather than greatest) fixpoint of (4.3) [vBW01a, DGJP03]. Similarly, the bisimulation metric is defined as the least symmetrical fixpoint of (4.3).

For a distance $d \in \mathcal{M}$ and two distributions $p, q \in \mathrm{Dist}(S)$, 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 $d(s,t)$. More precisely, $D(p,q)(d)$ is defined via the *trans-shipping problem,* as the minimum cost of shipping the distribution p into q, with edge costs d. Thus, $D(p,q)(d)$ is the solution of the following

linear programming (LP) problem over the set of variables $\{\lambda_{s,t}\}_{s,t\in S}$:

$$\text{Minimize} \quad \sum_{s,t\in S} d(s,t)\lambda_{s,t}$$

$$\text{subject to} \sum_{t\in S} \lambda_{s,t} = p(s), \quad \sum_{s\in S} \lambda_{s,t} = q(t), \quad \lambda_{s,t} \geq 0 \,.$$

Equivalently, we can define $D(p,q)(d)$ via the dual of the above LP problem [vBW01a]. Given a metric $d \in \mathcal{M}$, let $C(d) \subseteq \mathcal{F}$ be the subset of valuations $k \in \mathcal{F}$ such that $k(s) - k(t) \leq d(s,t)$ for all $s,t \in S$. Then the dual formulation is:

$$\text{Maximize} \quad \sum_{s\in S} p(s)\, k(s) - \sum_{s\in S} q(s)k(s) \tag{4.4}$$

$$\text{subject to } k \in C(d) \,.$$

The constraint $C(d)$ on the valuation k, states that the value of k across states cannot differ by more than d. This means, intuitively, that k behaves like the valuation of a $q\mu$ formula: as we will see, the logical characterization implies that d is a bound for the difference in valuation of $q\mu$ formulas across states. Indeed, the logical characterization of the metrics is proved by constructing formulas whose valuation approximate that of the optimal k. Plugging (4.4) into (4.3), we obtain:

$$H^{1MDP}_{post}(d)(s,t) = p(s,t) \sqcup \sup_{x_1\in\mathcal{D}_1(s)} \inf_{y_1\in\mathcal{D}_1(t)} \sup_{k\in C(d)} \left(\mathbb{E}^{x_1}_s(k) - \mathbb{E}^{y_1}_t(k)\right) . \tag{4.5}$$

We can interpret this definition as follows. State t is trying to simulate state s (this is a definition of a simulation metric). First, state s chooses a mixed move x_1, attempting to make simulation as hard as possible; then, state t chooses a mixed move y_1, trying to match the effect of x_1. Once x_1 and y_1 have been chosen, the resulting distance between s and t is equal to the maximal difference in expectation, for moves x_1 and y_1, of a valuation

$k \in C(d)$. We call the metric transformer H^{1MDP}_{post} the *a posteriori* metric transformer: the valuation k in (4.5) 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:

$$H^{1MDP}_{prio}(d)(s,t) = p(s,t) \sqcup \sup_{k \in C(d)} \sup_{x_1 \in \mathcal{D}_1(s)} \inf_{y_1 \in \mathcal{D}_1(t)} \left(\mathbb{E}^{x_1}_s(k) - \mathbb{E}^{y_1}_t(k) \right) . \tag{4.6}$$

Intuitively, in the a priori transformer, first a valuation $k \in C(d)$ is chosen. Then, state t must simulate state s with respect to the expectation of k. 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 2 below states that for MDPs, a priori and a posteriori simulation metrics coincide. In the next section, we will see that this is not the case for games.

Theorem 2 *For all MDPs,* $H^{1MDP}_{post} = H^{1MDP}_{prio}$.

Proof. Consider two states $s, t \in S$, and a metric $d \in \mathcal{M}$. We have to prove that

$$\sup_k \sup_{x_1} \inf_{y_1} [\mathbb{E}^{x_1}_s(k) - \mathbb{E}^{y_1}_t(k)] = \sup_{x_1} \inf_{y_1} \sup_k [\mathbb{E}^{x_1}_s(k) - \mathbb{E}^{y_1}_t(k)] . \tag{4.7}$$

In the left-hand side, we can exchange the two outer sups. Then, noticing that the difference in expectation is bi-linear in k and y_1 for a fixed x_1, that y_1 is a probability distribution, and that k is chosen from a compact convex subset, we apply the generalized minimax theorem [Sio58] to exchange $\sup_k \inf_{y_1}$ into $\inf_{y_1} \sup_k$, thus obtaining the right-hand side. ■

The metrics defined above are logically characterized by $q\mu$. Precisely, let $[\sim] \in \mathcal{M}$ be the least symmetrical fixpoint of $H^{1MDP}_{prio} = H^{1MDP}_{post}$. Then, Lemma 5.24 and Corollary 5.25 of [DGJP03], (originally stated for H^{1MDP}_{post}) state that for all states s, t of a 1-MDP, we have

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

4.2 Metrics for Concurrent Games

We now extend the simulation and bisimulation metrics from MDPs to general game structures. As we shall see, unlike for MDPs, the a priori and the a posteriori metrics do not coincide over games. In particular, we show that the a priori formulation satisfies both a tight logical characterization as well as reciprocity while, perhaps surprisingly, the more natural a posteriori version does not.

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 $s, t \in S$:

$$\begin{aligned} H_{\sqsubseteq_1}(d)(s,t) &= p(s,t) \sqcup \sup_{x_1 \in \mathcal{D}_1(s)} \inf_{y_1 \in \mathcal{D}_1(t)} \sup_{y_2 \in \mathcal{D}_2(t)} \inf_{x_2 \in \mathcal{D}_2(s)} D(\delta(s, x_1, x_2), \delta(t, y_1, y_2), d) \\ &= p(s,t) \sqcup \sup_{x_1 \in \mathcal{D}_1(s)} \inf_{y_1 \in \mathcal{D}_1(t)} \sup_{y_2 \in \mathcal{D}_2(t)} \inf_{x_2 \in \mathcal{D}_2(s)} \sup_{k \in C(d)} \left(\mathbb{E}^{x_1,x_2}_s(k) - \mathbb{E}^{y_1,y_2}_t(k)\right) . \end{aligned} \tag{4.8}$$

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 $s, t \in S$:

$$H_{\preceq_1}(d)(s,t) = p(s,t) \sqcup \sup_{k \in C(d)} \sup_{x_1 \in \mathcal{D}_1(s)} \inf_{y_1 \in \mathcal{D}_1(t)} \sup_{y_2 \in \mathcal{D}_2(t)} \inf_{x_2 \in \mathcal{D}_2(s)} \left(\mathbb{E}_s^{x_1,x_2}(k) - \mathbb{E}_t^{y_1,y_2}(k) \right)$$

$$= p(s,t) \sqcup \sup_{k \in C(d)} \left[\sup_{x_1 \in \mathcal{D}_1(s)} \inf_{x_2 \in \mathcal{D}_2(s)} \mathbb{E}_s^{x_1,x_2}(k) - \sup_{y_1 \in \mathcal{D}_1(t)} \inf_{y_2 \in \mathcal{D}_2(t)} \mathbb{E}_t^{y_1,y_2}(k) \right]$$

$$= p(s,t) \sqcup \sup_{k \in C(d)} \left(\mathrm{Pre}_1(k)(s) - \mathrm{Pre}_1(k)(t) \right) . \tag{4.9}$$

First, we show that $H_{\preceq_1}$ and $H_{\sqsubseteq_1}$ are monotonic in the lattice of metrics $(\mathcal{M}, \leq)$.

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

Proof. For $d, d' \in \mathcal{M}$, $d \leq d'$ implies $C(d) \subseteq C(d')$, and hence $\sup_{k \in C(d)} (\mathrm{Pre}_1(k)(s) - \mathrm{Pre}_1(k)(t)) \leq \sup_{k \in C(d')} (\mathrm{Pre}_1(k)(s) - \mathrm{Pre}_1(k)(t))$. This shows the monotonicity of $H_{\preceq_1}$.

The monotonicity of $H_{\sqsubseteq_1}$ can be shown in a similar fashion. From $d \leq d'$, reasoning as before we obtain

$$\sup_{k \in C(d)} \left(\mathbb{E}_s^{x_1,x_2}(k) - \mathbb{E}_t^{y_1,y_2}(k) \right) \leq \sup_{k \in C(d')} \left(\mathbb{E}_s^{x_1,x_2}(k) - \mathbb{E}_t^{y_1,y_2}(k) \right) .$$

The result then follows from the monotonicity of the operators $\sup_{x_1 \in \mathcal{D}_1(s)}$, $\inf_{y_1 \in \mathcal{D}_1(t)}$, $\sup_{y_2 \in \mathcal{D}_2(t)}$, $\inf_{x_2 \in \mathcal{D}_2(s)}$. ■

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 [dAHM03, Maj03]. 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

$$H_{\cong_1}(d) = H_{\sqsubseteq_1}(d) \sqcup Opp(H_{\sqsubseteq_1}(d)), \tag{4.10}$$

where $Opp(d) = \breve{d}$ 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

$$H_{\simeq_1}(d) = H_{\preceq_1}(d) \sqcup Opp(H_{\preceq_1}(d)) \ . \tag{4.11}$$

We wish to show that the metrics of Definition 1 can be computed via Picard iteration. To this end, it is necessary to show that the operators $H_{\sqsubseteq_1}$ and $H_{\preceq_1}$ on the lattice $(\mathcal{M}, \leq)$ are upper semi-continuous. In fact, a very similar proof shows that the operators are lower semi-continuous, and thus, continuous; we omit the proof of this more general fact as it is not required for the desired result about the applicability of Picard iteration.

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

Proof. Let $D \subseteq \mathcal{M}$ be an arbitrary set of distances, and let $d^* = \sup D$; note that d^* exists, as $(\mathcal{M}, \leq)$ is a complete lattice.

We first prove the result for $H_{\preceq_1}$. We need to prove that $H_{\preceq_1}(\sup D) = \sup_{d \in D} H_{\preceq_1}(d)$, which we abbreviate $H_{\preceq_1}(\sup D) = \sup H_{\preceq_1}(D)$. In one direction, $H_{\preceq_1}(\sup D) \geq \sup H_{\preceq_1}(D)$ follows from the monotonicity of $H_{\preceq_1}$ (Lemma 1). In the other direction, we will show that for all $\epsilon > 0$, there is $d \in D$ such that $|H_{\preceq_1}(d^*) - H_{\preceq_1}(d)| \leq \epsilon$, where for $d, d' \in \mathcal{M}$, $|d - d'|$ is the 1-norm distance between d and d'. For convenience, let $G(k) \in \mathcal{M}$ be defined as $G(k)(s,t) = \mathrm{Pre}_1(k)(s) - \mathrm{Pre}_1(k)(t)$, so that we can write $H_{\preceq_1}(d) = p(s,t) \sqcup \sup_{k \in C(d)} G(k)$.

Given $\epsilon > 0$, choose $d \in D$ such that for all $s, t \in S$, we have $d(s,t)/d^*(s,t) \geq 1 - \epsilon/4$ if $d^*(s,t) > 0$, and $d(s,t) = 0$ if $d^*(s,t) = 0$. Note that for all $k \in C(d^*)$, we have $(1 - \epsilon/4)k \in C(d)$ and $|k - (1 - \epsilon/4)k| \leq \epsilon/4$, as $|k| \leq 1$. Thus, $d \in D$ is such that for all $k \in C(d^*)$, there is $k' \in C(d)$ with $|k - k'| \leq \epsilon/4$. In other words, d is such that the Hausdorff distance between $C(d^*)$ and $C(d)$ is at most $\epsilon/4$. We now prove that for this d, we have

$$|\sup_{k \in C(d^*)} G(k) - \sup_{k \in C(d)} G(k)| \leq \epsilon \,. \tag{4.12}$$

In fact, let $k^* \in C(d^*)$ be such that

$$|G(k^*) - \sup_{k \in C(d^*)} G(k)| \leq \epsilon/2 \,. \tag{4.13}$$

and let $k' \in C(d)$ be such that $|k^* - k'| \leq \epsilon/4$. For $s, t \in S$, we have by definition $G(k^*)(s,t) = \mathrm{Pre}_1(k^*)(s) - \mathrm{Pre}_1(k^*)(t)$; let

$$x_1(s) = \arg \sup_{x \in \mathcal{D}_1(s)} \inf_{y \in \mathcal{D}_2(s)} \mathbb{E}_s^{x,y}(k^*) \,.$$

By employing $x_1(s)$ at all $s \in S$, player 1 can guarantee

$$|G(k')(s,t) - G(k^*)(s,t)| \leq \epsilon/2,$$

which together with (4.13) leads to (4.12). In turn, (4.12) yields the result.

We can prove the result for $H_{\sqsubseteq_1}$ following a similar argument. Precisely, in one direction, $H_{\sqsubseteq_1}(\sup D) \geq \sup H_{\sqsubseteq_1}(D)$ follows from the monotonicity of $H_{\sqsubseteq_1}$ (Lemma 1). In the other direction, we will show that for all $\epsilon > 0$, there is $d \in D$ such that $|H_{\sqsubseteq_1}(d^*) - H_{\sqsubseteq_1}(d)| \leq \epsilon$, where for $d, d' \in \mathcal{M}$, $|d - d'|$ is the 1-norm distance between d and d'. Again, let d be such that the Hausdorff distance between $C(d^*)$ and $C(d)$ is at most $\epsilon/2$. For such a d, we have that for all $s, t \in S$, and $x_1 \in \mathcal{D}_1(s)$, $y_1 \in \mathcal{D}_1(t)$, $x_2 \in \mathcal{D}_2(s)$, $y_2 \in \mathcal{D}_2(t)$,

$$\Big| \sup_{k \in C(d^*)} \left(\mathbb{E}_s^{x_1,x_2}(k) - \mathbb{E}_t^{y_1,y_2}(k)\right) - \sup_{k \in C(d)} \left(\mathbb{E}_s^{x_1,x_2}(k) - \mathbb{E}_t^{y_1,y_2}(k)\right) \Big| \leq \epsilon,$$

and this leads easily to the result. ■

This result implies that we can compute $[\preceq_1]$ as the fixpoint of $H_{\preceq_1}$ via Picard iteration; we denote by $d_n = H^n_{\preceq_1}(\mathbf{0})$ the n-iterate of this. Similarly, we can compute $[\sqsubseteq_1]$ as the fixpoint of $H_{\sqsubseteq_1}$ via Picard iteration.

Theorem 3 *The following assertions hold, for $i \in \{1, 2\}$:*

1. *Let $d_0 = d'_0 = \mathbf{0}$, and for $n \geq 0$, let*

$$d_{n+1} = H_{\preceq_i}(d_n) \quad \textit{and} \quad d'_{n+1} = H_{\sqsubseteq_i}(d'_n)\ . \tag{4.14}$$

   We have $\lim_{n\to\infty} d_n = [\preceq_i]$ and $\lim_{n\to\infty} d'_n = [\sqsubseteq_i]$.

2. *Let $b_0 = b'_0 = \mathbf{0}$, and for $n \geq 0$, let*

$$b_{n+1} = H_{\preceq_i}(b_n) \sqcup Opp(H_{\preceq_i}(b_n)) \ \textit{ and } \ b'_{n+1} = H_{\sqsubseteq_i}(b'_n) \sqcup Opp(H_{\sqsubseteq_i}(b'_n))\ . \tag{4.15}$$

We have $\lim_{n\to\infty} b_n = [\simeq_i]$ *and* $\lim_{n\to\infty} b'_n = [\cong_i]$.

Proof. The statements follow from the definitions of the metrics, and from Lemmas 1 and 2.

■

We now show some basic properties of these metrics. First, we show that the a priori fixpoints give a (directed) metric, i.e., they are non-negative and satisfy the triangle inequality. We also prove that the a priori and a posteriori metrics are distinct. We then focus on the a priori metrics, and show, through our results, that they are the natural metrics for concurrent games.

Theorem 4 *For all game structures G, and all states s, t, u of G, we have,*

1. $[s \preceq_1 t] \geq 0$ *and* $[s \preceq_1 u] \leq [s \preceq_1 t] + [t \preceq_1 u]$.

2. $[s \sqsubseteq_1 t] \geq 0$ *and* $[s \sqsubseteq_1 u] \leq [s \sqsubseteq_1 t] + [t \sqsubseteq_1 u]$.

Proof. We prove the following statement: if $d \in \mathcal{M}$ is a directed metric, then:

1. $H_{\preceq_1}(d)$ is a directed metric;

2. $H_{\sqsubseteq_1}(d)$ is a directed metric.

The theorem then follows by induction on the Picard iteration with which the a priori and a posteriori metrics can be computed (Theorem 3). We prove the result first for the a priori metric.

First, from $d' = H_{\preceq_1}(d)$ and $p(s,t) \geq 0$ for all $s, t \in S$, we immediately have $d' \geq 0$ (where inequalities are interpreted in pointwise fashion).

To prove the triangle inequality, we observe that $p(s,t) + p(t,u) \geq p(s,u)$ for all $s,t,u \in S$. Also,

$$\begin{aligned}
&\sup_{k \in C(d)} \big(\mathrm{Pre}_1(k)(s) - \mathrm{Pre}_1(k)(t)\big) + \sup_{k \in C(d)} \big(\mathrm{Pre}_1(k)(t) - \mathrm{Pre}_1(k)(u)\big) \\
&\qquad \geq \sup_{k \in C(d)} \big(\mathrm{Pre}_1(k)(s) - \mathrm{Pre}_1(k)(t) + \mathrm{Pre}_1(k)(t) - \mathrm{Pre}_1(k)(u)\big) \\
&\qquad = \sup_{k \in C(d)} \big(\mathrm{Pre}_1(k)(s) - \mathrm{Pre}_1(k)(u)\big) \,.
\end{aligned}$$

Thus, we obtain

$$\begin{aligned}
&H_{\preceq_1}(d)(s,t) + H_{\preceq_1}(d)(t,u) \\
&= \big(p(s,t) \sqcup \sup_{k \in C(d)} \big(\mathrm{Pre}_1(k)(s) - \mathrm{Pre}_1(k)(t)\big)\big) + \big(p(t,u) \sqcup \sup_{k \in C(d)} \big(\mathrm{Pre}_1(k)(t) - \mathrm{Pre}_1(k)(u)\big)\big) \\
&\qquad \geq \big(p(s,u) \sqcup \sup_{k \in C(d)} \big(\mathrm{Pre}_1(k)(s) - \mathrm{Pre}_1(k)(u)\big)\big) = H_{\preceq_1}(d)(s,u),
\end{aligned}$$

leading to the result.

For the a posteriori metric, let $d' = H_{\sqsubseteq_1}(d)$; again, we can prove $d' \geq 0$ as in the a priori case. To prove the triangle inequality for d', for $s,t \in S$, and for distributions $x_1 \in \mathcal{D}_1(s)$ and $y_1 \in \mathcal{D}_1(t)$, it is convenient to let

$$G(x_1,y_1)(s,t) = \sup_{y_2 \in \mathcal{D}_2(t)} \inf_{x_2 \in \mathcal{D}_2(s)} \sup_{k \in C(d)} \big(\mathbb{E}_s^{x_1,x_2}(k) - \mathbb{E}_t^{y_1,y_2}(k)\big),$$

With this notation, for $s,t,u \in S$, we have

$$H_{\sqsubseteq_1}(d)(s,u) = p(s,u) \sqcup \sup_{x_1 \in \mathcal{D}_1(s)} \inf_{z_1 \in \mathcal{D}_1(u)} G(x_1,z_1)(s,u) \,. \tag{4.16}$$

Intuitively, the quantity $G(x_1,z_1)(s,u)$ is the distance between s and u computed in the 2-MDP obtained when player 1 plays x_1 at s and z_1 at u. As a consequence of Theorem 2

(interpreted over 2-MDPs), and of the previous proof for the a-priori case, we have that

$$G(x_1, z_1)(s, u) \leq G(x_1, y_1)(s, t) + G(y_1, z_1)(t, u) \ . \tag{4.17}$$

for all $x_1 \in \mathcal{D}_1(s)$, $y_1 \in \mathcal{D}_1(t)$, and $z_1 \in \mathcal{D}_1(u)$. This observation will be useful in the following.

For any $\epsilon > 0$, let x_1^* realize the sup in (4.16) within ϵ, that is,

$$\inf_{z_1 \in \mathcal{D}_1(u)} G(x_1^*, z_1)(s, u) \geq \sup_{x_1 \in \mathcal{D}_1(s)} \inf_{z_1 \in \mathcal{D}_1(u)} G(x_1, z_1)(s, u) - \epsilon, \tag{4.18}$$

and let z_1^* realize the inf of the left-hand side of (4.18) also within ϵ. Intuitively, x_1^* is the player-1 distribution at s that is hardest to imitate from u, and z_1^* is the best imitation of x_1^* available at u. In the same fashion, let y_1^* realize the inf within ϵ in $\inf_{y_1 \in \mathcal{D}_1(t)} G(x_1^*, y_1)(s, t)$, and let z_1' realize the inf within ϵ in $\inf_{z_1 \in \mathcal{D}_1(u)} G(y_1^*, z_1)(t, u)$. In intuitive terms, y_1^* is the imitator of x_1^* in t, and z_1' is the imitator of y_1^* in u.

We consider two cases. If $p(s, u) = 1$, then we are sure that the triangle inequality

$$d'(s, u) \leq d'(s, t) + d'(t, u), \tag{4.19}$$

holds. Otherwise, note that

$$d'(s, u) \leq G(x_1^*, z_1^*)(s, u) + 2\epsilon \ . \tag{4.20}$$

Since x_1^* is not necessarily the distribution at s that is hardest to imitate from t, and since y_1^* is not necessarily the distribution at t that is hardest to imitate from u, we also have:

$$d'(s, t) \geq G(x_1^*, y_1^*)(s, t) - \epsilon \qquad d'(t, u) \geq G(y_1^*, z_1')(t, u) - \epsilon \ . \tag{4.21}$$

Since the triangle inequality holds for MDPs, as stated by (4.17), we have

$$G(x_1^*, z_1')(s, u) \quad \leq \quad G(x_1^*, y_1^*)(s, t) + G(y_1^*, z_1')(t, u) \quad \leq \quad d'(s, t) + d'(t, u) + 2\epsilon \ . \tag{4.22}$$

Since z_1^* is the best imitator of x_1^* at u, we also have

$$G(x_1^*, z_1^*)(s,u) - \epsilon \leq G(x_1^*, z_1')(s,u), \tag{4.23}$$

which together with (4.22) yields

$$G(x_1^*, z_1^*)(s,u) \leq d'(s,t) + d'(t,u) + 3\epsilon\ . \tag{4.24}$$

From the choice of x_1^*, this finally leads to

$$d'(s,u) \leq d'(s,t) + d'(t,u) + 5\epsilon,$$

for all $\epsilon > 0$, which yields the desired triangle inequality (4.19). ■

4.2.1 *A priori* and *a posteriori* Metrics are Distinct.

First, we show 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 (4.8), (4.9). It is easier to simulate a state s from a state t if the valuation is known in advance, as in a priori metrics (4.9), than if the valuation k is chosen after all the moves have been chosen, as in a posteriori metrics (4.8).

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 5 *The following assertions hold.*

1. *For all game structures G, and for all states s, t of G, we have $[s \preceq_1 t] \leq [s \sqsubseteq_1 t]$.*

$\delta(t,*,*)(w)$	f	g
b	$1/9$	$5/9$
c	$4/9$	$8/9$

$\delta(t,*,*)(u)$	f	g
b	$8/9$	$4/9$
c	$5/9$	$1/9$

$\delta(s,*,*)(w)$	f	g
a	$1/3$	$2/3$

$\delta(s,*,*)(u)$	f	g
a	$2/3$	$1/3$

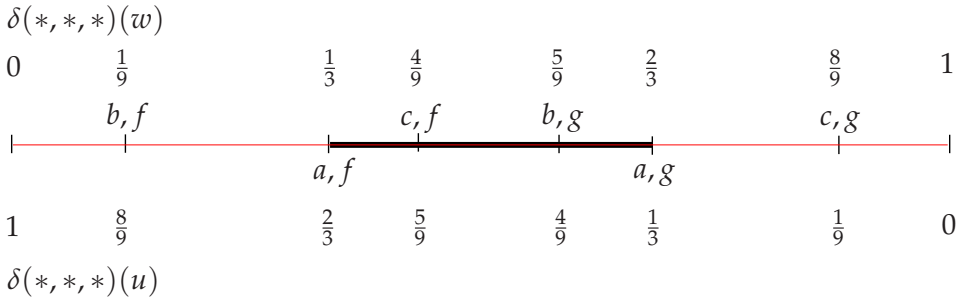


Figure 4.1: A game that shows that the a priori and the a posteriori metrics may not coincide. The tables above show the transition probabilities from states t and s to states w and u for pure moves of the two players. The row player is player 1 and the column player is player 2. The line below is the two dimensional probability simplex that shows the transition probabilities induced by convex combinations of pure moves of the two players.

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 $[\preceq_1] = [\sqsubseteq_1]$.*

Proof. The first assertion is a consequence of the fact that, for all functions $f : \mathbb{R}^2 \mapsto \mathbb{R}$, we have $\sup_x \inf_y f(x,y) \leq \inf_y \sup_x f(x,y)$. By repeated applications of this, we can show that, for all $d \in \mathcal{M}$, we have $H_{\preceq}(d) \leq H_{\sqsubseteq}(d)$ (with pointwise ordering). The result then follows from the monotonicity of $H_{\preceq}$ and $H_{\sqsubseteq}$.

For the second assertion, we give an example where a priori distances are strictly less than a posteriori distances. Consider a game with states $S = \{s, t, u, w\}$. States u and

w are sink states with $p(u,w) = 1$; states s and t are such that $p(s,t) = 0$. At states s and t, player 2 has moves $\{f,g\}$. Player-1 has a single move $\{a\}$ at state s, and moves $\{b,c\}$ at state t. The moves from s and t lead to u and w with transition probabilities indicated in Figure 4.1. In the figure, the point b,f indicates the probability of going to u and w when the move pair (b,f) is played, with $\delta(s,b,f)(u) + \delta(s,b,f)(w) = 1$; similarly for the other move pairs. The thick line segment between the points a,f and a,g represents the transition probabilities arising when player 1 plays move a, and player 2 plays a mixed move (a mix of f and g).

We show that, in this game, we have $[s \sqsubseteq_1 t] > 0$. Consider the metric d where $d(u,w) = 1$ (recall that $p(u,w) = 1$, and note the other distances do not matter, since u, w are the only two destinations). We need to show

$$\forall y_1 \in \mathcal{D}_1(t).\exists y_2 \in \mathcal{D}_2(t).\forall x_2 \in \mathcal{D}_2(s).\exists k \in C(d).\big(\mathbb{E}_s^{a,x_2}(k) - \mathbb{E}_t^{y_1,y_2}(k)\big) > 0\,. \tag{4.25}$$

Consider any mixed move $y_1 = \alpha b + (1-\alpha)c$, where b,c are the moves available to player 1 at t, and $0 \leq \alpha \leq 1$. If $\alpha \geq \frac{1}{2}$, choose move f from t as y_2, and choose $k(w) = 1$, $k(u) = 0$. Otherwise, choose move g from t as y_2, and choose $k(w) = 0$, $k(u) = 1$. With these choices, the transition probability $\delta(t,y_1,y_2)$ will fall outside of the segment $[(a,f),(a,g)]$ in Figure 4.1. Thus, with the choice of k above, we ensure that the difference in (4.25) is always positive.

To show that in the game we have $[s \preceq_1 t] = 0$, it suffices to show (given that $[s \preceq_1 t] \geq 0$) that

$$\forall k \in C(d).\exists y_1 \in \mathcal{D}_1(t).\forall y_2 \in \mathcal{D}_2(t).\exists x_2 \in \mathcal{D}_2(s).\big(\mathbb{E}_s^{a,x_2}(k) - \mathbb{E}_t^{y_1,y_2}(k)\big) \leq 0\,.$$

If $k(u) = k(w)$, the result is immediate. Assume otherwise, that $k(u) < k(w)$, and choose $y_1 = c$. For every y_2, the distribution over successor states (and of k-expectations) will be in the interval $[(c, f), (c, g)]$ in Figure 4.1. By choosing $x_2 = f$, we have that $\mathbb{E}_s^{a,f}(k) < \mathbb{E}_t^{c,y_2}(k)$ for all $y_2 \in \mathcal{D}_2(t)$, leading to the result. Similarly if $k(u) > k(w)$, by choosing $y_1 = b$, the distribution over successor states (and of k-expectations) will now be in the interval $[(b, f), (b, g)]$. By choosing $x_2 = g$, we have that $\mathbb{E}_s^{a,g}(k) < \mathbb{E}_t^{b,y_2}(k)$ for all $y_2 \in \mathcal{D}_2(t)$, again leading to the result.

The last assertion of the theorem is proved in the same way as Theorem 2. ■

4.3 Reciprocity of *a priori* Metric

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

Theorem 6 *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 $[\cong_1] \neq [\cong_2]$.*

Proof. For the first assertion, it suffices to show that, for all $d \in \mathcal{M}$, and states $s, t \in S$, we have $H_{\preceq_1}(d)(s,t) = H_{\preceq_2}(\check{d})(t,s)$. We proceed as follows:

$$\sup_{k \in C(d)} \big(\mathrm{Pre}_1(k)(s) - \mathrm{Pre}_1(k)(t)\big) \tag{4.26}$$

$$= \sup_{k \in C(d)} \big(-\mathrm{Pre}_2(1-k)(s) + \mathrm{Pre}_2(1-k)(t)\big) \tag{4.27}$$

$$= \sup_{k \in C(\check{d})} \big(\mathrm{Pre}_2(k)(t) - \mathrm{Pre}_2(k)(s)\big)\ . \tag{4.28}$$

The step from (4.26) to (4.27) uses $\mathrm{Pre}_1(k)(s) = 1 - \mathrm{Pre}_2(1-k)(s)$ [vNM44, dAM04], and the step from (4.27) to (4.28) uses the change of variables $k \to 1-k$.

For the second assertion, consider again the game of Figure 4.1. We will show that $[t \sqsubseteq_2 s] = 0$. Together with $[s \sqsubseteq_1 t] > 0$, as shown in the proof of Theorem 5, this leads to the result. To obtain the result, we will prove that for all d, we have:

$$\forall y_2 \in \mathcal{D}_2(t).\exists x_2 \in \mathcal{D}_2(s).\exists y_1 \in \mathcal{D}_1(t).\forall k \in C(d).\big(\mathbb{E}_t^{y_2,y_1}(k) - \mathbb{E}_s^{x_2,a}(k)\big) = 0\ .$$

where we have used the fact that player 1 at s plays $x_1 = a$. Any mixed move $y_2 \in \mathcal{D}_2(t)$ can be written as $y_2 = \alpha f + (1-\alpha)g$ for $0 \le \alpha \le 1$. Choose $y_1 = \alpha c + (1-\alpha)b$, and

$$x_2 = \alpha\Big(\frac{2}{3}f + \frac{1}{3}g\Big) + (1-\alpha)\Big(\frac{1}{3}f + \frac{2}{3}g\Big)\ .$$

Under this choice of mixed moves, we have:

$$\delta(t, y_1, y_2)(w) = \frac{4}{9}\alpha^2 + \alpha(1-\alpha) + \frac{5}{9}(1-\alpha)^2 = \frac{5}{9} - \frac{1}{9}\alpha$$

$$\delta(s, x_1, x_2)(w) = \alpha\Big(\frac{2}{3}\cdot\frac{1}{3} + \frac{1}{3}\cdot\frac{2}{3}\Big) + (1-\alpha)\Big(\frac{2}{3}\cdot\frac{2}{3} + \frac{1}{3}\cdot\frac{1}{3}\Big) = \frac{5}{9} - \frac{1}{9}\alpha\ .$$

As the probabilities of transitions to w are equal from t and s, we obtain that for all $k \in C(d)$, we have $\mathbb{E}_t^{y_2,y_1}(k) - \mathbb{E}_s^{x_2,a}(k) = 0$, as desired.

For the third assertion, we consider a modified version of the game depicted in Figure 4.1, obtained by adding two new moves to player 2 at state t, namely f' and g'. We define the transition probabilities of these new moves by

$$\delta(t, *, f') = \delta(s, a, f) \qquad \delta(t, *, g') = \delta(s, a, g) \ .$$

To prove $[s \sqsubseteq_1 t] > 0$, we can proceed as in the proof of Theorem 5, noting that we can choose y_2 as in that proof (this is possible, as player 2 at t has *more* moves available in the modified game). This leads to $[s \cong_1 t] > 0$.

To show that $[s \cong_2 t] = 0$, given the transition structure of the game, it suffices to show that $[s \sqsubseteq_2 t] = 0$ and $[t \sqsubseteq_2 s] = 0$. To show that $[s \sqsubseteq_2 t] = 0$, we show that for all d, we have:

$$\forall x_2 \in \mathcal{D}_2(s).\exists y_2 \in \mathcal{D}_2(t).\forall y_1 \in \mathcal{D}_1(t).\forall k \in C(d).\left(\mathbb{E}_s^{x_2,a}(k) - \mathbb{E}_t^{y_2,y_1}(k)\right) = 0 \ .$$

We can write any mixed move $x_2 \in \mathcal{D}_2(s)$ as $x_2 = \alpha f + (1-\alpha)g$. We can then choose $y_2 = \alpha f' + (1-\alpha)g'$, and since at t under f', g' the transition probabilities do not depend on the mixed move y_1 chosen by player 1, we have that the transition probabilities from s and t match for all $0 \leq \alpha \leq 1$.

To show that $[t \sqsubseteq_2 s] = 0$, we need to show that:

$$\forall y_2 \in \mathcal{D}_2(t).\exists x_2 \in \mathcal{D}_2(s).\exists y_1 \in \mathcal{D}_1(t).\forall k \in C(d).\left(\mathbb{E}_t^{y_2,y_1}(k) - \mathbb{E}_s^{x_2,a}(k)\right) = 0 \ .$$

Any mixed move $y_2 \in \mathcal{D}_2(t)$ can be written as

$$y_2 = \gamma\Big[\alpha f + (1-\alpha)g\Big] + (1-\gamma)\Big[\beta f' + (1-\beta)g'\Big],$$

for some $\alpha, \beta, \gamma \in [0,1]$. We choose x_2 and y_1 as follows:

$$x_2 = \alpha\gamma\Big[\frac{2}{3}f + \frac{1}{3}g\Big] + (1-\alpha)\gamma\Big[\frac{1}{3}f + \frac{2}{3}g\Big] + (1-\gamma)\Big[\beta f + (1-\beta)g\Big]$$

$$y_1 = \alpha c + (1-\alpha)b\ .$$

With these mixed moves, we have $\delta(s,a,x_2) = \delta(t,y_1,y_2)$, leading to the result. ■

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.

4.4 Logical Characterization of *a priori* Metric

We now prove that $q\mu$ provides a logical characterization for the a priori metrics. We first state and prove two lemmas that lead to the desired result. The proof of the lemmas use ideas from [Maj03] and [DGJP03]. We recall from Theorem 3 that we can compute $[\preceq_1]$ via Picard iteration, with $d_n = H^n_{\preceq_1}(\mathbf{0})$ being the n-iterate.

We prove the existence of a logical characterization via a sequence of the following two lemmas. 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 $[\![\varphi]\!](s) - [\![\varphi]\!](t) \leq [s \preceq_1 t]$.*

2. *For all $\varphi \in q\mu$, and for all $s,t \in S$, we have $|[\![\varphi]\!](s) - [\![\varphi]\!](t)| \leq [s \simeq_g t]$.*

Proof. We prove the first assertion. The proof is by induction on the structure of a (possibly open) formula $\varphi \in q\mu_1^+$. Call a variable valuation ξ *bounded* if, for all variables $V \in MVars$

and states s, t, we have that $\xi(V)(s) - \xi(V)(t) \leq [s \preceq_1 t]$. We prove by induction that for all $s, t \in S$, for all bounded variable valuations ξ, we have $[\![\varphi]\!]_\xi(s) - [\![\varphi]\!]_\xi(t) \leq [s \preceq_1 t]$. For clarity, we sometimes omit writing the variable valuation ξ.

The base case for constants is trivial, and the case for observation variables follows since $[s \equiv t] \leq [s \preceq_1 t]$. The case for variables $V \in \mathit{MVars}$ follows from the assumption of bounded variable valuations. For $\varphi_1 \vee \varphi_2$, assume the induction hypothesis for φ_1, φ_2, and note that

$$\begin{aligned}\big([\![\varphi_1]\!](s) \sqcup [\![\varphi_2]\!](s)\big) - \big([\![\varphi_1]\!](t) \sqcup [\![\varphi_2]\!](t)\big) \\ \leq \big([\![\varphi_1]\!](s) - [\![\varphi_1]\!](t)\big) \sqcup \big([\![\varphi_2]\!](s) - [\![\varphi_2]\!](t)\big) \leq [s \preceq_1 t]\,.\end{aligned}$$

The proof for $\wedge$ is similar. For $\varphi_1 \oplus c$ and $\varphi_1 \ominus c$, we have by induction hypothesis that $[\![\varphi_1]\!](s) - [\![\varphi_1]\!](t) \leq [s \preceq_1 t]$, and so the "shifted versions" also satisfy the same bound.

For the induction step for pre_1, assume the induction hypothesis for φ, and note that we can choose $k \in C([\preceq_1])$ such that $k(s) = [\![\varphi]\!](s)$ at all $s \in S$. We have, for all $s, t \in S$,

$$[\![\mathrm{pre}_1(\varphi)]\!](s) - [\![\mathrm{pre}_1(\varphi)]\!](t) \leq \sup_{k \in C([\preceq_1])} \big(\mathrm{Pre}_1(k)(s) - \mathrm{Pre}_1(k)(t)\big) \leq [s \preceq_1 t]\,. \tag{4.29}$$

where the last inequality follows by noting that $[\preceq_1]$ is a fixpoint of $H_{\preceq_1}$.

The proof for the fixpoint operators is performed by considering their Picard iterates. We consider the case $\mu Z.\varphi$, the proof for $\nu Z.\varphi$ is similar. Let ξ be a bounded variable valuation. Then, the variable valuation $\xi_0 = \xi[Z \mapsto \mathbf{0}]$ is also bounded, and by induction hypothesis, the formula φ when evaluated in the variable valuation ξ_0 satisfies

$$[\![\varphi]\!]_{\xi_0}(s) - [\![\varphi]\!]_{\xi_0}(t) \leq [s \preceq_1 t]\,. \tag{4.30}$$

Now consider the variable valuation $\xi_1 = \xi[Z \mapsto [\![\varphi]\!]_{\xi_0}]$. From Equation (4.30), we get that ξ_1 is bounded, and again, by induction hypothesis, we have that $[\![\varphi]\!]_{\xi_1}(s) - [\![\varphi]\!]_{\xi_1}(t) \leq [s \preceq_1 t]$. In general, for $k \geq 0$, consider the variable valuation $\xi_{k+1} = \xi[Z \mapsto [\![\varphi]\!]_{\xi_k}]$. By the above argument, each variable valuation ξ_k is bounded, and so for every $k \geq 0$, we have

$$[\![\varphi]\!]_{\xi_k}(s) - [\![\varphi]\!]_{\xi_k}(t) \leq [s \preceq_1 t] \,. \tag{4.31}$$

Taking the limit, as $k \to \infty$, we have that

$$\lim_{k\to\infty} ([\![\varphi]\!]_{\xi_k}(s) - [\![\varphi]\!]_{\xi_k}(t)) = [\![\mu Z.\varphi]\!]_{\xi}(s) - [\![\mu Z.\varphi]\!]_{\xi}(t) \leq [s \preceq_1 t] \,. \tag{4.32}$$

The proof of the second assertion can be done along the same lines, using the symmetry of $\simeq_g$. The proof is again by induction on the structure of the formula. In particular, (4.29) can be proved for either player: for $n \geq 0$ and $i \in \{1, 2\}$,

$$[\![\mathrm{pre}_i(\varphi)]\!](s) - [\![\mathrm{pre}_i(\varphi)]\!](t) \leq \sup_{k \in C([\simeq_g])} \big(\mathrm{Pre}_i(k)(s) - \mathrm{Pre}_i(k)(t)\big) \leq [s \simeq_g t] \,.$$

Negation can be dealt with by noting that $[\![\neg\varphi]\!](s) - [\![\neg\varphi(t)]\!] = [\![\varphi]\!](t) - [\![\varphi(s)]\!]$, and by using the symmetry of $\simeq_g$; the other cases are similar. ■

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.*

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

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

Proof. We show by induction on n that $d_n(s,t) \leq \sup_{\varphi \in q\mu}(\llbracket \varphi \rrbracket(s) - \llbracket \varphi \rrbracket(t))$. The base case is trivial. For the induction step, the distance is:

$$d_{i+1}(s,t) = \sup_{k \in C(d_i)} \left(\mathrm{Pre}_1(k)(s) - \mathrm{Pre}_1(k)(t)\right) . \tag{4.33}$$

The challenge is to show that, for all $s, t \in S$, we can construct a formula ψ_{st} that witnesses the distance within an arbitrary $\varepsilon > 0$:

$$d_{i+1}(s,t) - \varepsilon \leq \llbracket \psi_{st} \rrbracket(s) - \llbracket \psi_{st} \rrbracket(t) . \tag{4.34}$$

To this end, let $k^\star$ be the value of k that realizes the sup in (4.33) within $\varepsilon/4$. By induction hypothesis, for each pair of states s' and t' we can choose $\varphi'_{s't'}$ such that

$$d_i(s',t') - \varepsilon/4 \leq \llbracket \varphi'_{s't'} \rrbracket(s') - \llbracket \varphi'_{s't'} \rrbracket(t') . \tag{4.35}$$

Let $\varphi_{s't'}$ be a shifted version of $\varphi'_{s't'}$, such that $\varphi_{s't'}(s') = k^\star(s')$:

$$\varphi_{s't'} = \varphi'_{s't'} \oplus (k^\star(s') - \llbracket \varphi'_{s't'} \rrbracket(s')) . \tag{4.36}$$

We now prove that:

$$\llbracket \varphi_{s't'} \rrbracket(s') = k^\star(s') \tag{4.37}$$

$$\llbracket \varphi_{s't'} \rrbracket(t') \leq k^\star(t') + \varepsilon/4 . \tag{4.38}$$

Equality (4.37) is immediate from (4.36). We prove (4.38) as follows. We can rewrite (4.35) as

$$\llbracket \varphi'_{s't'} \rrbracket(t') - \varepsilon/4 \leq \llbracket \varphi'_{s't'} \rrbracket(s') - d_i(s',t') . \tag{4.39}$$

Since $k^\star \in C(d_i)$, we have $k^\star(s') - k^\star(t') \leq d_i(s',t')$, or

$$k^\star(t') - k^\star(s') \geq -d_i(s',t') . \tag{4.40}$$

Plugging this relation into (4.39), we obtain

$$[\![\varphi'_{s't'}]\!](t') - \varepsilon/4 \leq [\![\varphi'_{s't'}]\!](s') + k^\star(t') - k^\star(s') \; . \tag{4.41}$$

Plugging this relation into (4.36) evaluated at t', we obtain

$$[\![\varphi_{s't'}]\!](t') - \varepsilon/4 \quad \leq \quad [\![\varphi'_{s't'}]\!](s') + k^\star(t') - k^\star(s') \oplus \big(k^\star(s') - [\![\varphi'_{s't'}]\!](s')\big),$$

or

$$[\![\varphi_{s't'}]\!](t') - \varepsilon/4 \quad \leq \quad k^\star(t') - \big(k^\star(s') - [\![\varphi'_{s't'}]\!](s')\big) \oplus \big(k^\star(s') - [\![\varphi'_{s't'}]\!](s')\big) \quad \leq \quad k^\star(t'),$$

which proves (4.38). Define now $\varphi_{s'} = \bigwedge_{t'} \varphi_{s't'}$. From (4.37) and (4.38) we have

$$[\![\varphi_{s'}]\!](s') = k^\star(s') \tag{4.42}$$

$$[\![\varphi_{s'}]\!](t') \leq k^\star(t') + \varepsilon/4 \; . \tag{4.43}$$

Define then $\varphi = \bigvee_{s'} \varphi_{s'}$. From (4.42), (4.43), we have that

$$k^\star(s') \leq [\![\varphi]\!](s') \leq k^\star(s') + \varepsilon/4 \; . \tag{4.44}$$

for all $s' \in S$. As formula ψ_{st}, we propose thus to take the formula $\mathrm{pre}(\varphi)$. From (4.44), we have that $|[\![\psi_{st}]\!](s) - \mathrm{Pre}_1(k^\star)(s)| \leq \varepsilon/4$, and similarly, $|[\![\psi_{st}]\!](t) - \mathrm{Pre}_1(k^\star)(t)| \leq \varepsilon/4$. By comparison with (4.33), and by the fact that $k^\star$ realizes the sup within $\varepsilon/4$, we finally have (4.34), as desired. ■

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

Theorem 7 *The following assertions hold for all game structures G, and for all states s, t of G:*

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

We note that, due to Theorem 5, 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.

Our metrics are not characterized by the probabilistic temporal logic PCTL [HJ94, ASB+95]. In fact, the values of PCTL formulas can change from true to false when certain probabilities cross given thresholds, so that PCTL formulas can have different boolean values on games that are very close in transition probabilities, and hence, very close in our metric. Quantitative metrics such as the ones developed in this paper are suited to quantitative-valued formulas, such as those of $q\mu$.

4.5 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 8 *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

$$\forall k \in B(R).\forall x_1 \in \mathcal{D}_1(s).\exists y_1 \in \mathcal{D}_1(t).\forall y_2 \in \mathcal{D}_2(t).\exists x_2 \in \mathcal{D}_2(s).\left(\mathbb{E}_t^{y_1,y_2}(k) \geq \mathbb{E}_s^{x_1,x_2}(k)\right) . \tag{4.45}$$

Proof. The proof proceeds by induction on the computation of the fixpoint relation R. We first present the case for $\preceq_1$. Call R_n the n-th iterate of the simulation relation R, and let d_n be the n-th iterate of $[\preceq_1]$, as in Theorem 3. We prove by induction that, for all states $s, t \in S$, we have sR_nt iff $d_n(s,t) = 0$. We define $d_0(s,t) = p(s,t)$. The base case is then immediate because sR_0t iff $d_0(s,t) = 0$. Consider the induction step, for $n \geq 0$, and consider any states $s, t \in S$. Assume first that $d_{n+1}(s,t) > 0$: then, it is easy to show that we can find a value for k in (4.45) that witnesses $(s,t) \notin R_{n+1}$, since the constraints on k due to $B(R_n)$ are weaker than those due to $C(d_n)$. Conversely, assume that there is a $k \in B(R_n)$ that witnesses $(s,t) \notin R_{n+1}$. Then, by scaling all k values so that they are all smaller than the smallest non-zero value of $d_n(s',t')$ for any $s', t' \in S$, we can find a $k' \in C(d_n)$ which also witnesses $d_{n+1}(s,t) > 0$, as required.

The case for $\simeq_g$ is analogous, due to the similarity of the Picard iterations (4.14) for $\preceq_1$ and (4.15) for $\simeq_g$. ■

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 $[\![\varphi]\!](s) = [\![\varphi]\!](t)$ holds for every $\varphi \in q\mu$ and $s \preceq_1 t$ iff $[\![\varphi]\!](s) \leq [\![\varphi]\!](t)$ holds for every $\varphi \in q\mu_1^+$.*

4.6 Relation between Game Metrics and (Bi-)simulation Metrics

The a priori metrics assume an adversarial relationship between the players. We show that, on turn-based games, the a priori bisimulation metric coincides with the classical bisimulation metric where the players cooperate.

We define such "cooperative" simulation and bisimulation metrics $[\preceq_{12}]$ and $[\simeq_{12}]$ as the metric analog of classical (bi)simulation [Mil90, SL94]. We define the metric transformers $H_{\preceq_{12}} : \mathcal{M} \mapsto \mathcal{M}$ and $H_{\simeq_{12}} : \mathcal{M} \mapsto \mathcal{M}$, for all metrics $d \in \mathcal{M}$ and $s, t \in S$, by:

$$H_{\preceq_{12}}(d)(s,t) = p(s,t) \sqcup \sup_{k \in C(d)} \sup_{x_1 \in \mathcal{D}_1(s)} \sup_{x_2 \in \mathcal{D}_2(s)} \inf_{y_2 \in \mathcal{D}_2(t)} \inf_{y_1 \in \mathcal{D}_1(t)} \{\mathbb{E}_s^{x_1,x_2}(k) - \mathbb{E}_t^{y_1,y_2}(k)\} .$$

$$H_{\simeq_{12}}(d)(s,t) = H_{\preceq_{12}}(d)(s,t) \sqcup H_{\preceq_{12}}(d)(t,s) .$$

The metrics $[\preceq_{12}]$ and $[\simeq_{12}]$ are defined as the least fixed points of $H_{\preceq_{12}}$ and $H_{\simeq_{12}}$ respectively. The kernel of these metrics define the classical probabilistic simulation and bisimulation relations.

Theorem 9 *The following assertions hold.*

1. *On turn-based game structures,* $[\simeq_g] = [\simeq_{12}]$.

2. *There is a deterministic game structure G and states* s, t *in G such that* $[s \simeq_g t] > [s \simeq_{12} t]$.

3. *There is a deterministic game structure G and states* s, t *in G such that* $[s \simeq_g t] < [s \simeq_{12} t]$.

Proof. For the first part, since we have turn-based games, only one player has a choice of moves at each state. We say that a state s belongs to player $i \in \{1, 2\}$ if player $\sim i$ has

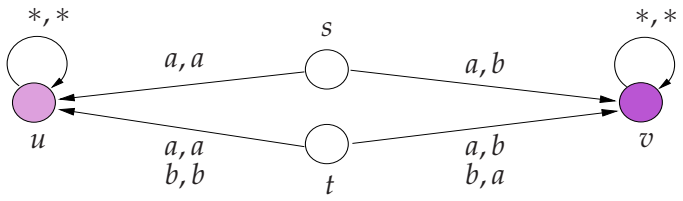


Figure 4.2: $[s \simeq_g t] = \frac{1}{2}$ and $[s \simeq_{12} t] = 0$

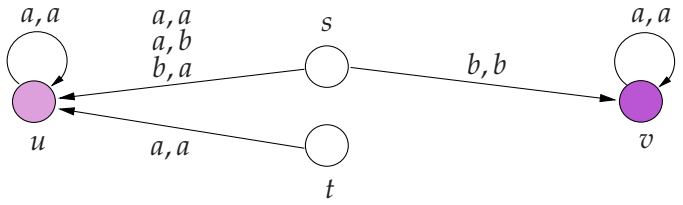


Figure 4.3: $[s \simeq_g t] = 0$ but $[s \simeq_{12} t] = 1$.

only one move at s. First, notice that due to the presence of the variable *turn*, the metric distance between states belonging to different players is always 1, for all the metrics we consider. Thus, we focus on the metric distances between states belonging to the same player. Consider two player 1 states $s, t \in S$. From the definitions of $H_{\preceq_1}$ and $H_{\preceq_{12}}$, for $d \in \mathcal{M}$, by dropping the moves of player 2, it is easy to see that $H_{\preceq_1}(d) = H_{\preceq_{12}}(d)$, and $H_{\simeq_g}(d) = H_{\simeq_{12}}(d)$. Since this holds for any $d \in \mathcal{M}$, it holds for the fixpoints, $[\simeq_g]$ and $[\simeq_{12}]$.

The second part is proved by the game in Figure 4.2, where $p(s,t) = 0$ and $p(u,v) = 1$. The latter yields $[u \simeq_g v] = 1$. Since player 1 has no choice of moves at state s, the maximum probability with which player 1 can guarantee a transition to either state u or state v is 0. But from state t, by playing moves a, b with probability $\frac{1}{2}$ each, player 1 can guarantee reaching states u and v with probability $\frac{1}{2}$, which implies that over all $k \in C(d)$, given that $d(u,v) = 1$ from $[u \simeq_g v] = 1$, the maximum k expectation that player 1 can

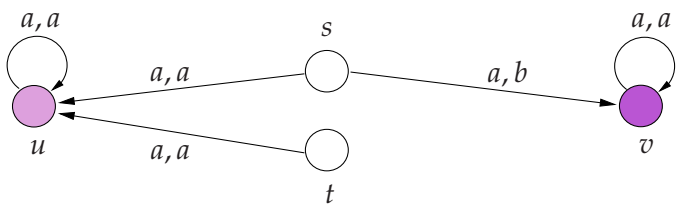


Figure 4.4: $[s \preceq_1 t] = 0$ and $[s \preceq_{12} t] = 1$. Also, $[t \preceq_1 s] = 1$ and $[t \preceq_{12} s] = 0$.

guarantee is $\frac{1}{2}$. Therefore $[s \simeq_g t] = \frac{1}{2}$. But if player 2 co-operates, then $[s \simeq_{12} t] = 0$.

The third part is proved by the game in Figure 4.3 where again $p(s,t) = 0$ and $p(u,v) = 1$. Since the players don't have any moves to transition to state v from state t, $[s \simeq_{12} t] = 1$, whereas $[s \simeq_g t] = 0$. ■

If we consider Markov decision processes (MDPs), we have that on i-MDPs, the metric $\preceq_i$ coincides with $\preceq_{12}$, since player $\sim i$ has no moves, for $i \in \{1,2\}$. On the other hand, the metric $\preceq_{\sim i}$ provides no information on $\preceq_{12}$.

Theorem 10 *The following assertions hold.*

1. *For i-MDPs we have* $[\preceq_i] = [\preceq_{12}]$.

2. *There is a deterministic 2-MDP G with states* s, t *such that* $[s \preceq_1 t] < [s \preceq_{12} t]$.

3. *There is a deterministic 2-MDP G with states* s, t *such that* $[s \preceq_1 t] > [s \preceq_{12} t]$.

Proof. From the definitions of $H_{\preceq_1}$ and $H_{\preceq_{12}}$, restricted to MDPs, where only one player has a choice of moves, the first assertion follows.

The second and third assertions are proved by the deterministic 2-MDP in Figure 4.4, where again $p(s,t) = 0$ and $p(u,v) = 1$. For the second assertion we note that since $d(u,v) = 1$, for any choice of $k \in C(d)$, player 1 cannot get a higher expectation of k

from state s when compared to state t, because at state s, player 2 always has a move that will lead to a state yielding a lower k expectation. Therefore, $[s \preceq_1 t] = 0$. Further, for $k(v) = 1$ and $k(u) = 0$, which satisfies the constraints on k, we have no moves for either player from state t, which implies $[s \preceq_{12} t] = 1$.

We prove the third assertion by showing that, for the 2-MDP of Figure 4.4, we have $[t \preceq_1 s] > [t \preceq_{12} s]$ (which is the third assertion, with s and t exchanged). Note that when player 2 cooperates, the expectation of any $k \in C(d)$ from state s is always at least as much as the expectation from state t. Thus $[t \preceq_{12} s] = 0$. Finally, there exists a $k \in C(d)$, with $k(u) = 1$ and $k(v) = 0$, for which $[t \preceq_1 s] = 1$, which completes the proof. ■

4.7 Decidability

We now show that the metrics are computable to any degree of precision. This follows since the definition of the distance between two states of a given game, as the least fixpoint of the metric transformer (4.9), can be written as a formula in the theory of reals, which is decidable [Tar51]. Since the distance between two states may not be rational, we can only guarantee an approximate computation in general.

Without loss of generality, we assume that the states of G are labeled $\{s_1, \ldots, s_n\}$ for some $n \in \mathbb{N}$. The construction is standard (see, e.g., [dAM04]), we recapitulate the main steps. We denote by $\mathbf{R}$ the real-closed field $(\mathbb{R}, +, \cdot, 0, 1, \leq)$ of the reals with addition and multiplication. An *atomic formula* is an expression of the form $p > 0$ or $p = 0$ where p is a (possibly) multi-variate polynomial with integer coefficients. An *elementary formula* is

constructed from atomic formulas by the grammar

$$\varphi ::= a \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \exists x.\varphi \mid \forall x.\varphi,$$

where a is an atomic formula, $\wedge$ denotes conjunction, $\vee$ denotes disjunction, $\neg$ denotes complementation, and $\exists$ and $\forall$ denote existential and universal quantification respectively. We write $\varphi \rightarrow \varphi'$ as shorthand for $\neg\varphi \vee \varphi'$. The semantics of elementary formulas are given in a standard way [CK73]. A variable x is *free* in the formula φ if it is not in the scope of a quantifier $\exists x$ or $\forall x$. An *elementary sentence* is a formula with no free variables. The theory of real-closed fields is decidable [Tar51].

We introduce additional atomic formulas as syntactic sugar: for polynomials p_1 and p_2, we write $p_1 = p_2$ for $p_1 - p_2 = 0$, $p_1 > p_2$ for $p_1 - p_2 > 0$, and $p_1 \geq p_2$ for $p_1 - p_2 = 0 \vee p_1 - p_2 > 0$. Also, we write $p_1 \leq p_2$ for $p_2 \geq p_1$ and $p_1 < p_2$ for $p_2 > p_1$. Let $\vec{x}, \vec{y}$ denote vectors of variables, where the dimensions of the vectors will be clear from the context. For $\sim \in \{=, \leq, \geq\}$, we write $\vec{x} \sim \vec{y}$ for the pointwise ordering, that is, if $\bigwedge_i x_i \sim y_i$. A subset $C \subseteq \mathbb{R}^m$ is *definable* in $\mathbf{R}$ if there exists an elementary formula $\varphi_C(\vec{x})$ such that for any $\vec{x}_0 \in \mathbb{R}^m$, we have $\varphi_C(\vec{x}_0)$ holds in $\mathbf{R}$ iff $\vec{x}_0 \in C$. A function $f : \mathbb{R}^k \rightarrow \mathbb{R}^m$ is *definable* in $\mathbf{R}$ if there exists an elementary formula $\varphi_f(\vec{y}, \vec{x})$ with free variables $\vec{y}$, $\vec{x}$ such that for all constants $\vec{y}_0 \in \mathbb{R}^m$ and $\vec{x}_0 \in \mathbb{R}^k$ the formula $\varphi_f(\vec{y}_0, \vec{x}_0)$ is true in $\mathbf{R}$ iff $\vec{y}_0 = f(\vec{x}_0)$. We start with some simple observations about definability.

Lemma 5 *(a) If functions $f_1 : \mathbb{R}^k \rightarrow \mathbb{R}^m$ and $f_2 : \mathbb{R}^k \rightarrow \mathbb{R}^m$ are definable in $\mathbf{R}$ then so are the functions*

$$(f_1 - f_2)(\vec{x}) = f_1(\vec{x}) - f_2(\vec{x})$$

$$(f_1 \sqcup f_2)(\vec{x}) = f_1(\vec{x}) \sqcup f_2(\vec{x})$$

(b) If $f : \mathbb{R}^{k+l} \to \mathbb{R}^m$ is definable in $\mathbf{R}$, and $C \subseteq \mathbb{R}^k$ is definable in $\mathbf{R}$, then $(\sup_C f) : \mathbb{R}^l \to \mathbb{R}^m$ defined as

$$(\sup_C f)(\vec{y}) = \sup_{\vec{x} \in C} f(\vec{x}, \vec{y})$$

is definable in $\mathbf{R}$.

Proof. For part (a), let $\varphi_1(\vec{y}, \vec{x})$ and $\varphi_2(\vec{y}, \vec{x})$ be formulas defining f_1 and f_2 respectively. Then, $f_1 - f_2$ is defined by the formula

$$\exists \vec{z}_1 . \exists \vec{z}_2 . (\varphi_1(\vec{z}_1, \vec{x}) \wedge \varphi_2(\vec{z}_2, \vec{x}) \wedge \vec{y} = \vec{z}_1 - \vec{z}_2),$$

and $f_1 \sqcup f_2$ is defined by the formula

$$\exists \vec{z}_1 . \exists \vec{z}_2 . (\varphi_1(\vec{z}_1, \vec{x}) \wedge \varphi_2(\vec{z}_2, \vec{x}) \wedge \bigwedge_i [(\vec{z}_{1,i} \geq \vec{z}_{2,i} \wedge \vec{y}_i = \vec{z}_{1,i}) \vee (\vec{z}_{1,i} < \vec{z}_{2,i} \wedge \vec{y}_i = \vec{z}_{2,i})]) .$$

For part (b), let $\varphi_f(\vec{z}, \vec{x}, \vec{y})$ define f, where $\vec{x}$ is of dimension k, $\vec{y}$ of dimension l, and $\vec{z}$ of dimension m, respectively. Let $\psi_C(\vec{x})$ define C. Then, the following formula with free variables $\vec{z}$, $\vec{y}$ (call it $\varphi(\vec{z}, \vec{y})$) states that $\vec{z}$ is an upper bound of $f(\vec{x}, \vec{y})$ for all $\vec{x} \in C$:

$$\forall \vec{x}_1 . \forall \vec{z}_1 . (\psi_C(\vec{x}_1) \wedge \varphi_f(\vec{z}_1, \vec{x}_1, \vec{y}) \to \vec{z}_1 \leq \vec{z}),$$

and $\sup_C f$ is defined by the formula with free variables $\vec{z}$, $\vec{y}$ given by:

$$\varphi(\vec{z}, \vec{y}) \wedge \forall \vec{z}_1 . (\varphi(\vec{z}_1, \vec{y}) \to \vec{z} \leq \vec{z}_1) .$$

■

Theorem 11 *Let G be a game structure and s, t states of G. For all rationals v, and all $\epsilon > 0$, it is decidable if $|[s \preceq_1 t] - v| < \epsilon$ and if $|[s \simeq_g t] - v| < \epsilon$. It is decidable if $s \preceq_1 t$ and if $s \simeq_g t$.*

Proof. First, we use a result of Weyl [Wey50] that the minmax value of a matrix game with payoffs in $\mathbb{R}$ can be written as an elementary formula in the theory of real-closed fields. This implies that for any state s, the function $\mathrm{Pre}_1(\vec{k})(s)$ is definable in $\mathbf{R}$. Also, for $d \in \mathcal{M}$, the set $C(d)$ is definable in $\mathbf{R}$ (since conjunctions of linear constraints are definable in $\mathbf{R}$). Hence, by Lemma 5(a) and (b), we have that $\sup_{\vec{k} \in C(d)} \left(\mathrm{Pre}_1(\vec{k})(s) - \mathrm{Pre}_1(\vec{k})(t)\right)$ is definable for any metric $d \in \mathcal{M}$, and states s and t of G. By another application of Lemma 5(a), we have that the function

$$H_{\preceq_1}(d)(s,t) = (s \equiv t) \sqcup \sup_{\vec{k} \in C(d)} \left(\mathrm{Pre}_1(\vec{k}(s) - \mathrm{Pre}_1(\vec{k})(t)\right) .$$

is definable for $d \in \mathcal{M}$ and states s and t of G.

Consider the set of free variables $\{y(s,t), d(s,t) \mid s,t \in S\}$, where d is a vector of n^2 free variables defining the metric d, and where y is a vector of n^2 variables. Let $\varphi(y,d)$ be a formula in $\mathbf{R}$, with free variables in the above set, such that $\varphi(y,d)$ is true iff $y(s,t) = H_{\preceq_1}(d)(s,t)$ holds for all $s,t \in S$. Then the formula $\varphi^*(y)$ with free variables y, defined as:

$$\exists d.(\varphi(y,d) \wedge y = d),$$

defines a fixpoint of $H_{\preceq_1}(d)$. Finally, the formula $\psi(y)$, given by

$$\varphi^*(y) \wedge \forall y'.(\varphi^*(y') \rightarrow y \leq y') .$$

defines the least fixpoint of $H_{\preceq_1}$ (again, $y' = \{y'(s,t) \mid s,t \in S\}$ is a matrix of n^2 variables, and $y \leq y'$ iff $y(s,t) \leq y'(s,t)$ for all $s,t \in S$). Thus, $\psi(y)$ is true iff $y(s,t) = [s \preceq_1 t]$ for all $s,t \in S$.

While this shows that $[s \preceq_1 t]$ is algebraic, there are game structures G with all transition probabilities being rational, but with states s and t of G such that $[s \preceq_1 t]$ is

irrational. So, we use the formula above to approximate the value of $[s \preceq_1 t]$ to within a constant ϵ. For states s, t and rationals v, ϵ, we have that $|[s \preceq_1 t] - v| < \epsilon$ iff $\exists y.(\psi(y) \wedge |y(s,t) - v| < \epsilon)$ is valid, and this can be decided since $\mathbf{R}$ is decidable.

A similar construction shows that the question whether $|[s \simeq_g t] - v| < \epsilon$, is decidable for states s, t and rationals v, ϵ: we ensure that y is a symmetric fixpoint by conjoining to $\varphi^*(y)$ constraints $y(s,t) = y(t,s)$ for all states s, t.

If the formula $\exists y.(\psi(y) \wedge y(s,t) = 0)$, where we assert that the distance between s and t is zero, is valid, we can conclude that $s \preceq_1 t$. This implies that the relation $s \preceq_1 t$ is decidable for any game structure G and states s and t of G. A similar construction for $\simeq_g$ shows that the relation $s \simeq_g t$ is also decidable for any game structure G and states s, t of G. ■

4.8 Discussion

Our derivation of $\preceq_i$ and $\simeq_g$, for $i \in \{1, 2\}$, as kernels of metrics, seems somewhat abstruse: most equivalence or similarity relations have been defined, after all, without resorting to metrics. We now point out how a generalization of the usual definitions [SL94, AHKV98, DGJP99, DGJP03], suggested in [dAHM03, Maj03], fails to produce the "right" relations. Furthermore, the flawed relations obtained as a generalization of [SL94, AHKV98, DGJP99, DGJP03] are no simpler than our definitions, based on kernel metrics. Thus, our study of game relations as kernels of metrics carries no drawbacks in terms of leading to more complicated definitions. Indeed, we believe that the metric approach is the superior one for the study of game relations.

We outline the flawed generalization of [SL94, AHKV98, DGJP99, DGJP03] as proposed in [dAHM03, Maj03], explaining why it would seem a natural generalization. The alternating simulation of [AHKV98] is defined over deterministic game structures. Player-i alternating simulation, for $i \in \{1,2\}$, is the largest relation R satisfying the following conditions, for all states $s,t \in S$: $s\,R\,t$ implies $s \equiv t$ and $\forall x_i \in \Gamma_i(s)\,.\,\exists y_i \in \Gamma_i(t)\,.\,\forall y_{\sim i} \in \Gamma_{\sim i}(t)\,.\,\exists x_{\sim i} \in \Gamma_{\sim i}(s)\,.\,\tau(s,x_1,x_2)\;R\;\tau(t,y_1,y_2)$.

The MDP relations of [SL94], later extended to metrics by [DGJP99, DGJP03], rely on the fixpoint (4.2), where sup plays the role of $\forall$, inf plays the role of $\exists$, and R is replaced by distribution equality modulo R, or $\sqsubseteq_R$. This strongly suggests — incorrectly — that equivalences for general games (probabilistic, concurrent games) can be obtained by taking the double quantifier alternation $\forall\exists\forall\exists$ in the definition of alternating simulation, changing all $\forall$ into sup, all $\exists$ into inf, and replacing R by $\sqsubseteq_R$. The definition that would result is as follows. We parameterize the new relations by a player $i \in \{1,2\}$, as well as by whether mixed moves or only pure moves are allowed. For a relation $R \subseteq S \times S$, for $M \in \{\Gamma, \mathcal{D}\}$, for all $s,t \in S$ and $i \in \{1,2\}$ consider the following conditions:

- **(loc)** $s\,R\,t$ implies $s \equiv t$.
- **(M-i-altsim)** $s\,R\,t$ implies

  $\forall x_i \in M_i(s)\,.\,\exists y_i \in M_i(t)\,.\,\forall y_{\sim i} \in M_{\sim i}(t)\,.\,\exists x_{\sim i} \in M_{\sim i}(s)\,.\,\delta(s,x_1,x_2) \sqsubseteq_R \delta(t,y_1,y_2)$;

We then define the following relations:

- For $i \in \{1,2\}$ and $M \in \{\Gamma, \mathcal{D}\}$, *player-$i$ M-alternating simulation* $\sqsubseteq_i^M$ is the largest relation that satisfies (loc) and (M-i-altsim).

– For $i \in \{1,2\}$ and $M \in \{\Gamma, \mathcal{D}\}$, *player-i M-alternating bisimulation* $\cong_i^M$ is the largest symmetrical relation that satisfies (loc) and (M-i-altsim).

Over deterministic game structures, the definitions of $\sqsubseteq_i^\Gamma$ and $\cong_i^\Gamma$ coincide with the alternating simulation and bisimulation relations of [AHKV98]. In fact, $\sqsubseteq_i^\Gamma$ and $\cong_i^\Gamma$ capture the *deterministic* semantics of $q\mu$, and thus in some sense generalize the results of [AHKV98] to probabilistic game structures.

Theorem 12 *For any game structure G and states s, t of G, the following assertions hold:*

1. $s \cong_i^\Gamma t$ *iff* $[\![\varphi]\!]^\Gamma(s) = [\![\varphi]\!]^\Gamma(t)$ *holds for every* $\varphi \in q\mu_i$.

2. $s \sqsubseteq_i^\Gamma t$ *iff* $[\![\varphi]\!]^\Gamma(s) \leq [\![\varphi]\!]^\Gamma(t)$ *holds for every* $\varphi \in q\mu_i^+$.

The following lemma states that $\sqsubseteq_i^\mathcal{D}$ and $\cong_i^\mathcal{D}$ are the kernels of $[\sqsubseteq_i]$ and $[\cong_i]$, connecting thus the result of combining the definitions of [SL94] and [AHKV98] with a posteriori metrics.

Lemma 6 *For all game structures G, all players $i \in \{1,2\}$, and all states s, t of G, we have $s \sqsubseteq_i^\mathcal{D} t$ iff $[s \sqsubseteq_i t] = 0$, and $s \cong_i^\mathcal{D} t$ iff $[s \cong_i t] = 0$.*

We are now in a position to prove that neither the Γ-relations not the $\mathcal{D}$-relations are the "canonical" relations on general concurrent games, since neither characterizes $[\![q\mu]\!]$. In particular, the $\mathcal{D}$-relations are too fine, and the Γ-relations are incomparable with the relations $\preceq_i$ and $\simeq_g$, for $i \in \{1,2\}$. We prove these negative results first for the $\mathcal{D}$-relations. They follow from Theorem 5 and 7.

Theorem 13 *The following assertions hold:*

1. *For all game structures G, all states s, t of G, and all $i \in \{1,2\}$, we have that $s \sqsubseteq_i^{\mathcal{D}} t$ implies $s \preceq_i t$, and $s \cong_i^{\mathcal{D}} t$ implies $s \simeq_i t$.*

2. *There is a game structure G, and states s, t of G, such that $s \preceq_i t$ but $s \not\sqsubseteq_i^{\mathcal{D}} t$.*

3. *There is a game structure G, and states s, t of G, such that $[\![\varphi]\!](s) = [\![\varphi]\!](t)$ for all $\varphi \in q\mu$, but $s \not\cong_i^{\mathcal{D}} t$ for some $i \in \{1,2\}$.*

We now turn our attention to the Γ-relations, showing that they are incomparable with $\preceq_i$ and $\simeq_g$, for $i \in \{1,2\}$.

Theorem 14 *The following assertions hold:*

1. *There exists a deterministic game structure G and states s, t of G such that $s \sqsubseteq_1^{\Gamma} t$ but $s \not\preceq_1 t$, and $s \cong_1^{\Gamma} t$ but $s \not\simeq_g t$.*

2. *There exists a turn-based game structure G and states s, t of G such that $s \preceq_1 t$ but $s \not\sqsubseteq_1^{\Gamma} t$. and $s \simeq_g t$ but $s \not\cong_1^{\Gamma} t$.*

Proof. The first assertion is proved via the deterministic game in Figure 4.5, where $p(s,t) = 0$ and $p(u,v) = 1$ and $\Gamma_1(s) = \Gamma_2(s) = \{a,b\}$ and $\Gamma_1(t) = \Gamma_2(t) = \{a,b,c\}$. In the figure, we use the variables x and y to represent moves: if player 1 and player 2 moves coincide, u is the successor state, otherwise it is v. Thus, the game from s is the usual "penny-matching" game; the game from t is a version of "penny-matching" with 3-sided pennies.

It can be seen that $s \sqsubseteq_1^{\Gamma} t$. On the other hand, we have $s \not\preceq_1 t$. Indeed, from state s, by playing both a and b with probability $\frac{1}{2}$, player 1 can ensure that the probability of a transition to u is $\frac{1}{2}$. On the other hand, from state t, player 1 can achieve at most

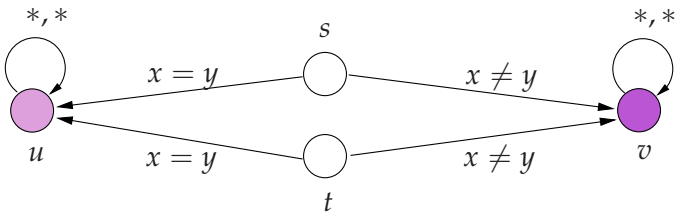


Figure 4.5: $s \sqsubseteq_1^\Gamma t$ but $s \not\preceq_1 t$ and $s \cong_1^\Gamma t$ but $s \not\simeq_g t$

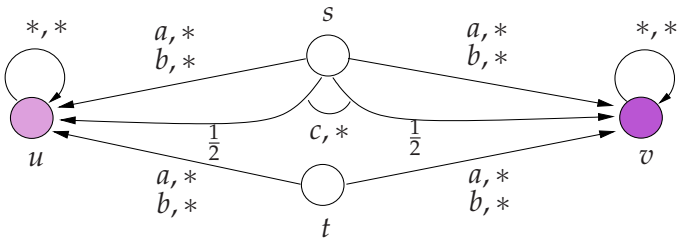


Figure 4.6: $s \preceq_1 t$ but $s \not\sqsubseteq_1^\Gamma t$ and $s \simeq_g t$ but $s \not\cong_1^\Gamma t$.

probability $\frac{1}{3}$ of reaching u (this maximal probability is achieved by playing all of a, b, c with probability $\frac{1}{3}$). The result then follows using Theorem 7.

The second assertion is proved via the game in Figure 4.6. We have $s \not\sqsubseteq_1^\Gamma t$: clearly, player-1's move c at state s cannot be mimicked at t when the game is restricted to pure moves. On the other hand, we have $s \preceq_1 t$: since the move c at s can be imitated via the mixed move that plays both a and b at t with probability $\frac{1}{2}$ each, all $q\mu$ formulas have the same value, under $[\![\cdot]\!]$, at s and t, and the result follows once more using Theorem 7.

■

Finally, we remark that, in view of Theorem 8, the definitions of the relations $\preceq_i$ and $\simeq_g$ for $i \in \{1, 2\}$ are no more complex than the definitions of $\sqsubseteq_1^{\mathcal{D}}$, $\sqsubseteq_1^\Gamma$, $\cong_1^{\mathcal{D}}$, and $\cong_1^\Gamma$.

Chapter 5

Discounted, Average and Total Rewards

In this chapter we show that the *undiscounted* a priori metrics we developed in Chapter 4 also provide a bound for the difference in discounted and average values across states of a game structure. We claim that given these results, in addition to the reciprocity and the logical characterization, the a priori metrics are the *canonical* metrics in stochastic games. We therefore refer to the a priori metrics simply as game metrics. We introduce *discounted* game simulation and bisimulation metrics and show that the discounted metrics do not provide a bound for the difference in discounted values across states of a game. We then introduce a new *total reward metric* that provides a bound for the discounted values, average reward values and total reward values across states of a game and conclude by showing that the kernels of the undiscounted game metrics, the discounted game metrics and the total reward metric coincide.

We begin by generalizing valuations from functions that return values in the unit interval to those that return values in a fixed, non-singleton real interval $[\theta_1, \theta_2]$. 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 $a, b \in \mathbb{R}$, the definitions of $a \sqcup b$, $a \sqcap b$, $a \oplus b$ and $a \ominus b$ carry over from Chapter 3. We extend $\sqcup, \sqcap, +, -, \oplus, \ominus$ to valuations by interpreting them in pointwise fashion.

Metrics for the discounted quantitative μ-calculus. We call $d\mu^\alpha$ the discounted μ-calculus with all discount parameters $\leq \alpha$. We define the discounted metrics via an α-discounted metric transformer $H^\alpha_\preceq : \mathcal{M} \mapsto \mathcal{M}$, defined for all $d \in \mathcal{M}$ and all $s, t \in S$ by:

$$H^\alpha_{\preceq_1}(d)(s,t) \;=\; p(s,t) \sqcup \alpha \cdot \sup_{k \in C(d)} \left(\mathrm{Pre}_1(k)(s) - \mathrm{Pre}_1(k)(t)\right) . \tag{5.1}$$

Again, $H^\alpha_{\preceq_1}$ is continuous and monotonic in the lattice $(\mathcal{M}, \leq)$. The *α-discounted simulation metric* $[\preceq_1]^\alpha$ is the least fixpoint of $H^\alpha_{\preceq_1}$, and the *α-discounted bisimulation metric* $[\simeq_1]^\alpha$ is the least symmetrical fixpoint of $H^\alpha_{\preceq_1}$. The following result follows easily by induction on the Picard iterations used to compute the distances [dAHM03]; for all states $s, t \in S$ and a discount factor $\alpha \in [0, 1)$,

$$[s \preceq_1 t]^\alpha \leq [s \preceq_1 t] \qquad\qquad [s \simeq_1 t]^\alpha \leq [s \simeq_1 t] . \tag{5.2}$$

Using techniques similar to the undiscounted case, we can prove that for every game structure G and discount factor $\alpha \in [0, 1)$, the fixpoint $[\preceq_i]^\alpha$ is a directed metric and $[\simeq_i]^\alpha$ is a

metric, and that they are *reciprocal*, i.e., $[\preceq_1]^\alpha = [\succeq_2]^\alpha$ and $[\simeq_1]^\alpha = [\simeq_2]^\alpha$. Given the discounted bisimulation metric coincides for the two players, we write $[\simeq_g]^\alpha$ instead of $[\simeq_1]^\alpha$ and $[\simeq_2]^\alpha$. We now state without proof that the discounted μ-calculus provides a logical characterization of the discounted metric. The proof is based on induction on the structure of formulas, and closely follows the result for the undiscounted case. Let $d\mu^\alpha$ (respectively, $d\mu_1^{\alpha,+}$) consist of all discounted μ-calculus formulas (respectively, all discounted μ-calculus formulas with only the Pre_1 operator and all negations before atomic propositions). It follows that for all game structures G and states $s, t \in S$,

$$[s \preceq_1 t]^\alpha = \sup_{\varphi \in d\mu_1^{\alpha,+}} (\llbracket\varphi\rrbracket(s) - \llbracket\varphi\rrbracket(t)) \qquad [s \simeq_g t]^\alpha = \sup_{\varphi \in d\mu^\alpha} |\llbracket\varphi\rrbracket(s) - \llbracket\varphi\rrbracket(t)| \,. \tag{5.3}$$

Metric kernels. The kernel of the metric $[\simeq_g]$ ($[\simeq_g]^\alpha$) defines an equivalence relation $\simeq_g$ ($\simeq_g^\alpha$) on the states of a game structure: $s \simeq_g t$ ($s \simeq_g t)^\alpha$ iff $[s \simeq_g t] = 0$ ($[s \simeq_g t]^\alpha = 0$); the relation $\simeq_g$ is called the *game bisimulation* relation and the relation $\simeq_g^\alpha$ is called the *discounted game bisimulation* relation. 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 $[s \preceq_1 t] = 0$. The *discounted game simulation* preorder is defined analogously.

5.1 Bounds for Average and Discounted Payoff Games

In this chapter, we show 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.

5.1.1 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 $v_1^\alpha(s, \pi_1, \pi_2)$ 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

$$v_1^\alpha(s, \pi_1, \pi_2) = (1-\alpha) \cdot \sum_{n=0}^{\infty} \alpha^n \cdot \mathbb{E}_s^{\pi_1,\pi_2}\big([r](X_n)\big), \tag{5.4}$$

where X_n is a random variable representing the n-th state of the game. The discounted payoff for player 2 is defined as $v_2^\alpha(s, \pi_1, \pi_2) = -v_1^\alpha(s, \pi_1, \pi_2)$. 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:

$$w_i^\alpha(s) = \sup_{\pi_i \in \Pi_i} \inf_{\pi_{\sim i} \in \Pi_{\sim i}} v_i(s, \pi_i, \pi_{\sim i}) \; .$$

These values can be computed as the limit of the sequence of α-discounted, n-step rewards, for $n \to \infty$. For $i \in \{1,2\}$, we define a sequence of valuations $w_i^\alpha(0)(s)$, $w_i^\alpha(1)(s)$, $w_i^\alpha(2)(s)$, ... as follows: for all $s \in S$ and $n \geq 0$:

$$w_i^\alpha(n+1)(s) = (1-\alpha) \cdot [r](s) + \alpha \cdot \mathrm{Pre}_i(w_i^\alpha(n))(s) \; . \tag{5.5}$$

where the initial valuation $w_i^\alpha(0)$ is arbitrary. Shapley proved that $w_i^\alpha = \lim_{n\to\infty} w_i^\alpha(n)$ [Sha53].

5.1.2 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

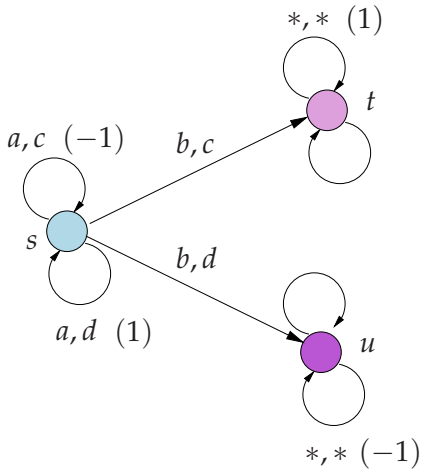
$$v_1(s, \pi_1, \pi_2) = \lim \inf_{n \to \infty} \frac{1}{n} \sum_{k=0}^{n-1} \mathbb{E}_s^{\pi_1, \pi_2}\big([r](X_k)\big), \tag{5.6}$$

where X_k is a random variable representing the k-th state of the game. The reward for player 2 is $v_2(s, \pi_1, \pi_2) = -v_1(s, \pi_1, \pi_2)$. 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 as

$$w_i(s) = \sup_{\pi_i \in \Pi_i} \inf_{\pi_{\sim i} \in \Pi_{\sim i}} v_i(s, \pi_i, \pi_{\sim i}) \ .$$

Mertens and Neyman established the determinacy of average reward 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 $\lim_{\alpha \to 1} w_i^\alpha(s) = w_i(s)$ [MN81]. It is easy to show that the average value of a game is a valuation. The average reward at a state is the average of the stream of expected rewards starting at that state. We present the following example to illustrate average reward values in games.

Example 7 (The Big Match) Consider the game in Figure 5.1. This game, called the Big Match, was introduced by Blackwell and Ferguson in [BF68], where a complete average reward analysis of this game can be found. Player 1 has moves $\{a, b\}$ and player 2 has moves $\{c, d\}$ in state s. The states t and u are absorbing. The reward for transitioning to state t for player 1 is 1 and the reward for transitioning to state u for player 1 is -1.



Payoffs at s	c	d
a	-1	1
b	1	-1

Figure 5.1: Example that illustrates the long-run average values of a game.

We analyze w_i, the player 1 average reward value of the game. The player 1 objective is to maximize the average reward value at every state, assuming the game starts in that state, while the player 2 objective is to minimize it. The immediate rewards at state s based on move choices are shown in the table in Figure 5.1. The stream of rewards for a game starting at state t is $1, 1, 1, \ldots$, giving us $w_i(n)(t) = \frac{1}{n}(n \cdot 1) = 1$, which in the limit yields 1; therefore $w_i(t) = 1$. Similarly the average reward at state u for player 1 is $w_i(u) = -1$. Blackwell and Ferguson in [BF68] show that in this game, there are no stationary optimal strategies for player 1 whereas a stationary strategy that chooses moves c and d with probability $\frac{1}{2}$ is player 2 optimal. Consider the game where player 2 has fixed his strategy to his optimal strategy. If player 1 chooses the pure strategy a at state s, then the game remains in state s and the immediate reward at each step is $\frac{1}{2} \cdot (-1) + \frac{1}{2} \cdot (1) = 0$. If player 1 chooses move a with probability λ and move b with probability $(1 - \lambda)$, with $\lambda \in [0, 1)$, then the game transitions to either state t or u at some step n with equal probability of $\lambda^n \cdot (1 - \lambda) \cdot \frac{1}{2}$. There are exactly two paths in this game, $s^+ t^\omega$

and s^+u^ω, each occurring with equal probability. Therefore, for all strategies of player 1, given player 2 plays his optimal strategy, the average reward after n steps at state s is $\lambda^n \cdot \frac{1}{2} \cdot 1 + \lambda^n \cdot \frac{1}{2} \cdot (-1) + \lambda^{n-1} \cdot (1-\lambda) \cdot \frac{1}{2} \cdot 1 + \lambda^{n-1} \cdot (1-\lambda) \cdot \frac{1}{2} \cdot (-1) = 0$, which in the limit is 0. Therefore, $w_i(s) = 0$, $w_i(t) = 1$ and $w_i(u) = -1$ in this game. ■

5.1.3 Metrics for Discounted and Average Payoffs

We show that the game simulation metric $[\preceq_1]$ provides a bound for discounted and long-run rewards. The discounted metric $[\preceq_1]^\alpha$ on the other hand does not provide such a bound as the following example shows.

Figure 5.2: Example that shows that the discounted metric may not be an upper bound for the difference in the discounted value across states.

Example 8 Consider a game consisting of four states s, t, s', t', and a variable r, with $[r](s) = 2$, $[r](s') = 2.1$, $[r](t) = 5$, and $[r](t') = 8$ as shown in Figure 5.2. All players have only one move at each state, and the transition relation is deterministic. Consider a discount factor $\alpha = 0.9$. The 0.9-discounted metric distance between states s' and s, is $[s' \simeq_g s]^{0.9} = 0.9 \cdot (8-5) = 2.7$. For the difference in discounted values between the states we proceed as follows. Using formulation 5.5, taking $w^\alpha(0)(t) = 5$, since state t is absorbing, we get $w^\alpha(1)(t) = (1-0.9) \cdot 5 + 0.9 \cdot 5 = 5$ which leads to $w^\alpha(n)(t) = 5$ for all $n \geq 0$. Similarly $w^\alpha(n)(t') = 8$ for all $n \geq 0$. Therefore, the difference in discounted values between s and s', again using 5.5, is given by: $w^\alpha(s') - w^\alpha(s) = (1-0.9) \cdot (2.1-2) + 0.9 \cdot (8-5) = 2.71$.

In the following we consider player 1 rewards (the case for player 2 is identical).

Theorem 15 *The following assertions hold.*

1. *For all game structures G, α-discounted rewards w_1^α, for all states $s, t \in S$, we have, (a) $w_1^\alpha(s) - w_1^\alpha(t) \leq [s \preceq_1 t]$ and (b) $|w_1^\alpha(s) - w_1^\alpha(t)| \leq [s \simeq_g t]$.*

2. *There exists a game structure G, states $s, t \in S$, such that for all α-discounted rewards w_1^α, $w_1^\alpha(t) - w_1^\alpha(s) > [t \simeq_g s]^\alpha$.*

Proof. We first prove assertion (1)(a). As the metric can be computed via Picard iteration, we have for all $n \geq 0$:

$$[s \preceq_1^n t] = p(s,t) \sqcup \sup_{k \in C([\preceq_1^{n-1}])} (\mathrm{Pre}_1(k)(s) - \mathrm{Pre}_1(k)(t)), \tag{5.7}$$

where X_j is a random variable representing the j-th state of the game. We prove by induction on $n \geq 0$ that $w_1^\alpha(n)(s) - w_1^\alpha(n)(t) \leq [s \preceq_1^n t]$. For all $s \in S$, taking $w_1^\alpha(0)(s) = [r](s)$, the base case follows. Assume the result holds for $n - 1 \geq 0$. We have:

$$\begin{aligned} w_1^\alpha(n)(s) - w_1^\alpha(n)(t) &= (1-\alpha) \cdot [r](s) + \alpha \cdot \mathrm{Pre}_1(w^\alpha(n-1))(s) - \\ &\qquad (1-\alpha) \cdot [r](t) \ - \alpha \cdot \mathrm{Pre}_1(w^\alpha(n-1))(t) \\ &= (1-\alpha) \cdot \big([r](s) - [r](t)\big) + \\ &\qquad \alpha \cdot \big(\mathrm{Pre}_1(w^\alpha(n-1))(s) - \mathrm{Pre}_1(w^\alpha(n-1))(t)\big) \\ &\leq (1-\alpha) \cdot p(s,t) + \alpha \cdot [s \preceq_1^n t] \ \leq \ [s \preceq_1^n t], \end{aligned}$$

where the last step follows by (5.7), since by the induction hypothesis we have $w_1^\alpha(n-1) \in C([\preceq_1^{n-1}])$. This proves assertion (1)(a). Given (1)(a), from the definition of $[s \simeq_g t] = [s \preceq_1 t] \sqcup [t \preceq_1 s]$, (1)(b) follows.

The example shown in Figure 5.2 proves the second assertion. ■

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

Corollary 2 *For all game structures G and states s and t, we have (a) $w(s) - w(t) \leq [s \preceq_1 t]$ and (b) $|w(s) - w(t)| \leq [s \simeq_g t]$.*

5.1.4 Metrics for Total Rewards

The *total reward* $v_1^T(s, \pi_1, \pi_2)$ for player 1 at a state s for a variable $r \in \mathcal{V}$ and the strategies $\pi_1 \in \Pi_1$ and $\pi_2 \in \Pi_2$ is defined as [FV97]:

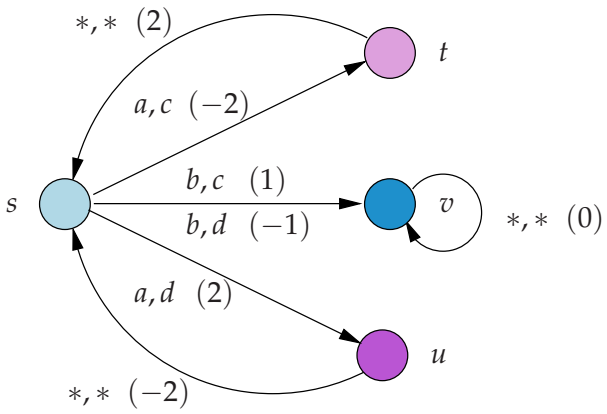
$$v_1^T(s, \pi_1, \pi_2) = \lim \inf_{n \to \infty} \frac{1}{n} \sum_{k=0}^{n-1} \sum_{j=0}^{k} \mathbb{E}_s^{\pi_1, \pi_2}\big([r](X_j)\big) \; . \tag{5.8}$$

The payoff $v_2^T(s, \pi_1, \pi_2)$ for player 2 is defined by replacing $[r]$ with $-[r]$ in (5.8). The *total-reward value* of the game G at s for player $i \in \{1, 2\}$ is defined analogously to the average value, via,

$$w_i^T(s) = \sup_{\pi_i \in \Pi_i} \inf_{\pi_{\sim i} \in \Pi_{\sim i}} v_i^T(s, \pi_1, \pi_2) \; .$$

The total reward at a state is the Cesàro average of the stream of expected rewards starting at that state; it is the average of the partial sums of expected rewards starting at that state. We illustrate the total reward values in game structures using the following example.

Example 9 (The Bad Match) Consider the game in Figure 5.3, called the Bad Match [FV97]. In the Big Match, the total reward value does not exist for player 1 at state s whereas in the Bad Match the total reward values exist at all states. Consider the Big Match



Payoffs at s	c	d
a	-2	2
b	1	-1

Figure 5.3: Example that illustrates the total reward values of a game.

and the player 1 total reward objective, $\sup_{\pi_i \in \Pi_i} \inf_{\pi_{\sim i} \in \Pi_{\sim i}} v_i^T(s, \pi_1, \pi_2)$, where player 1 chooses her move before player 2. Suppose player 1 chooses move a at s, then by choosing move c, player 2 forces the game to remain in state s while ensuring an endless stream of player 1 payoffs of -1, which is player 2's gain of 1. Suppose player 1 plays a with probability λ and b with probability $(1-\lambda)$ for $\lambda \in [0,1)$. If player 2 now chooses move d, then the game will eventually transition to state u, generating a finite stream of player 1 payoffs of 1 initially followed by an endless stream of player 1 payoffs of -1. Therefore, $\sup_{\pi_i \in \Pi_i} \inf_{\pi_{\sim i} \in \Pi_{\sim i}} v_i^T(s, \pi_1, \pi_2) = -\infty$. On the other hand, if player 2 plays first, then by choosing c and d with probability $\frac{1}{2}$, he ensures an immediate expected payoff of 0 at every step of the game, like in the average reward analysis of the Big Match. We have:

$$-\infty = \sup_{\pi_i \in \Pi_i} \inf_{\pi_{\sim i} \in \Pi_{\sim i}} v_i^T(s, \pi_1, \pi_2) \neq \inf_{\pi_{\sim i} \in \Pi_{\sim i}} \sup_{\pi_i \in \Pi_i} v_i^T(s, \pi_1, \pi_2) = 0 \,.$$

Therefore, the total reward does not exist in state s for player 1. Now consider the Bad Match, shown in Figure 5.3, where player 1 has moves $\{a, b\}$ and player 2 has moves $\{c, d\}$ in state s. The state v is absorbing. Every transition to state t is followed by a transition

back to state s and similarly for state u. The rewards for various transitions are shown in the figure. We analyze w_i^T, the player 1 total reward value of the game. Similar to the analysis of the Big Match, it is easy to check that the average reward values are 0 at all states of the game in Figure 5.3. Moreover, it has been shown that Player 2 has a stationary optimal strategy, which is to choose moves c and d with probability $\frac{1}{2}$ [FV97] whereas player 1 has no stationary optimal strategies. The total reward for either player at state v is 0. From state s, a transition to state t results in a player 1 payoff of -2, which is regained in the subsequent transition from t to s. Similarly, a transition to state u results in a player 1 payoff of 2, which is relinquished in the subsequent transition from u to s. Consider the game where player 2 has fixed his strategy to his optimal strategy. For a game that starts in state s, if player 1 chooses move a at all times, then the paths $(st)^+$ and $(su)^+$ occur with equal probability. It is easy to check that the total expected reward for every step of the game at state s in this case is 0, which in the limit gives a total reward value of 0. If player 1 chooses move a with probability λ and move b with probability $(1-\lambda)$, for $\lambda \in [0,1)$, given player 2 plays his optimal strategy, the set of paths is characterized by $(s[t|u])^+v^\omega$; the game eventually transitions to state v which is absorbing, yielding a stream of payoffs that has a finite non-zero prefix followed by an endless stream of 0 payoffs, which in the limit gives $w_i^T(s) = 0$. Given $w_i^T(s) = 0$, we have $w_i^T(t) = 2$, which is the reward for starting at state t and similarly $w_i^T(u) = -2$, which is the reward for starting at state u. ■

While the game simulation metric $[\simeq_g]$ provides an upper bound for the difference in discounted reward across states, as well as for the difference in average reward across states, it does not provide a bound for the difference in total reward. We now in-

troduce a new metric, the *total reward metric*, $[\bowtie_g]$, which provides such a bound. For a discount factor $\alpha \in [0,1)$, we define a metric transformer $H^{\alpha}_{\trianglelefteq_1} : \mathcal{M} \mapsto \mathcal{M}$ as follows. For all $d \in \mathcal{M}$ and $s, t \in S$, we let:

$$H^{\alpha}_{\trianglelefteq_1}(d)(s,t) = p(s,t) + \alpha \cdot \sup_{k \in C(d)} \left(\mathrm{Pre}_1(k)(s) - \mathrm{Pre}_1(k)(t) \right) . \tag{5.9}$$

The metric $[\trianglelefteq_1]^{\alpha}$ (resp. $[\bowtie_1]^{\alpha}$) is obtained as the least (resp. least symmetrical) fixpoint of (5.9). We write $[\trianglelefteq_1]$ for $[\trianglelefteq_1]^1$, and $[\bowtie_1]$ for $[\bowtie_1]^1$. These metrics are *reciprocal*, i.e., $[\trianglelefteq_1]^{\alpha} = [\trianglerighteq_2]^{\alpha}$ and $[\bowtie_1]^{\alpha} = [\bowtie_2]^{\alpha}$. If $\alpha < 1$ we get the *discounted total reward metric* and if $\alpha = 1$ we get the *undiscounted total reward metric*. While the discounted total reward metric is bounded, the undiscounted total reward metric may not be bounded. The total metrics provide bounds for the difference in discounted, average, and total reward between states.

Theorem 16 *The following assertions hold.*

1. *For all game structures G, for all discount factors $\alpha \in [0,1)$, for all states $s, t \in S$,*

   (a) $[s \trianglelefteq_1 t]^{\alpha} \leq (\theta_2 - \theta_1)/(1-\alpha)$, (b) $[s \trianglelefteq_1 t]^{\alpha} \leq [s \trianglelefteq_1 t]$,

   (c) $w_1^{\alpha}(s) - w_1^{\alpha}(t) \leq [s \trianglelefteq_1 t]^{\alpha}$, (d) $w_1(s) - w_1(t) \leq [s \trianglelefteq_1 t]$,

   (e) $w_1^{T}(s) - w_1^{T}(t) \leq [s \trianglelefteq_1 t]$.

2. *There exists a game structure G and states $s, t \in S$ such that, $[s \trianglelefteq_1 t] = \infty$.*

Proof. For assertion (1)(a), notice that $p(s,t) \leq (\theta_2 - \theta_1)$. Consider the n-step Picard iterate towards the metric distance. We have,

$$[s \trianglelefteq_1^n t]^{\alpha} \leq \sum_{i=0}^{n} \alpha^i \cdot (\theta_2 - \theta_1) .$$

In the limit this yields $[s \trianglelefteq_1 t]^\alpha \leq (\theta_2 - \theta_1)/(1-\alpha)$. Assertion (1)(b) follows by induction on the Picard iterations that realize the metric distance. For all $n \geq 0$, $[s \trianglelefteq_1^n t]^\alpha \leq [s \trianglelefteq_1^n t]$. Assertion (1)(c) follows by the definition of the discounted total reward metric where we have replaced the $\sqcup$ with a $+$. By induction, for all $n \geq 0$, from the proof of Theorem 15 we have,

$$w_1^\alpha(n)(s) - w_1^\alpha(n)(t) \leq$$

$$(1-\alpha) \cdot p(s,t) + \alpha \cdot (\mathrm{Pre}_1(w_1^\alpha(n-1))(s) - \mathrm{Pre}_1(w_1^\alpha(n-1))(t)) \leq$$

$$[s \trianglelefteq_1^n t]^\alpha \ .$$

For assertion (1)(d), towards an inductive argument on the Picard iterates that realize the metric, for all $n \geq 0$, we have $[s \preceq_1^n t] \leq [s \trianglelefteq_1^n t]$, which in the limit gives $[s \preceq_1 t] \leq [s \trianglelefteq_1 t]$. This leads to $w_1(s) - w_1(t) \leq [s \trianglelefteq_1 t]$, using Corollary 2. This proves assertion (1)(d). We now prove assertion (1)(e) by induction and show that for all $n \geq 0$, $w_1^T(n)(s) - w_1^T(n)(t) \leq [s \trianglelefteq_1^n t]$. As the metric can be computed via Picard iteration, we have for all $n \geq 0$:

$$[s \trianglelefteq_1^n t] = p(s,t) + \sup_{k \in C([\trianglelefteq_1^{n-1}])} (\mathrm{Pre}_1(k)(s) - \mathrm{Pre}_1(k)(t)) \ . \tag{5.10}$$

We define a valuation transformer $u : \mathcal{F} \mapsto \mathcal{F}$ as $u(0) = [r]$ and for all $n > 0$ and state $s \in S$ as,

$$u(n)(s) = [r](s) + \mathrm{Pre}_1(u(n-1))(s)$$

We take $w_1^T(0) = u(0) = [r]$ and for $n > 0$, from the definition of total rewards (5.8), we get the n-step total reward value at a state $s \in S$ in terms of u as,

$$w_1^T(n)(s) = \frac{1}{n} \cdot \sum_{i=1}^{n} u(i)(s) \ .$$

Notice that $w_1^T(n)(s) \leq u(n)$ for all $n \geq 0$. When $n = 0$, the result is immediate by the definition of $w_1^T(0)$, noticing that $[s \trianglelefteq_1^0 t] = p(s,t)$. Assume the result holds for $n-1 \geq 0$. We have:

$$\begin{aligned} w_1^T(n)(s) - w_1^T(n)(t) &= \frac{1}{n} \cdot \sum_{i=1}^{n} u(i)(s) - \frac{1}{n} \cdot \sum_{i=1}^{n} u(i)(t) \\ &= \frac{1}{n} \cdot \sum_{i=1}^{n} (u(i)(s) - u(i)(t)) \\ &= \frac{1}{n} \cdot \sum_{i=1}^{n} (([r](s) - [r](t)) + \\ &\qquad (\mathrm{Pre}_1(u(i-1))(s) - \mathrm{Pre}_1(u(i-1))(t))) \qquad (5.11) \\ &\leq \frac{1}{n} \cdot \sum_{i=1}^{n} [s \trianglelefteq_1^i t] \qquad (5.12) \\ &\leq [s \trianglelefteq_1^n t], \qquad (5.13) \end{aligned}$$

where (5.12) follows from (5.11) by (5.10), since by our induction hypothesis we have $w_1^T(i) \leq u(i) \in C([\trianglelefteq_1^i])$ for all $0 \leq i < n$ and (5.13) follows from (5.12) from the monotonicity of the undiscounted total reward metric. To prove assertion (2), consider the game structure on the left hand side in Figure 5.2. The total reward at state s is unbounded; $w_1^T(s) = 2 + 5 + \ldots = \infty$ Now consider a modified version of the game, with identical structure and with states s' and t' corresponding to s and t of the original game. Let $[r](t') = 0$. In the modified game, $w_1^T(s') = 2$. From result (1)(e), since $w_1^T(s) = \infty$ and $w_1^T(s') = 2$, we have $[s \trianglelefteq_1 s'] = \infty$. ■

It is a very simple observation that the quantitative μ-calculus does not provide a logical characterization for $[\trianglelefteq_1^\alpha]$ or $[\trianglelefteq_1]$. In fact, all formulas of the quantitative μ-calculus have valuations in the interval $[\theta_1, \theta_2]$, while as stated in Theorem 16, the total reward can be unbounded. The difference is essentially due to the fact that our version of the

quantitative μ-calculus lacks a "+" operator. It is not clear how to introduce such a + operator in a context sufficiently restricted to provide a logical characterization for $[\trianglelefteq_1^\alpha]$; above all, it is not clear whether a canonical calculus, with interesting formal properties, would be obtained.

5.1.5 Metric Kernels

We now show that the kernels of all the metrics defined in the paper coincide: an algorithm developed for the game kernels $\preceq_1$ and $\simeq_g$, compute the kernels of the corresponding discounted and total reward metrics as well.

Theorem 17 *For all game structures G, states s and t, all discount factors $\alpha \in [0,1)$, the following statements are equivalent:*

$$(a)\ [s \preceq_1 t] = 0 \qquad (b)\ [s \preceq_1 t]^\alpha = 0 \qquad (c)\ [s \trianglelefteq_1 t]^\alpha = 0\,.$$

Proof. We prove $(a) \Rightarrow (b) \Rightarrow (c) \Rightarrow (a)$. We assume $0 < \alpha < 1$. Assertion (a) implies that $p(s,t) = 0$ and $\sup_{k \in C([\preceq_1])}(\mathrm{Pre}_1(k)(s) - \mathrm{Pre}_1(k)(t)) \leq 0$; Since $C([\preceq_1]^\alpha) \subseteq C([\preceq_1])$ from (5.2), (b) follows. We prove $(b) \Rightarrow (c)$ by induction on the Picard iterations that compute $[s \preceq_1 t]^\alpha$ and $[s \trianglelefteq_1 t]^\alpha$. The base case is immediate. Assume that for all states s and t, $[s \preceq_1^{n-1} t]^\alpha = 0$ implies $[s \trianglelefteq_1^{n-1} t]^\alpha = 0$. Towards a contradiction, assume $[s \preceq_1^n t]^\alpha = 0$ but $[s \trianglelefteq_1^n t]^\alpha > 0$. Then there must be $k \in C([\trianglelefteq_1^{n-1}]^\alpha)$ such that $\mathrm{Pre}_1(k)(s) - \mathrm{Pre}_1(k)(t) > 0$. By our induction hypothesis, there exists a $\delta > 0$ such that $k' = \delta \cdot k \in C([\preceq_1^{n-1}]^\alpha)$. Since Pre is multi-linear, the player optimal responses in $\mathrm{Pre}_1(k)(s)$ remain optimal for k'. But this means $(\mathrm{Pre}_1(k')(s) - \mathrm{Pre}_1(k')(t)) > 0$ for $k' \in C([\preceq_1^{n-1}]^\alpha)$, leading to $[s \preceq^n t]^\alpha > 0$; a contradiction. Therefore, $(b) \Rightarrow (c)$. In a similar fashion we can show that $(c) \Rightarrow (a)$. ■

Chapter 6

Algorithms

In this chapter we present algorithms for computing the metrics and the kernels of the metrics, which are the simulation and bisimulation relations, for MDPs, turn-based games and concurrent games. We begin with turn-based games and MDPs.

6.1 Algorithms for 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 [vBSW08]. Finally, we present improved algorithms for the important case of the kernel of the metrics. Since by Theorem 17 the kernels of the metrics introduced in this paper coincide, we present

our algorithms for the kernel of the undiscounted metric. For the bisimulation kernel our algorithm is significantly more efficient compared to previous algorithms.

6.1.1 Algorithms for the Metrics

For turn-based games and 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, $\sup_{k \in C(d)} \big(\mathrm{Pre}_1(k)(s) - \mathrm{Pre}_1(k)(t)\big)$, which is the same as, $\sup_{k \in C(d)} \sup_{x \in \mathcal{D}_1(s)} \inf_{y \in \mathcal{D}_1(t)} (\mathbb{E}^x_s(k) - \mathbb{E}^y_t(k))$, since $\mathrm{Pre}_1(k)(s) = \sup_{x \in \mathcal{D}_1(s)}(\mathbb{E}^x_s(k))$ and $\mathrm{Pre}_1(k)(t) = \sup_{y \in \mathcal{D}_1(t)}(\mathbb{E}^y_t(k))$ as player 1 is the only player with a choice of moves at state s. By expanding the expectations, we get the following form,

$$\sup_{k \in C(d)} \sup_{x \in \mathcal{D}_1(s)} \inf_{y \in \mathcal{D}_1(t)} \Bigg(\sum_{u \in S} \sum_{a \in \Gamma_1(s)} \delta(s,a)(u) \cdot x(a) \cdot k(u) - \sum_{v \in S} \sum_{b \in \Gamma_1(t)} \delta(t,b)(v) \cdot y(b) \cdot k(v) \Bigg) . \quad (6.1)$$

We observe that the one-step distance as defined in (6.1) is a *sup-inf non-linear (quadratic)* optimization problem. We now present two lemmas by which we transform (6.1) to an *inf linear* optimization problem, which we solve by linear programming (LP). The first lemma reduces (6.1) to an equivalent formulation that considers only pure moves at state s. The second lemma further reduces (6.1), using duality, to a formulation that can be solved using LP.

Lemma 7 *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,*

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

Proof. We prove the result for player 1 states s and t, with the proof being identical for player 2. Given a metric $d \in \mathcal{M}$, we have,

$$\begin{aligned}
\sup_{k \in C(d)} \sup_{x \in \mathcal{D}_1(s)} \inf_{y \in \mathcal{D}_1(t)} (\mathbb{E}_s^x(k) - \mathbb{E}_t^y(k)) &= \sup_{k \in C(d)} \Big(\sup_{x \in \mathcal{D}_1(s)} \mathbb{E}_s^x(k) - \sup_{y \in \mathcal{D}_1(t)} \mathbb{E}_t^y(k) \Big) \\
&= \sup_{k \in C(d)} \Big(\sup_{a \in \Gamma_1(s)} \mathbb{E}_s^a(k) - \sup_{y \in \mathcal{D}_1(t)} \mathbb{E}_t^y(k) \Big) \qquad (6.2) \\
&= \sup_{k \in C(d)} \sup_{a \in \Gamma_1(s)} \inf_{y \in \mathcal{D}_1(t)} (\mathbb{E}_s^a(k) - \mathbb{E}_t^y(k)) \\
&= \sup_{a \in \Gamma_1(s)} \sup_{k \in C(d)} \inf_{y \in \mathcal{D}_1(t)} (\mathbb{E}_s^a(k) - \mathbb{E}_t^y(k)) \qquad (6.3) \\
&= \sup_{a \in \Gamma_1(s)} \inf_{y \in \mathcal{D}_1(t)} \sup_{k \in C(d)} (\mathbb{E}_s^a(k) - \mathbb{E}_t^y(k)) \qquad (6.4)
\end{aligned}$$

For a fixed $k \in C(d)$, since pure optimal strategies exist at each state for turn-based games and MDPs, we replace the $\sup_{x \in \mathcal{D}_1(s)}$ with $\sup_{a \in \Gamma_1(s)}$ yielding (6.2). Since the difference in expectations is multi-linear, $y \in \mathcal{D}_1(t)$ is a probability distribution and $C(d)$ is a compact convex set, we can use the generalized minimax theorem [Sio58], and interchange the innermost sup inf to get (6.4) from (6.3). ■

The proof of Lemma 7 is illustrated using the following example.

Example 10 Consider the example in Figure 6.1. In the MDPs shown in the figure, every move leads to a unique successor state, with the exception of move $e \in \Gamma_1(s)$, which leads to states u and v with equal probability. Assume the variable valuations are such that

(a) MDP 1 (b) MDP 2

Figure 6.1: An example illustrating the proof of Lemma 7.

all states are at a propositional distance of 1. Without loss of generality, assume that the valuation $k \in C(d)$ is such that $k(u) > k(v)$. By the linearity of expectations, for move $c \in \Gamma_1(s)$, $\mathbb{E}^c_s(k) > \mathbb{E}^x_s(k)$ for all $x \in \mathcal{D}_1(s)$. Similar arguments can be made for $k(u) < k(v)$. This gives an informal justification for step (6.2) in the proof; given a $k \in C(d)$, there exist pure optimal strategies for the single player with a choice of moves at each state. While we can use pure moves at states s and t *if* $k \in C(d)$ *is known*, the principle difficulty in directly computing the left hand side of the equality arises from the uncountably many values for k; the distance is the supremum over all possible values of k. In the final equality, step (6.4), and hence by this theorem, we have avoided this difficulty, by showing an equivalent expression that picks a $k \in C(d)$ to show the difference in distributions induced over states. As we shall see, this enables computing the one-step metric distance using a trans-shipping formulation. We remark that while we can use pure moves at state s, we cannot do so at state t in the right hand side of step (6.4) of the proof. Firstly, the proof of the theorem depends on $y \in \mathcal{D}_1(t)$ being convex. Secondly, if we could restrict our attention to pure moves at state t, then we can replace $\inf_{y \in \mathcal{D}_1(t)}$ with $\inf_{f \in \Gamma_1(t)}$ on the right hand side. But this yields too fine a one-step distance. Consider move e at state s. We see that neither c nor

b at state t yield distributions over states that match the distribution induced by e. We can then always pick $k \in C(d)$ such that $\mathbb{E}_s^e(k) - \mathbb{E}_t^f(k) > 0$. If we choose $y \in \mathcal{D}_1(t)$ such that $y(b) = y(c) = \frac{1}{2}$, we match the distribution induced by move e from state s, which implies that for any choice of $k \in C(d)$, $\mathbb{E}_s^e(k) - \mathbb{E}_t^{y(b)=y(c)=\frac{1}{2}}(k) = 0$. Intuitively, the right hand side of the equality can be interpreted as a game between a protagonist and an antagonist, with the protagonist picking $y \in \mathcal{D}_1(t)$, for every pure move $a \in \Gamma_1(s)$, to match the induced distributions over states. The antagonist then picks a $k \in C(d)$ to maximize the difference in induced distributions. If the distributions match, then no choice of $k \in C(d)$ yields a difference in expectations bounded away from 0. ■

From Lemma 7, given $d \in \mathcal{M}$, we can write the player 1 one-step distance between states s and t as follows,

$$\mathsf{OneStep}(s,t,d) = \sup_{a \in \Gamma_1(s)} \inf_{y \in \mathcal{D}_1(t)} \sup_{k \in C(d)} \left(\mathbb{E}_s^a(k) - \mathbb{E}_t^y(k)\right). \tag{6.5}$$

Hence we compute for all $a \in \Gamma_1(s)$, the expression,

$$\mathsf{OneStep}(s,t,d,a) = \inf_{y \in \mathcal{D}_1(t)} \sup_{k \in C(d)} \left(\mathbb{E}_s^a(k) - \mathbb{E}_t^y(k)\right),$$

and then choose the maximum, i.e., $\max_{a \in \Gamma_1(s)} \mathsf{OneStep}(s,t,d,a)$. We now present a lemma that helps reduce the above $\inf - \sup$ optimization problem to a linear program. We first introduce some notation. We denote by λ the set of variables $\lambda_{u,v}$, for $u, v \in S$. Given $a \in \Gamma_1(s)$, and a distribution $y \in \mathcal{D}_1(t)$, we write $\lambda \in \Phi(s,t,a,y)$ if the following linear constraints are satisfied:

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

$$\text{(3) for all } u, v \in S : \lambda_{u,v} \geq 0\,.$$

Lemma 8 *For all turn-based game structures and MDPs G, for all $d \in \mathcal{M}$, and for all $s, t \in S$, the following assertion holds:*

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

Proof. Since duality always holds in LP, from the LP duality based results of [vBW01a], for all $a \in \Gamma_1(s)$ and $y \in \mathcal{D}_1(t)$, the maximization over all $k \in C(d)$ can be re-written as a minimization problem as follows:

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

The formula on the right hand side of the above equality is the *trans-shipping formulation*, which solves for the minimum cost of shipping the distribution $\delta(s,a)$ into $\delta(t,y)$, with edge costs d. The result of the lemma follows. ■

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

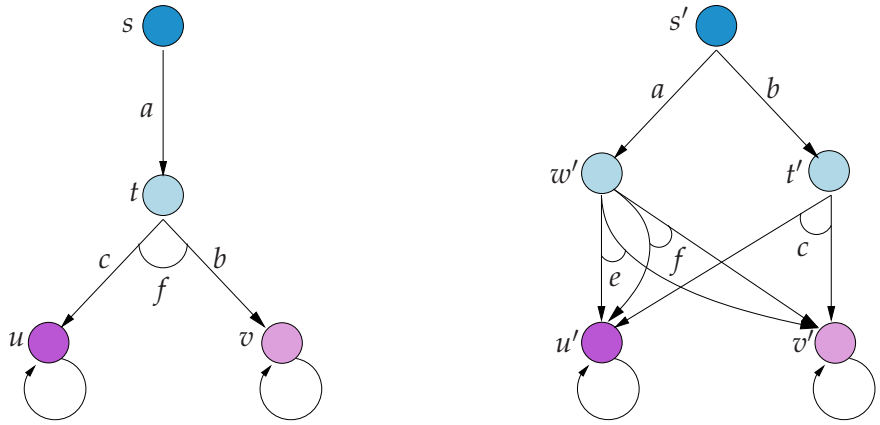
$$\text{Minimize} \quad \sum_{u,v \in S} d(u,v) \cdot \lambda_{u,v} \quad \text{subject to} \tag{6.6}$$

$$\text{(1) for all } v \in S : \sum_{u \in S} \lambda_{u,v} = \delta(s,a)(v); \qquad \text{(2) for all } u \in S : \sum_{v \in S} \lambda_{u,v} = \sum_{b \in \Gamma_1(t)} y_b \cdot \delta(t,b)(u);$$

$$\text{(3) for all } u,v \in S : \lambda_{u,v} \geq 0; \qquad \text{(4) for all } b \in \Gamma_1(t) : y_b \geq 0; \qquad \text{(5)} \sum_{b \in \Gamma_1(t)} y_b = 1 .$$

Example 11 We now use the MDPs in Figure 6.2(a) and 6.2(b) to compute the simulation distance between states using the results in Lemma 7 and Lemma 8. In the figure, states of the same color have a propositional distance of 0 and states of different colors have a propositional distance of 1; $p(s,s') = p(t,t') = p(u,u') = p(v,v') =$

$p(t', w') = 0$. In MDP 1, shown in Figure 6.2(a), $\delta(s, a)(t) = \delta(t, b)(v) = \delta(t, c)(u) = 1$ and $\delta(t, f)(u) = \delta(t, f)(v) = \frac{1}{2}$. In MDP 2, shown in Figure 6.2(b), $\delta(s', a)(w') = \delta(s', b)(t') = 1$, $\delta(t', c)(u') = \frac{1}{2} - \epsilon$, $\delta(t', c)(v') = \frac{1}{2} + \epsilon$, $\delta(w', e)(u') = \delta(w', f)(v') = 1 - \epsilon$ and $\delta(w', e)(v') = \delta(w', f)(u') = \epsilon$.



(a) MDP 1 (b) MDP 2

Figure 6.2: An example used to compute the simulation metric between states. States of the same color have a propositional distance of 0.

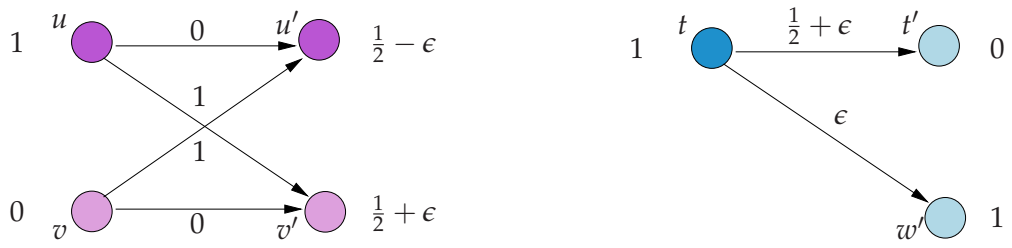
t	w'		t'	
$\Gamma_1(t)$	$x \in \mathcal{D}_1(w')$	*Cost*	$x \in \mathcal{D}_1(t')$	*Cost*
b	$x(f) = 1$	ϵ	$x(c) = 1$	$\frac{1}{2} - \epsilon$
c	$x(e) = 1$	ϵ	$x(c) = 1$	$\frac{1}{2} + \epsilon$
f	$x(f) = x(e) = \frac{1}{2}$	0	$x(c) = 1$	ϵ

Table 6.1: The moves from states w' and t' that minimize the trans-shipping cost for each $a \in \Gamma_1(t)$ and the corresponding costs.

In Table 6.2, we show the simulation metric distance between states of the MDPs in Figure 6.2(a) and Figure 6.2(b). Consider states t and t'. c is the only move available to

$[\preceq]$	s'	t'	w'	u'	v'
s	ϵ	1	1	1	1
t	1	$\frac{1}{2}+\epsilon$	ϵ	1	1
u	1	1	1	0	1
v	1	1	1	1	0

Table 6.2: The simulation metric distance between states in MDP 1 and states in MDP 2.



(a) $[t \preceq t'] = \frac{1}{2}+\epsilon$ (b) $[s \preceq s'] = \epsilon$

Figure 6.3: The trans-shipping formulation that gives the metric distances between states.

player 1 from state t' and it induces a transition probability of $\frac{1}{2}+\epsilon$ to state v' and $\frac{1}{2}-\epsilon$ to state u'. For the pure move c at state t, the induced transition probabilities and edge costs in the trans-shipping formulation are shown in Figure 6.3(a). It is easy to see that the trans-shipping cost in this case is $\frac{1}{2}+\epsilon$; shown in Table 6.1 along the row corresponding to move c from state t and column corresponding to state t'. Similarly, the trans-shipping cost for the moves b and f from state t are $\frac{1}{2}-\epsilon$ and ϵ respectively. The metric distance $[t \preceq t']$, which is the maximum over these trans-shipping costs is then $\frac{1}{2}+\epsilon$. Now consider the states t and w'. In Table 6.1, we show for each pure move $a \in \Gamma_1(t)$, the move $x \in \mathcal{D}_1(w')$ that minimizes the trans-shipping cost together with the minimum cost. In this case it is

easy to see that $[t \preceq w'] = \epsilon$. Given $[t \preceq t'] = \frac{1}{2} + \epsilon$ and $[t \preceq w'] = \epsilon$, we can calculate the distance $[s \preceq s']$ from the trans-shipping formulation shown in Figure 6.3(b); the minimum cost is ϵ that entails choosing move a from state s', giving us $[s \preceq s'] = \epsilon$. ■

Theorem 18 *For all turn-based game structures and MDPs G, given $d \in \mathcal{M}$, for all states $s, t \in S$, we can compute $H_{\preceq_1}(d)(s,t)$ in polynomial time by the Linear Program (6.6).*

For all states $s, t \in S$, iteration of $\mathsf{OneStep}(s,t,d)$ 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 $d(s,t) = \mathsf{OneStep}(s,t,d)$. We observe that since $H_{\preceq_1}$ is non-decreasing, we have $\mathsf{OneStep}(s,t,d) \geq d(s,t)$. It follows that the equality $d(s,t) = \mathsf{OneStep}(s,t,d)$ holds iff for every $a \in \Gamma_1(s)$, of which there are finitely many, all the linear inequalities of LP (6.6) are satisfied, and $d(s,t) = \sum_{u,v \in S} d(u,v) \cdot \lambda_{u,v}$ holds. It then follows that $d(s,t) = \mathsf{OneStep}(s,t,d)$ 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, in the fixpoint, $[s \preceq_1 t] \leq r$, as follows:

$$\exists d \in \mathcal{M}.[(\bigwedge_{u,v \in S} \mathsf{OneStep}(u,v,d) = d(u,v)) \wedge (d(s,t) \leq r)] \,.$$

If the formula $\Phi(r)$ is true, then there exists a fixpoint d, such that $d(s,t)$ is bounded by r, which implies that in the least fixpoint $d(s,t)$ is bounded by r. Conversely, if in the least fixpoint $d(s,t)$ 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 [Can88], we have the following result.

Theorem 19 (Decision complexity for exact distance). *For all turn-based game structures and MDPs G, given a rational r, and two states s and t, whether* $[s \preceq_1 t] \leq r$ *can be decided in PSPACE.*

Approximation. Given a rational $\epsilon > 0$, using binary search and $\mathcal{O}(\log(\frac{\theta_2-\theta_1}{\epsilon}))$ 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 3 (Approximation for exact distance). *For all turn-based game structures and MDPs 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 PSPACE.*

6.1.2 Algorithms for the Kernel

The kernel of the simulation metric $\preceq_1$ can be computed as the limit of the series $\preceq_1^0, \preceq_1^1, \preceq_1^2, \ldots$, 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 $\mathsf{OneStep}(s, t, 1_{\preceq_1^n}) = 0$. Checking the condition $\mathsf{OneStep}(s, t, 1_{\preceq_1^n}) = 0$, 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 (6.6). We note that this is the same LP feasibility problem that was introduced in [ZH07] as part of an algorithm to decide simulation of probabilistic systems in which each label may lead to one or more distributions over states.

For the bisimulation kernel, we present a more efficient algorithm, which also improves on the algorithms presented in [ZH07]. The idea is to proceed by partition refinement, as usual for bisimulation computations. 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 $\mathcal{Q}^0$, $\mathcal{Q}^1$, $\mathcal{Q}^2$, ..., 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 (4.11), $[s \simeq_g t] = 0$ iff $[s \preceq_1 t] = 0$ and $[t \preceq_1 s] = 0$, two states s, t in a given class of $\mathcal{Q}^n$ remain in the same class in $\mathcal{Q}^{n+1}$ iff both (s, t) and (t, s) satisfy the set of feasibility LP problems $\mathsf{OneStepBis}(s, t, \mathcal{Q}^n)$ as given below:

> $\mathsf{OneStepBis}(s, t, \mathcal{Q})$ 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:
>
> $$\text{(1) for all } b \in \Gamma(t) : \; x_b \geq 0, \qquad \text{(2)} \sum_{b \in \Gamma(t)} x_b = 1,$$
>
> $$\text{(3) for all } V \in \mathcal{Q} : \sum_{b \in \Gamma(t)} \sum_{u \in V} x_b \cdot \delta(t, b)(u) \geq \sum_{u \in V} \delta(s, a)(u) \; .$$

In the following theorem we show that two states $s, t \in S$ are $n+1$ step bisimilar iff $\mathsf{OneStepBis}(s, t, \mathcal{Q}^n)$ and $\mathsf{OneStepBis}(t, s, \mathcal{Q}^n)$ are feasible.

Theorem 20 *For all turn-based game structures and MDPs G, for all $n \geq 0$, given two states $s, t \in S$ and an n-step bisimulation partition of states $\mathcal{Q}^n$ such that $\forall V \in \mathcal{Q}^n$, $\forall u, v \in V$, $[u \simeq_g v]^n = 0$, the following holds,*

$$[s \simeq_g t]^{n+1} = 0 \textit{ iff } \mathsf{OneStepBis}(s, t, \mathcal{Q}^n) \textit{ and } \mathsf{OneStepBis}(t, s, \mathcal{Q}^n) \textit{ are both feasible.}$$

Proof. We proceed by induction on n. Assume the result holds for all iteration steps up to n and consider the case for $n+1$. In one direction, if $[s \simeq_g t]^{n+1} = 0$, then $[s \preceq_1 t]^{n+1} = [t \preceq_1 s]^{n+1} = 0$ by the definition of the bisimulation metric. We need to show that given $[s \preceq_1 t]^{n+1} = 0$, $\mathsf{OneStepBis}(s, t, \mathcal{Q}^n)$ is feasible. The proof is identical for $[t \preceq_1 s]^{n+1} = 0$. From the definition of the $n+1$ step simulation distance, given $p(s,t) = 0$ by our induction hypothesis, we have,

$$\forall b \in \Gamma_1(s) \inf_{x \in \mathcal{D}_1(t)} \sup_{k \in C(d^n)} (\mathbb{E}_s^b(k) - \mathbb{E}_t^x(k)) \leq 0\,. \tag{6.7}$$

Consider a player 1 move $a \in \Gamma_1(s)$. Since we can interchange the order of the inf and sup by the generalized minimax theorem in $\inf_{x \in \mathcal{D}_1(t)} \sup_{k \in C(d^n)} (\mathbb{E}_s^a(k) - \mathbb{E}_t^x(k))$, the optimal values of $x \in \mathcal{D}_1(t)$ and $k \in C(d^n)$ exist and only depend on a. Let x_a and k_a be the optimal values of x and k that realize the inf and sup in $\inf_{x \in \mathcal{D}_1(t)} \sup_{k \in C(d^n)} (\mathbb{E}_s^a(k) - \mathbb{E}_t^x(k))$. Using x_a and k_a in (6.7) we have:

$$\mathbb{E}_t^{x_a}(k_a) \geq \mathbb{E}_s^a(k_a)$$

$$\sum_{u \in S} \delta(t, x_a)(u) \cdot k_a(u) \geq \sum_{v \in S} \delta(s, a)(v) \cdot k_a(v)$$

$$\sum_{V \in \mathcal{Q}^n} \sum_{u \in V} \delta(t, x_a)(u) \cdot k_a(u) \geq \sum_{V \in \mathcal{Q}^n} \sum_{v \in V} \delta(s, a)(v) \cdot k_a(v) \tag{6.8}$$

$$\sum_{V \in \mathcal{Q}^n} \sum_{u \in V} \delta(t, x_a)(u) \geq \sum_{V \in \mathcal{Q}^n} \sum_{v \in V} \delta(s, a)(v) \tag{6.9}$$

$$\forall V \in \mathcal{Q}^n. \Big(\sum_{u \in V} \delta(t, x_a)(u) \geq \sum_{u \in V} \delta(s, a)(u) \Big), \tag{6.10}$$

where (6.9) follows from (6.8) by noting that for all $V \in \mathcal{Q}^n$, for all states $u, v \in V$, $d^n(u,v) = d^n(v,u) = 0$, by our hypothesis, leading to $k(u) - k(v) \leq d^n(u,v) = 0$ and $k(v) - k(u) \leq d^n(v,u) = 0$, which implies $k(u) = k(v)$ for all $k \in C(d^n)$. To show

(6.10) follows from (6.9), assume towards a contradiction that there exists a $V' \in \mathcal{Q}^n$ such that $\sum_{u \in V'} \delta(t, x_a)(u) < \sum_{u \in V'} \delta(s, a)(u)$. Then there must be a $V'' \in \mathcal{Q}^n$ such that $\sum_{u \in V''} \delta(t, x_a)(u) > \sum_{u \in V''} \delta(s, a)(u)$ since $\delta(t, x_a)$ is a probability distribution and the sum of the probability mass allocated to each equivalence class should be 1. Further, for all $V \in \mathcal{Q}^n$, for all $u, v \in V$, we have $d^n(u, v) = d^n(v, u) = 0$ and for all $u \in V$ and for all $w \in S \setminus V$, we have $d^n(u, w) = d^n(w, u) = 1$. Therefore, we can pick a feasible $k' \in C(d^n)$ such that $k'(v) > 0$ for all $v \in V''$ and $k'(v) = 0$ for all other states. Using k' we get $\mathbb{E}_s^a(k') - \mathbb{E}_t^{x_a}(k') > 0$ which means k_a is not optimal, contradicting (6.7).

In the other direction, assume that $\mathsf{OneStepBis}(s, t, \mathcal{Q}^n)$ is feasible. We need to show that $[s \preceq_1 t]^{n+1} = 0$. Since $\mathsf{OneStepBis}(s, t, \mathcal{Q}^n)$ is feasible, there exists a distribution $x_a \in \mathcal{D}_1(t)$ for all $a \in \Gamma_1(s)$ such that, $\forall V \in \mathcal{Q}^n.(\sum_{u \in V} \delta(t, x_a)(u) \geq \sum_{v \in V} \delta(s, a)(v))$. By our induction hypothesis, this implies that for all $k \in C(d^n)$, we have $(\mathbb{E}_s^a(k) - \mathbb{E}_t^{x_a}(k)) \leq 0$ and in particular $\sup_{k \in C(d^n)}(\mathbb{E}_s^a(k) - \mathbb{E}_t^{x_a}(k)) \leq 0$. Since $p(s, t) = 0$ by our hypothesis and we have shown,

$$\forall a \in \Gamma_1(s) \inf_{x \in \mathcal{D}_1(t)} \sup_{k \in C(d^n)} (\mathbb{E}_s^a(k) - \mathbb{E}_t^x(k)) \leq 0,$$

we have, from Lemma 7,

$$[s \preceq_1 t]^{n+1} = p(s, t) \sqcup \sup_{a \in \Gamma_1(s)} \inf_{x \in \mathcal{D}_1(t)} \sup_{k \in C(d^n)} (\mathbb{E}_s^a(k) - \mathbb{E}_t^x(k)) = 0.$$

In a similar fashion, if $\mathsf{OneStepBis}(t, s, \mathcal{Q}^n)$ is feasible then $[t \preceq_1 s]^{n+1} = 0$, which leads to $[s \simeq_g t]^{n+1} = 0$ by the definition of the bisimulation metric, as required. ■

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 needs to be solved. Let us denote by $\mathsf{LPF}(n,m)$ the complexity of solving the feasibility of m linear inequalities over n variables. We obtain the following result.

Theorem 21 *For all turn-based game structures and MDPs G, the following assertions hold:*

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

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

Remark 1 The best known algorithm for $\mathsf{LPF}(n,m)$ works in time $\mathcal{O}(n^{2.5} \cdot \log(n))$ [Ye06] (assuming each arithmetic operation takes unit time). The previous algorithm for the bisimulation kernel checked two way simulation and hence has the complexity $\mathcal{O}(n^4 \cdot m \cdot (n^2+m)^{2.5} \cdot \log(n^2+m))$, whereas our algorithm works in time $\mathcal{O}(n^4 \cdot m \cdot m^{2.5} \cdot \log(m))$. For most practical purposes, the number of moves at a state is constant (i.e., m is constant). For the case when m is constant, the previous best known algorithm worked in $\mathcal{O}(n^9 \cdot \log(n))$ time, whereas our algorithm works in time $\mathcal{O}(n^4)$.

6.2 Algorithms for Concurrent Games

In this section we first show 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 [EY07, GGJ76]. Next, we present algorithms based on a decision procedure for the theory

of real closed fields, for both checking the bounds of the exact distance and the kernel of the metrics.

6.2.1 Reduction of Reachability Games to Metrics

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 22 *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*

$$[s \preceq_1 t'] = \sup_{\pi_1 \in \Pi_1} \inf_{\pi_2 \in \Pi_2} \Pr_s^{\pi_1,\pi_2}(\Diamond q) \,.$$

Proof. The concurrent game structure G' is obtained from G by adding an absorbing state t'. The states that are not labeled by q, and the additional state t', are labeled by its complement $\neg q$. Observe there is only one proposition sequence from t', and it is $(\neg q)^\omega$. To prove the desired claim we show that for all $s \in S$ we have $[s \preceq_1 t'] = \sup_{\pi_1 \in \Pi_1} \inf_{\pi_2 \in \Pi_2} \Pr_s^{\pi_1,\pi_2}(\Diamond q)$. From a state s in G the possible proposition sequences can be expressed as the following ω-regular expression: $(\neg q)^\omega \cup (\neg q)^* \cdot q^\omega$. Since the proposition sequence from t' is $(\neg q)^\omega$, the supremum of the difference in values over $q\mu$ formulas

at s and t' is obtained by satisfying the set of paths formalized as $(\neg q)^* \cdot q^\omega$ at s. The set of paths defined as $(\neg q)^* \cdot q^\omega$ is the same as reaching q in any number of steps, since all states labeled by q are absorbing. Hence,

$$\sup_{\varphi \in q\mu^+} (\llbracket \varphi \rrbracket(s) - \llbracket \varphi \rrbracket(t')) = \llbracket \mu X.(q \vee \mathrm{Pre}_1(X)) \rrbracket(s) \ .$$

It follows from the results of [dAM04] that for all $s \in S$ we have,

$$\llbracket \mu X.(q \vee \mathrm{Pre}_1(X)) \rrbracket(s) = \sup_{\pi_1 \in \Pi_1} \inf_{\pi_2 \in \Pi_2} \Pr_s^{\pi_1,\pi_2}(\Diamond q) \ .$$

From the above equalities and the logical characterization in Theorem 7, we obtain the desired result. ■

6.2.2 Algorithms for the Metrics

We first prove a lemma that helps to obtain reduced-complexity algorithms for concurrent games. The lemma states that the distance $[s \preceq_1 t]$ is attained by restricting player 2 to pure moves at state t, for all states $s, t \in S$.

Lemma 9 *For all concurrent game structures G and all metrics $d \in \mathcal{M}$, we have,*

$$\sup_{k \in C(d)} \sup_{x_1 \in \mathcal{D}_1(s)} \inf_{y_1 \in \mathcal{D}_1(t)} \sup_{y_2 \in \mathcal{D}_2(t)} \inf_{x_2 \in \mathcal{D}_2(s)} (\mathbb{E}_s^{x_1,x_2}(k)) - \mathbb{E}_t^{y_1,y_2}(k))$$

$$= \sup_{k \in C(d)} \sup_{x_1 \in \mathcal{D}_1(s)} \inf_{y_1 \in \mathcal{D}_1(t)} \sup_{b \in \Gamma_2(t)} \inf_{x_2 \in \mathcal{D}_2(s)} (\mathbb{E}_s^{x_1,x_2}(k) - \mathbb{E}_t^{y_1,b}(k)) \ . \quad (6.11)$$

Proof. To prove our claim we fix $k \in C(d)$, and player 1 mixed moves $x \in \mathcal{D}_1(s)$, and $y \in \mathcal{D}_1(t)$. We then have,

$$\sup_{y_2 \in \mathcal{D}_2(t)} \inf_{x_2 \in \mathcal{D}_2(s)} (\mathbb{E}_s^{x,x_2}(k)) - \mathbb{E}_t^{y,y_2}(k)) = \inf_{x_2 \in \mathcal{D}_2(s)} \mathbb{E}_s^{x,x_2}(k) - \inf_{y_2 \in \mathcal{D}_2(t)} \mathbb{E}_t^{y,y_2}(k) \tag{6.12}$$

$$= \inf_{x_2 \in \mathcal{D}_2(s)} \mathbb{E}_s^{x,x_2}(k) - \inf_{b \in \Gamma_2(t)} \mathbb{E}_t^{y,b}(k) \tag{6.13}$$

$$= \sup_{b \in \Gamma_2(t)} \inf_{x_2 \in \mathcal{D}_2(s)} (\mathbb{E}_s^{x,x_2}(k) - \mathbb{E}_t^{y,b}(k)),$$

where (6.13) follows from (6.12) since the decomposition on the rhs of (6.12) yields two independent linear optimization problems; the optimal values are attained at a vertex of the convex hulls of the distributions induced by pure player 2 moves at the two states. This easily leads to the result. ■

We now present algorithms for metrics in concurrent games. Due to the reduction from concurrent reachability games, shown in Theorem 22, 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.

The statements improve on the complexity that can be achieved by a direct translation from the definition of the game simulation metric to the theory of real closed fields. The complexity reduction is based on the observation that using Lemma 9, we can replace a sup operator with finite conjunction, and therefore reduce the quantifier complexity of the resulting formula. Fix a game structure G and states s and t of G. We proceed to con-

struct a statement in the theory of reals that can be used to decide if $[s \preceq_1 t] \leq r$, for a given rational r.

In the following, we use variables x_1, y_1 and x_2 to denote a set of variables $\{x_1(a) \mid a \in \Gamma_1(s)\}$, $\{y_1(a) \mid a \in \Gamma_1(t)\}$ and $\{x_2(b) \mid b \in \Gamma_2(s)\}$ respectively. We use k to denote the set of variables $\{k(u) \mid u \in S\}$, and d for the set of variables $\{d(u,v) \mid u,v \in S\}$. The variables $\alpha, \alpha', \beta, \beta'$ range over reals. For convenience, we assume $\Gamma_2(t) = \{b_1, \ldots, b_l\}$.

First, notice that we can write formulas that state that a variable x is a mixed move for a player at state s, and k is a constructible predicate (i.e., $k \in C(d)$):

$$\mathsf{IsDist}(x, \Gamma_1(s)) \equiv \bigwedge_{a \in \Gamma_1(s)} x(a) \geq 0 \wedge \bigwedge_{a \in \Gamma_1(s)} x(a) \leq 1 \wedge \sum_{a \in \Gamma_1(s)} x(a) = 1$$

$$\mathsf{kBounded}(k,d) \equiv \bigwedge_{u \in S} \left[k(u) \geq \theta_1 \wedge k(u) \leq \theta_2 \right] \wedge \bigwedge_{u,v \in S} \left(k(u) - k(v) \leq d(u,v) \right) .$$

In the following, we write bounded quantifiers of the form "$\exists x_1 \in \mathcal{D}_1(s)$" or "$\forall k \in C(d)$" which mean respectively $\exists x_1.\mathsf{IsDist}(x_1, \Gamma_1(s)) \wedge \cdots$ and $\forall k.\mathsf{kBounded}(k,d) \rightarrow \cdots$.

Let $\eta(k, x_1, x_2, y_1, b)$ be the polynomial $\mathbb{E}_s^{x_1,x_2}(k) - \mathbb{E}_t^{y_1,b}(k)$. Notice that η is a polynomial of degree 3. We write $a = \max\{a_1, \ldots, a_l\}$ for variables $a, a_1, \ldots, a_l$ for the formula

$$(a = a_1 \wedge \bigwedge_{i=1}^{l} a_1 \geq a_i) \vee \ldots \vee (a = a_l \wedge \bigwedge_{i=1}^{l} a_l \geq a_i) .$$

We construct the formula for game simulation in stages. First, we construct a formula $\Phi_1(d, s, t, k, x, \alpha)$ with free variables d, k, x, α such that $\Phi_1(d, s, t, k, x_1, \alpha)$ holds for a valuation to the variables iff

$$\alpha = \inf_{y_1 \in \mathcal{D}_1(t)} \sup_{b \in \Gamma_2(t)} \inf_{x_2 \in \mathcal{D}_2(s)} (\mathbb{E}_s^{x_1,x_2}(k) - \mathbb{E}_t^{y_1,b}(k)) .$$

We use the following observation to move the innermost inf ahead of the sup over the finite set $\Gamma_2(t)$ (for a function f):

$$\sup_{b\in\Gamma_2(t)}\inf_{x_2\in\mathcal{D}_2(s)} f(b,x_2,x) = \inf_{x_2^{b_1}\in\mathcal{D}_2(s)}\ldots\inf_{x_2^{b_l}\in\mathcal{D}_2(s)} \max(f(b_1,x_2^{b_1},x),\ldots,f(b_l,x_2^{b_l},x)) \ .$$

The formula $\Phi_1(d,s,t,k,x_1,\alpha)$ is given by:

$$\begin{aligned}
&\forall y_1\in\mathcal{D}_1(t).\forall x_2^{b_1}\in\mathcal{D}_2(s)\ldots x_2^{b_l}\in\mathcal{D}_2(s).\forall w_1\ldots w_l.\forall a.\forall\alpha'.\\
&\qquad\exists\hat{y}_1\in\mathcal{D}_1(t).\exists\hat{x}_2^{b_1}\in\mathcal{D}_2(s)\ldots\hat{x}_2^{b_l}\in\mathcal{D}_2(s).\exists\hat{w}_1\ldots\hat{w}_l.\exists\hat{a}.\\
&\qquad\left[\left\{\begin{array}{c}\left(w_1=\eta(k,x_1,x_2^{b_1},y_1,b_1)\right)\\ \wedge\cdots\wedge\\ \left(w_l=\eta(k,x_1,x_2^{b_l},y_1,b_l)\right)\wedge\\ \left(a=\max\{w_1,\ldots,w_l\}\right)\end{array}\right\}\rightarrow(a\geq\alpha)\right]\wedge\\
&\qquad\left[\left\{\begin{array}{c}\left(\hat{w}_1=\eta(k,x_1,\hat{x}_2^{b_1},\hat{y}_1,b_1)\right)\\ \wedge\cdots\wedge\\ \left(\hat{w}_l=\eta(k,x_1,\hat{x}_2^{b_l},\hat{y}_1,b_l)\right)\wedge\\ \left(\hat{a}=\max\{\hat{w}_1,\ldots,\hat{w}_l\}\wedge\hat{a}\geq\alpha'(s,t)\right)\end{array}\right\}\rightarrow(\alpha\geq\alpha')\right] .
\end{aligned}$$

Using Φ_1, we construct a formula $\Phi(d,s,t,\alpha)$ with free variables $d\in\mathcal{M}$ and $\alpha\in\mathcal{M}$ such that $\Phi(d,s,t,\alpha)$ is true iff:

$$\alpha = \sup_{k\in C(d)}\sup_{x_1\in\mathcal{D}_1(s)}\inf_{y_1\in\mathcal{D}_1(t)}\sup_{b\in\Gamma_2(t)}\inf_{x_2\in\mathcal{D}_2(s)}(\mathbb{E}_s^{x_1,x_2}(k)-\mathbb{E}_t^{y_1,b}(k)) \ .$$

The formula Φ is defined as follows:

$$\forall k \in C(d).\forall x_1 \in \mathcal{D}_1(s).\forall \beta.\forall \alpha'.$$

$$\left[\begin{array}{c} \Phi_1(d,s,t,k,x_1,\beta) \rightarrow (\beta(s,t) \leq \alpha) \wedge \\ (\forall k' \in C(d).\forall x_1' \in \mathcal{D}_1(s).\forall \beta'.\Phi_1(d,s,t,k',x_1',\beta') \wedge \beta'(s,t) \leq \alpha') \rightarrow \alpha \leq \alpha' \end{array} \right]. \quad (6.14)$$

Finally, given a rational r, we can check if $[s \preceq_1 t] \leq r$ by checking if the following sentence is true:

$$\exists d \in \mathcal{M}.\exists a \in \mathcal{M}.[(\bigwedge_{u,v \in S} \Phi(d,u,v,a(u,v)) \wedge (d(u,v) = a(u,v))) \wedge (d(s,t) \leq r)]. \quad (6.15)$$

The above sentence is true iff in the least fixpoint, $d(s,t)$ is bounded by r. Like in the case of turn-based games and MDPs, given a rational $\epsilon > 0$, using binary search and $\mathcal{O}(\log(\frac{\theta_2 - \theta_1}{\epsilon}))$ calls to a decision procedure to check the sentence (6.15), we can compute an interval $[l,u]$ with $u - l \leq \epsilon$, such that $[s \preceq_1 t] \in [l,u]$.

Complexity. Note that Φ is of the form $\forall\exists\forall$, because Φ_1 is of the form $\forall\exists$, and appears in negative position in Φ. The formula Φ has $(|S| + |\Gamma_1(s)| + 3)$ universally quantified variables, followed by $(|S| + |\Gamma_1(s)| + 3 + 2(|\Gamma_1(t)| + |\Gamma_2(s)| \cdot |\Gamma_2(t)| + |\Gamma_2(t)| + 2))$ existentially quantified variables, followed by $2(|\Gamma_1(t)| + |\Gamma_2(s)| \cdot |\Gamma_2(t)| + |\Gamma_2(t)| + 1)$ universal variables. The sentence (6.15) introduces $|S|^2 + |S|^2$ existentially quantified variables ahead of Φ. The matrix of the formula is of length at most quadratic in the size of the game, and the maximum degree of any polynomial in the formula is 3. We define the size of a game G as: $|G| = |S| + T$, where $T = \sum_{s,t \in S} \sum_{a,b \in Moves} |\delta(s,a,b)(t)|$. Using the complexity of deciding a formula in the theory of real closed fields [Bas99], which states that a formula with i quantifier blocks, where each block has l_i variables, of p polynomials, has a time complexity bound of $\mathcal{O}(p^{\mathcal{O}(\Pi(l_i+1))})$, we get the following result.

Theorem 23 (Decision complexity for exact distance). *For all concurrent game structures G, given a rational r, and two states s and t, whether $[s \preceq_1 t] \leq r$ can be decided in time $\mathcal{O}(|G|^{\mathcal{O}(|G|^5)})$.*

Approximation. Given a rational $\epsilon > 0$, using binary search and $\mathcal{O}(\log(\frac{\theta_2-\theta_1}{\epsilon}))$ calls to check the formula 6.15, 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 4 (Approximation for exact distance). *For all concurrent game structures 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 $\mathcal{O}(\log(\frac{\theta_2-\theta_1}{\epsilon}) \cdot |G|^{\mathcal{O}(|G|^5)})$.*

In contrast, the formula to check whether $[s \preceq_1 t] \leq r$, for a rational r, as implied by the definition of $H_{\preceq_1}(d)(s, t)$, that does not use Lemma 9, has five quantifier alternations due to the inner sup, which when combined with the $2 \cdot |S|^2$ existentially quantified variables in the sentence (6.15), yields a decision complexity of $\mathcal{O}(|G|^{\mathcal{O}(|G|^7)})$.

6.2.3 Computing the 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 $\preceq_1^0, \preceq_1^1, \preceq_1^2, \ldots$, 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 following sentence Φ_s is true:

$$\forall a.\Phi(d^n, s, t, a) \rightarrow a = 0,$$

where Φ is defined as in (6.14) and at step n in the iteration, the distance between any pair of states $u, v \in S$ is defined as follows,

$$\forall u, v \in S.\, d^n(u,v) = \begin{cases} 0 & \text{if (s, t)} \in \preceq_1^n \\ \\ 1 & \text{if (s, t)} \notin \preceq_1^n \end{cases} .$$

To compute the bisimulation kernel, we again proceed by partition refinement. For a set of partitions $\mathcal{Q}^0, \mathcal{Q}^1, \ldots$, where $(s,t) \in V$ for $V \in \mathcal{Q}^n$ implies $(s,t) \in \simeq_1^n$, $(s,t) \in \simeq^{n+1}$ iff the following sentence Φ_b is true for the state pairs (s,t) and (t,s):

$$\forall a.\Phi(d^n, s, t, a) \rightarrow a = 0,$$

where Φ is again as defined in (6.14) and at step n in the iteration, the distance between any pair of states $u, v \in S$ is defined as follows,

$$\forall u, v \in S.\, d^n(u,v) = \begin{cases} 0 & \text{if (s, t)} \in \simeq_1^n \\ \\ 1 & \text{if (s, t)} \notin \simeq_1^n \end{cases} .$$

Complexity. In the worst case we need $\mathcal{O}(|S|^2)$ partition refinement steps for computing both the simulation and the bisimulation relation. At each partition refinement step the number of state pairs we consider is bounded by $\mathcal{O}(|S|^2)$. We can check if Φ_s and Φ_b are true using a decision procedure for the theory of real closed fields. Therefore, we need $\mathcal{O}(|S|^4)$ decisions to compute the kernels. The partitioning of states based on the decisions can be done by any of the partition refinement algorithms, such as [PT87].

Theorem 24 *For all concurrent game structures 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.*

Part II

Games for Synthesis

Chapter 7

Protocol Synthesis

In this chapter we present an application of game models for synthesis. We study the problem of automatically synthesizing *attack-free* fair non-repudiation protocols as participant refinements. We show that classical co-synthesis fails and weak co-synthesis generates solutions that are not attack-free. We then formulate and solve the synthesis problem using assume-guarantee synthesis and prove that the assume-guarantee refinements are attack-free. We provide an alternate characterization of the refinements that are solutions to assume-guarantee synthesis and show that the Kremer-Markowitch fair non-repudiation protocol is a solution to assume-guarantee synthesis whereas the Asokan-Shoup-Waidner protocol and the Garay-Jakobsson-MacKenzie protocol are not. We provide a game-theoretic justification of the need for a trusted third party and conclude by presenting a new *symmetric* attack-free fair non-repudiation protocol.

7.1 Fair Non-Repudiation Protocols

In this section we introduce fair non-repudiation protocols. We first define a protocol model and an attack model. We then introduce the agents and the trusted third party that participate in fair exchange protocols, the messages that they may send and receive, and the channels over which they communicate. Finally, we introduce a set of predicates that are set by the agents and the trusted third party, based on messages sent or received.

A protocol model. Let V be a finite set of variables that take values in some domain D_v. A *valuation* f over the variables V is a function $f : V \mapsto D_v$ that assigns to each variable $v \in V$, a value $f(v) \in D_v$; we take $\mathcal{F}$ as the set of all valuations. Let $\mathcal{M}$ be a finite set of messages that are exchanged between a set $A = \{A_i \mid 0 \leq i \leq n\}$ of participants. We define each participant as a tuple, $A_i = (L_i, V_i, \Lambda_i, \delta_i)$ where L_i is a finite set of line numbers, $V_i \subseteq V$ is a set of variables, $\Lambda_i : \mathcal{F}_i \mapsto 2^{\mathcal{M}}$ is a message assignment, that given a valuation $f \in \mathcal{F}_i$, where $\mathcal{F}_i$ is the set of valuations on V_i, returns the set of messages that can be sent by A_i at f; this set includes all messages that can be composed by A_i based on what she knows in the valuation f. We take $V = \bigcup_{i=0}^{n} V_i$ and assume that the sets V_i form a partition of V. An A_i transition function is $\delta_i : L_i \times \mathcal{F}_i \times \mathcal{M} \mapsto L_i \times \mathcal{F}_i$, that given a line number, a valuation over V_i and a message either sent or received by A_i, returns the next line number of A_i and an updated valuation. The participants may send messages simultaneously and independently. We interpret the elements of A as the most general participants in an exchange and the interaction between the elements of A as the most general exchange program. Every participant in an exchange has her own specification to satisfy. We take the specification of a participant as a set of desired sequences of messages.

A realization of an exchange protocol is a restriction of the most general exchange program that consists of the set $A' = \{A'_i \mid 0 \leq i \leq n\}$ of participants, with behaviors restricted by the rules of the protocol. We take $A'_i = (L'_i, V_i, \Lambda'_i, \delta'_i)$, where $L'_i \subseteq L_i$; V_i is the same set of variables as in A_i; for every valuation $f \in \mathcal{F}_i$ we have $\Lambda'_i(f) \subseteq \Lambda_i(f)$; and $\delta'_i : L'_i \times \mathcal{F}_i \times \mathcal{M} \mapsto L'_i \times \mathcal{F}_i$ is the transition function, that given a line number in L'_i, a valuation over V_i and a move either sent or received by A'_i returns the next line number of A'_i and an updated valuation such that, for $l \in L'_i$, $v \in \mathcal{F}_i$ and $m \in \mathcal{M}$, we have $\delta'_i(l, v, m) = \delta_i(l, v, m)$. We define a *protocol instance* (also called a *protocol run*) as any sequence of messages generated by the participants in A'.

An attack model. We define an *attack* on a protocol as the behavior of a subset of protocol participants such that the resulting sequence of messages is in their specification but not in the specification of at least one of the other participants. Formally, let $Y \subseteq A$ be a subset of the most general participants with $(A \setminus Y)'$ being the remaining participants, that follow the rules of the protocol. A protocol has a Y-attack if the most general participants in Y can generate a message sequence, given $(A \setminus Y)'$ follow the protocol, that is not in the specification of at least one participant in $(A \setminus Y)'$ but is in the specifications of all participants in Y. A protocol is *attack-free*, if there exists no Y-attack for all $Y \in 2^A$.

Agents. An *agent* in a two-party exchange protocol is one of the two participating entities signing an online contract. Based on whether an agent proposes a contract or accepts a contract originating from another agent, we get two roles that an agent can play; that of an *originator* of a contract, designated by O or the *recipient* of a contract, designated by R. Agents communicate with each other over channels.

Trusted third party (TTP). The *trusted third party* or TTP is a participant who is trusted by the agents and adjudicates and resolves disputes. It is known that a fair exchange protocol cannot be realized without the TTP [EY80, PG99]. We model the TTP explicitly as a participant, define her objective and using our formulation give a game-theoretic justification that the TTP is necessary. Agents and the TTP communicate with each other over channels.

Messages. A *message* is an encrypted stream of bytes; we treat each message as an atomic unit. We are not concerned with the exact contents of each message, but in what each message conveys. We define the set $\mathcal{M}$ of messages as follows:

- m_1 is a message that may be sent by O to R. The intent of this message is to convey O's desire to sign a contract with a recipient R.
- m_2 is a message that may be sent by R upon receiving m_1 to O. This conveys R's intent to sign the contract sent by O.
- m_3 is a message that may be sent by O to R upon receiving m_2 and contains the actual signature of O.
- m_4 is a message that contains the actual signature of R and may be sent by R to O upon receiving m_3.
- a_1^{O} is a message that may be sent by O to the TTP and conveys O's desire to *abort* the protocol.
- a_2^{O} (resp. a_2^{R}) is a message that may be sent by the TTP to O (resp. R) that confirms the abort by including an abort token for O (resp. R).

– r_1^{O} (resp. r_1^{R}) is a message that may be sent by O (resp. R) to the TTP and conveys O's (resp. R's) desire to get the TTP to *resolve* a protocol instance by explicitly requesting the TTP to adjudicate. We do not specify the content of r_1^{O} or r_1^{R} but make the assumption that the TTP needs m_1 to recover the protocol for R and similarly needs m_2 to recover the protocol for O.

– r_2^{O} (resp. r_2^{R}) is a message that may be sent by the TTP to O (resp. R) and contains a universally verifiable signature in lieu of the signature of R (resp. O).

The messages that each participant can send in a state depends on what the participant *knows* in that state. We assume that every recipient can check if the message she receives contains what she expects and that it originates from the purported sender. Since our concern in this paper is not to synthesize messages impervious to attacks, we assume this task can be accomplished by the use of appropriate cryptographic primitives. We remark that primitives such as *private contract signatures (PCS)* introduced by Garay et al., in [GJM99], can be used with protocols that are synthesized using our technique to ensure such properties as the *designated verifier property*. In our formulations, we consider a *reasonable TTP* that satisfies the following restrictions on behavior:

1. The TTP will never send a message unless it receives an abort or a resolve request.

2. The TTP processes messages in a first-in-first-out fashion.

3. If the first message received by the TTP is an abort request, then the TTP will eventually send an abort token.

4. If the first message received by the TTP is a resolve request, then the TTP will even-

tually send an agent signature.

Channels. A channel is used to deliver a *message*. There are three types of channels that are typically modeled in the literature. We present them here in decreasing order of reliability:

1. An *operational* channel delivers all messages within a known, finite amount of time.

2. A *resilient* channel eventually delivers all messages, but there is no fixed finite bound on the time to deliver a message.

3. An *unreliable* channel may not deliver all messages eventually.

An operational channel is impractical as the protocols are typically executed over asynchronous networks. We model the channels between the agents as unreliable and those between the agents and the TTP as resilient as in prevailing models; messages sent to the TTP and by the TTP will be eventually delivered. We do not model the channels explicitly, but synthesize protocols irrespective of channel behavior. In particular, unreliable channels may never deliver messages and messages sent to the TTP may arrive in any order at the TTP.

Scheduler. The scheduler is not explicitly part of any fair exchange protocol. The protocol needs to provide all agents the ability to send messages asynchronously. This implies that the agents can choose their actions simultaneously and independently. We model this behavior by using a fair scheduler that assigns each participant a turn and we synthesize refinements against all possible behaviors of a fair scheduler.

Predicates. We introduce the following set of predicates.

– M_1 is set by O, when she sends message m_1 to R.

– EOO, referred to as the *Evidence Of Origin*, is set by R when either m_1 or r_2^{R} is received.

– EOR, referred to as the *Evidence of Receipt*, is set by O when either m_2 or r_2^{O} is received.

– $\mathrm{EOO}_k^{\mathrm{O}}$ and $\mathrm{EOO}_k^{\mathrm{TTP}}$ are referred to as *O's signature*. $\mathrm{EOO}_k^{\mathrm{O}}$ is set by R when R receives m_3 and $\mathrm{EOO}_k^{\mathrm{TTP}}$ is set by R when he receives r_2^{R}.

– $\mathrm{EOR}_k^{\mathrm{R}}$ and $\mathrm{EOR}_k^{\mathrm{TTP}}$ are referred to as *R's signature*. $\mathrm{EOR}_k^{\mathrm{R}}$ is set by O when O receives m_4 and $\mathrm{EOR}_k^{\mathrm{TTP}}$ is set by O when she receives r_2^{O}.

– AO is set by O and indicates that a_2^{O} has been received.

– AR is set by R and indicates that a_2^{R} has been received.

– ABR is set by the TTP when an abort request, a_1^{O} is received.

– RES is set by the TTP when a resolve request, r_1^{O} or r_1^{R}, is received.

All predicates are *monotonic* in that once they are set, they remain set for the duration of a protocol instance [SM02]. We distinguish between a signature sent by an agent and the signature sent by the TTP as a replacement for an agent's signature in the predicates. Distinguishing these signatures enables modeling TTP accountability [SM02]. The non-repudiation of origin for R, denoted by NRO, means that R has received both O's intent to sign a contract and O's signature on the contract so that O cannot deny having signed the contract to a third party. Formally, NRO is defined as: $\mathrm{NRO} = \mathrm{EOO} \wedge (\mathrm{EOO}_k^{\mathrm{O}} \vee \mathrm{EOO}_k^{\mathrm{TTP}})$. The non-repudiation of receipt for O, denoted by NRR, means that O has received both the

intent and signature of R on a contract so that R cannot deny having signed the contract to a third party. Formally, NRR is defined as: $\text{NRR} = \text{EOR} \wedge (\text{EOR}_k^{\text{R}} \vee \text{EOR}_k^{\text{TTP}})$.

7.2 LTL Specifications for Protocol Requirements

In this section, we define specifications for fair non-repudiation protocols, specifications for the agents and the TTP and show that satisfaction of the specifications of the agents and the TTP imply satisfaction of the specifications of the protocols. We first formally define the properties that are required to be satisfied by fair non-repudiation protocols. In our specifications, we use the usual LTL notations $\Box$ and $\Diamond$ to denote *always* (safety) and *eventually* (reachability) specifications, respectively.

Fairness. Informally, fairness for O can be stated as *"For all protocol instances if the non-repudiation of origin (NRO) is ever true, then eventually the non-repudiation of receipt (NRR) is also true"* [KR03]. The fairness property for O is expressed by the LTL formula

$$\varphi_f^{\text{O}} = \Box(\text{NRO} \Rightarrow \Diamond\text{NRR}) \ .$$

Similarly, the fairness property for R is expressed by the LTL formula $\varphi_f^{\text{R}} = \Box(\text{NRR} \Rightarrow \Diamond\text{NRO})$. We say that a protocol is fair, if in all instances of the protocol, fairness for both O and R holds. Hence the fairness requirement for the protocol is expressed by the formula

$$\varphi_f = \varphi_f^{\text{O}} \wedge \varphi_f^{\text{R}} \ . \tag{7.1}$$

Abuse-freeness. The definition of abuse-freeness as given in [GJM99], is the following: *"An optimistic contract signing protocol is abuse-free if it is impossible for a single player at any point in the protocol to be able to prove to an outside party that he has the power to terminate*

(abort) or successfully complete the contract". In [CKS01], the authors introduce a property called *balance* and in [KfR02], the authors show that balance implies abuse-freeness. We translate balance for O as the following LTL formula

$$\varphi_b^{\mathrm{O}} = \Box((\mathrm{NRO} \wedge \mathrm{AR}) \Rightarrow \Diamond \mathrm{NRR}) \ .$$

Similarly, the balance property for R is expressed by the LTL formula $\varphi_b^{\mathrm{R}} = \Box((\mathrm{NRR} \wedge \mathrm{AO}) \Rightarrow \Diamond \mathrm{NRO})$. We say a protocol is balanced and hence abuse-free [KfR02], if in all instances of the protocol, balance for both O and R holds. Hence the balance requirement for the protocol is expressed by the formula

$$\varphi_b = \varphi_b^{\mathrm{O}} \wedge \varphi_b^{\mathrm{R}} \ . \tag{7.2}$$

The following lemma shows that given the above definitions of fairness and balance, fairness implies balance.

Lemma 10 (Fairness implies balance) *We have $\varphi_f \Rightarrow \varphi_b$.*

Proof. For all paths that satisfy φ_f, we have, either $\Diamond$NRO and $\Diamond$NRR are both satisfied or $\Diamond$NRO and $\Diamond$NRR are both violated. In either case, it is easy to see that both φ_b^{O} and φ_b^{R} are satisfied and hence φ_b is satisfied as well. ■

While protocols that ensure fairness are also balanced by Lemma 10, as noted by the authors of [KfR02], a protocol that satisfies balance may not be abuse-free if the availability of partial information to either agent is considered to be a problem. Specifically, in all protocol instances where EOO or EOR is true, but NRO and NRR are both false, one or both agents have the other agent's intent to sign the contract, compromising abuse-freeness if they can prove this intent to anyone other than the TTP. We note that by the

use of PCS, which provides the designated verifier property, neither agent can prove the other agent's intent to sign the contract to anyone other than the TTP. Therefore, ensuring abuse-freeness requires the use of PCS.

Timeliness. Informally, timeliness is defined as follows: *"A protocol respects timeliness, if both agents always have the ability to reach, in a finite amount of time, a point in the protocol where they can stop the protocol while preserving fairness"*. We do not model timeliness in this paper as the cases in the literature where timeliness is compromised involve the lack of an abort subprotocol. Since we explicitly include the capability to abort the protocol, our solution provides timeliness as guaranteed by existing protocols. Alternatively, timeliness could be explicitly modeled in the specifications of the agents and the TTP, but in the interest of keeping the objectives simpler so that we convey the more interesting idea of using assume-guarantee synthesis, we avoid modeling timeliness explicitly.

Signature exchange. A protocol is an exchange protocol if it enables the exchange of signatures. For an exchange protocol to be a non-repudiation protocol, at the end of every run of the protocol, either the agents have their respective non-repudiation evidences or if they do not have their non-repudiation evidences, then they should have the abort token.

We now present the specifications for the agents and the trusted third party and show that satisfaction of these objectives implies that the protocols we synthesize provide fairness and balance.

Specification for the originator O. The objective of the originator O is expressed as follows:

– In all protocol instances, she eventually sends the evidence of origin. This is ex-

pressed by the LTL formula $\varphi_{\mathrm{O}}^1 = \Diamond M_1$.

– In all protocol instances, one of the following statements should be true:

1. (a) The originator eventually gets the recipient's signature $\mathrm{EOR}_k^{\mathrm{R}}$ or, (b) she eventually gets the recipient's signature $\mathrm{EOR}_k^{\mathrm{TTP}}$ and never gets the abort token AO. This is expressed by the LTL formula

$$\varphi_{\mathrm{O}}^2 = (\Diamond \mathrm{EOR}_k^{\mathrm{R}} \vee (\Diamond \mathrm{EOR}_k^{\mathrm{TTP}} \wedge \Box \neg \mathrm{AO})) \; .$$

2. (a) The originator eventually gets the abort token and, (b) the recipient never gets her signature $\mathrm{EOO}_k^{\mathrm{O}}$ and never gets her signature $\mathrm{EOO}_k^{\mathrm{TTP}}$ from the TTP. This is expressed by the LTL formula

$$\varphi_{\mathrm{O}}^3 = \Diamond \mathrm{AO} \wedge (\Box \neg \mathrm{EOO}_k^{\mathrm{O}} \wedge \Box \neg \mathrm{EOO}_k^{\mathrm{TTP}}) = \Diamond \mathrm{AO} \wedge \Box (\neg \mathrm{EOO}_k^{\mathrm{O}} \wedge \neg \mathrm{EOO}_k^{\mathrm{TTP}}) \; .$$

The objective φ_{O} of O can therefore be expressed by the following LTL formula

$$\varphi_{\mathrm{O}} = \varphi_{\mathrm{O}}^1 \wedge \Box (\varphi_{\mathrm{O}}^2 \vee \varphi_{\mathrm{O}}^3) \; . \tag{7.3}$$

There are two interpretations of the abort token in the literature. On the one hand the abort token was never intended to serve as a proof that a protocol instance was not successfully completed; it was to guarantee that the TTP would never resolve a protocol after it has been aborted. On the other hand, there is mention of the abort token being used by the recipient to prove that the protocol was aborted. We take the position that the abort token may be used to ensure TTP accountability as noted in [SM02] and hence include it in the objective of O. If the TTP misbehaves and issues both $\mathrm{EOR}_k^{\mathrm{TTP}}$ and AO, we claim that the objective φ_{O} of the originator should be violated, but in this case, she has the power

to prove that the TTP misbehaved by presenting both $\mathrm{EOR}_k^{\mathrm{TTP}}$ and AO to demonstrate inconsistent behavior. While having both $\mathrm{EOR}_k^{\mathrm{TTP}}$ and AO is a violation of φ_{O}, having both $\mathrm{EOR}_k^{\mathrm{R}}$ and AO is not a violation of φ_{O}; once O receives $\mathrm{EOR}_k^{\mathrm{R}}$, we take it that the objective φ_{O} is satisfied. While having both $\mathrm{EOR}_k^{\mathrm{R}}$ and $\mathrm{EOR}_k^{\mathrm{TTP}}$ may be interpreted as O having inconsistent signatures, we do not consider this to be a violation of O's objective; given the nature of asynchronous networks it may well be the case that both these evidences arrive eventually, one from the TTP and the other from R, as O did not wait long enough before sending r_1^{O}.

Specification for the recipient R. The objective of the recipient R can be expressed as follows:

- In all protocol instances, if he gets the evidence of origin EOO, then one of the following statements should be true:

  1. (a) The recipient eventually gets the originator's signature $\mathrm{EOO}_k^{\mathrm{O}}$ or, (b) he eventually gets the originator's signature $\mathrm{EOO}_k^{\mathrm{TTP}}$ and never gets the abort token AR. This is expressed by the LTL formula

  $$\varphi_{\mathrm{R}}^1 = (\Diamond \mathrm{EOO}_k^{\mathrm{O}} \vee (\Diamond \mathrm{EOO}_k^{\mathrm{TTP}} \wedge \Box \neg \mathrm{AR})) \,.$$

  2. (a) The recipient eventually gets the abort token and, (b) the originator never gets his signature $\mathrm{EOR}_k^{\mathrm{R}}$ and never gets his signature $\mathrm{EOR}_k^{\mathrm{TTP}}$ from the TTP. This is expressed by the LTL formula

  $$\varphi_{\mathrm{R}}^2 = \Diamond \mathrm{AR} \wedge (\Box \neg \mathrm{EOR}_k^{\mathrm{R}} \wedge \Box \neg \mathrm{EOR}_k^{\mathrm{TTP}}) = \Diamond \mathrm{AR} \wedge \Box (\neg \mathrm{EOR}_k^{\mathrm{R}} \wedge \neg \mathrm{EOR}_k^{\mathrm{TTP}}) \,.$$

The objective φ_{R} can therefore be expressed by the LTL formula

$$\varphi_{\mathrm{R}} = \Box(\mathrm{EOO} \Rightarrow (\varphi_{\mathrm{R}}^1 \vee \varphi_{\mathrm{R}}^2)) \,. \tag{7.4}$$

If the TTP misbehaves and issues both $\mathrm{EOO}_k^{\mathrm{TTP}}$ and AR, we claim that the objective φ_{R} of the recipient should be violated, but in this case he has the power to prove that the TTP misbehaved by presenting both $\mathrm{EOO}_k^{\mathrm{TTP}}$ and AR. Similar to the case of O, once R receives $\mathrm{EOO}_k^{\mathrm{O}}$, the objective φ_{R} is satisfied whether or not abort tokens or non-repudiation evidences are issued by the TTP.

Specification for the trusted third party TTP. The objective of the trusted third party is to treat both agents symmetrically and be accountable to both agents. It can be expressed as follows:

– In all protocol instances, if the abort request a_1^{O} or a resolve request r_1^{O} or r_1^{R} is received, then eventually the TTP sends the abort token AO or the abort token AR or the originator's signature $\mathrm{EOO}_k^{\mathrm{TTP}}$ or the recipient's signature $\mathrm{EOR}_k^{\mathrm{TTP}}$. This can be expressed by the LTL formula

$$\varphi_{\mathrm{TTP}}^1 = \Box((\mathrm{ABR} \vee \mathrm{RES}) \Rightarrow (\Diamond \mathrm{AO} \vee \Diamond \mathrm{AR} \vee \Diamond \mathrm{EOO}_k^{\mathrm{TTP}} \vee \Diamond \mathrm{EOR}_k^{\mathrm{TTP}})) \,.$$

– In all protocol instances, if the originator's signature $\mathrm{EOO}_k^{\mathrm{TTP}}$ has been sent to the recipient, then the originator should eventually get the recipient's signature $\mathrm{EOR}_k^{\mathrm{TTP}}$ and the agents should never get the abort token. This can be expressed by the LTL formula

$$\varphi_{\mathrm{TTP}}^2 = \Box(\mathrm{EOO}_k^{\mathrm{TTP}} \Rightarrow (\Diamond \mathrm{EOR}_k^{\mathrm{TTP}} \wedge \Box(\neg \mathrm{AO} \wedge \neg \mathrm{AR}))) \,.$$

– Symmetrically, in all protocol instances, if the recipient's signature $\mathrm{EOR}_k^{\mathrm{TTP}}$ has been sent to the originator, then the recipient should eventually get the originator's signature $\mathrm{EOO}_k^{\mathrm{TTP}}$ and the agents should never get the abort token. This can be expressed by the LTL formula

$$\varphi_{\mathrm{TTP}}^3 = \Box(\mathrm{EOR}_k^{\mathrm{TTP}} \Rightarrow (\Diamond \mathrm{EOO}_k^{\mathrm{TTP}} \wedge \Box(\neg \mathrm{AO} \wedge \neg \mathrm{AR}))) \,.$$

– In all protocol instances, if the originator gets the abort token AO, then the recipient should eventually get the abort token AR and the originator should never get the recipient's signature $\mathrm{EOR}_k^{\mathrm{TTP}}$ and the recipient should never get the originator's signature $\mathrm{EOO}_k^{\mathrm{TTP}}$. This can be expressed by the LTL formula

$$\varphi_{\mathrm{TTP}}^4 = \Box(\mathrm{AO} \Rightarrow (\Diamond \mathrm{AR} \wedge \Box(\neg \mathrm{EOO}_k^{\mathrm{TTP}} \wedge \neg \mathrm{EOR}_k^{\mathrm{TTP}}))) \,.$$

– Symmetrically, in all protocol instances, if the recipient gets the abort token AR, then the originator should eventually get the abort token AO and the originator should never get the recipient's signature $\mathrm{EOR}_k^{\mathrm{TTP}}$ and the recipient should never get the originator's signature $\mathrm{EOO}_k^{\mathrm{TTP}}$. This can be expressed by the LTL formula

$$\varphi_{\mathrm{TTP}}^5 = \Box(\mathrm{AR} \Rightarrow (\Diamond \mathrm{AO} \wedge \Box(\neg \mathrm{EOO}_k^{\mathrm{TTP}} \wedge \neg \mathrm{EOR}_k^{\mathrm{TTP}}))) \,.$$

The objective φ_{TTP} of the TTP is then defined as:

$$\varphi_{\mathrm{TTP}} = \varphi_{\mathrm{TTP}}^1 \wedge \varphi_{\mathrm{TTP}}^2 \wedge \varphi_{\mathrm{TTP}}^3 \wedge \varphi_{\mathrm{TTP}}^4 \wedge \varphi_{\mathrm{TTP}}^5 \,. \tag{7.5}$$

We remark that the specifications of the participants in our protocol model are sequences of messages. Using predicates that are set when messages are sent or received by the agents

or the TTP, we transform those informal specifications into formal objectives using the predicates and LTL. The following theorem shows that the objectives we have introduced (7.3), (7.4) and (7.5) imply fairness (9.1) and balance (7.2).

Theorem 25 (Objectives imply fairness and balance) *The following assertions hold:*

1. *Objectives imply fairness:* $\varphi_O \wedge \varphi_R \wedge \varphi_{TTP} \Rightarrow \varphi_f$, *and*

2. *Objectives imply balance and hence abuse-freeness:* $\varphi_O \wedge \varphi_R \wedge \varphi_{TTP} \Rightarrow \varphi_b$.

Proof. For assertion (1), assume towards a contradiction that there exists a path that satisfies $\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}}$ but does not satisfy φ_f. We consider the case when the path does not satisfy the first conjunct $\varphi_f^{\mathrm{O}} = \Box(\mathrm{NRO} \Rightarrow \Diamond \mathrm{NRR})$ (a similar argument applies to the second conjunct). If the path does not satisfy φ_f^{O}, then there is a suffix of the path, where $\mathrm{EOO} \wedge (\mathrm{EOO}_k^{\mathrm{O}} \vee \mathrm{EOO}_k^{\mathrm{TTP}})$ holds but $\mathrm{EOR} \wedge (\mathrm{EOR}_k^{\mathrm{R}} \vee \mathrm{EOR}_k^{\mathrm{TTP}})$ does not hold at all states of the suffix. It follows that the path satisfies

$$\Diamond\Box(\mathrm{EOO} \wedge (\mathrm{EOO}_k^{\mathrm{O}} \vee \mathrm{EOO}_k^{\mathrm{TTP}}) \wedge (\neg\mathrm{EOR} \vee (\neg\mathrm{EOR}_k^{\mathrm{R}} \wedge \neg\mathrm{EOR}_k^{\mathrm{TTP}}))) \,. \tag{7.6}$$

Consider the objective $\varphi_{\mathrm{O}}^2 = (\Diamond\mathrm{EOR}_k^{\mathrm{R}} \vee (\Diamond\mathrm{EOR}_k^{\mathrm{TTP}} \wedge \Box\neg\mathrm{AO}))$. Since all predicates are monotonic, i.e., once they are set to true in a given protocol instance, they remain set to true for the remainder of that instance, we can rewrite φ_{O}^2 as follows:

$$\varphi_{\mathrm{O}}^2 = \Diamond\Box(\mathrm{EOR}_k^{\mathrm{R}} \vee (\mathrm{EOR}_k^{\mathrm{TTP}} \wedge \neg\mathrm{AO})) \,.$$

Similarly, we can rewrite φ_{O}^3 as follows:

$$\varphi_{\mathrm{O}}^3 = \Diamond\Box(\mathrm{AO} \wedge \neg\mathrm{EOO}_k^{\mathrm{O}} \wedge \neg\mathrm{EOO}_k^{\mathrm{TTP}}) \,.$$

If a path satisfies (7.6), then it also satisfies $\Diamond\Box(\text{EOO}_k^{\text{O}} \vee \text{EOO}_k^{\text{TTP}})$. By the monotonicity of the predicates, we have $\Diamond\Box(\text{EOO}_k^{\text{O}} \vee \text{EOO}_k^{\text{TTP}})$ is equivalent to $\Diamond\Box\text{EOO}_k^{\text{O}} \vee \Diamond\Box\text{EOO}_k^{\text{TTP}}$. We consider the following cases to complete the proof:

1. *Case 1. Path satisfies* $\Diamond\Box EOO_k^{O}$. If the path satisfies $\Diamond\Box\text{EOO}_k^{\text{O}}$, then the path does not satisfy φ_{O}^3. We now show that the path also does not satisfy φ_{O}^2. Since the path satisfies $\Diamond\Box\text{EOO}_k^{\text{O}}$, it must be the case that message m_2 was received by O, as otherwise O will not send EOO_k^{O}. This implies that the path satisfies $\Diamond\Box\text{EOR}$. Since the path satisfies both $\Diamond\Box\text{EOR}$ and (7.6), it follows that the path must satisfy $\Diamond\Box(\neg\text{EOR}_k^{\text{R}} \wedge \neg\text{EOR}_k^{\text{TTP}})$. Hence the path does not satisfy $\Diamond\Box\text{EOR}_k^{\text{R}}$ and $\Diamond\Box\text{EOR}_k^{\text{TTP}}$ leading to the path violating φ_{O}^2. Since the path does not satisfy both φ_{O}^2 and φ_{O}^3, it does not satisfy φ_{O}, which is a contradiction.

2. *Case 2. Path satisfies* $\Diamond\Box EOO_k^{TTP}$. If the path satisfies $\Diamond\Box\text{EOO}_k^{\text{TTP}}$, then either O or R must have sent the resolve request. If the TTP resolves the protocol only to the agent that sends the resolve request and not the other, then the path does not satisfy φ_{TTP}, leading to a contradiction. For φ_{TTP} to hold, the TTP must have sent both $\text{EOO}_k^{\text{TTP}}$ and $\text{EOR}_k^{\text{TTP}}$, which given the channels between the agents and the TTP are resilient implies, (a) EOR must have been set by O upon receiving $\text{EOR}_k^{\text{TTP}}$ leading to the path satisfying $\Diamond\Box\text{EOR}$ and (b) the path satisfies $\Diamond\Box\text{EOR}_k^{\text{TTP}}$. Since the path satisfies $\Diamond\Box\text{EOR}$ and $\Diamond\Box\text{EOR}_k^{\text{TTP}}$, it cannot satisfy (7.6), leading to a contradiction.

For assertion (2), by assertion (1) and Lemma 10, we have $\varphi_{\text{O}} \wedge \varphi_{\text{R}} \wedge \varphi_{\text{TTP}} \Rightarrow \varphi_f \Rightarrow \varphi_b$ as required. ■

7.3 Co-synthesis

In this section we first define processors, schedulers and specifications for fair exchange protocols. Next we define traditional co-operative [CE82] and strictly competitive [PR89, RW87] versions of the co-synthesis problem; we refer to them as *weak co-synthesis* and *classical co-synthesis*, respectively. We then define a formulation of co-synthesis introduced in [CH07] called *assume-guarantee synthesis*.

Variables, valuations, and traces. Let X be a finite set of variables such that each variable $x \in X$ has a finite domain D_x. A *valuation* f on X is a function $f : X \to \bigcup_{x \in X} D_x$ that assigns to each variable $x \in X$ a value $f(x) \in D_x$. We write $\mathcal{F}[X]$ for the set of valuations on X. A *trace* on X is an infinite sequence $(v_0, v_1, v_2, \ldots) \in \mathcal{F}[X]^\omega$ of valuations on X. Given a valuation $f[X] \in \mathcal{F}[X]$ and a subset $Y \subseteq X$ of the variables, we denote by $f[X] \downarrow Y$ the restriction of the valuation $f[X]$ to the variables in Y. Similarly, for a trace $\tau(X) = (v_0, v_1, v_2, \ldots)$ on X, we write $\tau(X) \downarrow Y = (v_0 \downarrow Y, v_1 \downarrow Y, v_2 \downarrow Y, \ldots)$ for the restriction of $\tau(X)$ to the variables in Y. The restriction operator is lifted to sets of valuations, and to sets of traces.

Processes and refinement. Let *Moves* be a finite set of moves. For $i \in \{1, 2, 3\}$, a *process* is defined by the tuple $P_i = (X_i, \Gamma_i, \delta_i)$ where,

1. X_i is a finite set of variables of process P_i with $X = \bigcup_{i=1}^{3} X_i$ being the set of all process variables,

2. $\Gamma_i : \mathcal{F}_i[X_i] \to 2^{Moves} \setminus \emptyset$ is a move assignment that given a valuation in $\mathcal{F}_i[X_i]$, returns a non-empty set of moves, where $\mathcal{F}_i[X_i]$ is the set of valuations on X_i, and

3. $\delta_i : \mathcal{F}_i[X_i] \times \mathit{Moves} \to 2^{\mathcal{F}_i[X_i]} \setminus \emptyset$ is a non-deterministic transition function.

The set of process variables X may be shared between processes. The processes only choose amongst available moves at every valuation of their variables as determined by their move assignment. The transition function maps a present valuation and a process move to a nonempty set of possible successor valuations such that each successor valuation has a unique pre-image. The uniqueness of the pre-image is a property of fair exchange protocols; unique messages convey unique content and generate unique valuations.

A *refinement* of process $P_i = (X_i, \Gamma_i, \delta_i)$ is a process $P_i' = (X_i', \Gamma_i', \delta_i')$ such that:

1. $X_i \subseteq X_i'$,

2. for all valuations $f_i[X_i']$ on X_i', we have $\Gamma_i'(f_i[X_i']) \subseteq \Gamma_i(f_i[X_i'] \downarrow X_i)$, and

3. for all valuations $f_i[X_i']$ on X_i' and for all moves $a \in \Gamma_i'(f_i[X_i'])$, we have $\delta_i'(f_i[X_i'], a) \downarrow X_i \subseteq \delta_i(f_i[X_i'] \downarrow X_i, a)$.

In other words, the refined process P_i' has possibly more variables than the original process P_i, at most the same moves as the moves of the original process P_i at every valuation, and every possible update of the variables in X_i given Γ_i' by P_i' is a possible update by P_i. We write $P_i' \preceq P_i$ to denote that P_i' is a refinement of P_i. Given refinements P_1' of P_1, P_2' of P_2 and P_3' of P_3, we write $X' = X_1' \cup X_2' \cup X_3'$ for the set of variables of all refinements, and we denote the set of valuations on X' by $\mathcal{F}[X']$.

Schedulers. Given processes P_i, where $i \in \{1,2,3\}$, a *scheduler* Sc for P_i chooses at each computation step whether it is process P_1's turn, process P_2's turn or process P_3's turn to update her variables. Formally, the scheduler Sc is a function $\mathrm{Sc} : \mathcal{F}[X]^* \to \{1,2,3\}$ that

maps every finite sequence of global valuations (representing the history of a computation) to $i \in \{1,2,3\}$, signaling that process P_i is next to update her variables. The scheduler Sc is *fair* if it assigns turns to P_1, P_2 and P_3 infinitely often; i.e., for all traces $(v_0, v_1, v_2, \ldots) \in \mathcal{F}[X]^\omega$, there exist infinitely many $j_i \geq 0$, such that $\mathrm{Sc}(v_0, \ldots, v_{j_1}) = 1$, $\mathrm{Sc}(v_0, \ldots, v_{j_2}) = 2$ and $\mathrm{Sc}(v_0, \ldots, v_{j_3}) = 3$. Given three processes $P_1 = (X_1, \Gamma_1, \delta_1)$, $P_2 = (X_2, \Gamma_2, \delta_2)$ and $P_3 = (X_3, \Gamma_3, \delta_3)$, a scheduler Sc for P_1, P_2 and P_3, and a start valuation $v_0 \in \mathcal{F}[X]$, the set of possible traces is:

$$\begin{aligned}[[(P_1 \parallel P_2 \parallel P_3 \parallel \mathrm{Sc})(v_0)]] = \{(v_0, v_1, v_2, \ldots) \in \mathcal{F}[X]^\omega \mid \forall j \geq 0.\, \mathrm{Sc}(v_0, \ldots, v_j) = i;\\ v_{j+1} \downarrow (X \setminus X_i) = v_j \downarrow (X \setminus X_i);\\ v_{j+1} \downarrow X_i \in \delta_i(v_j \downarrow X_i, a) \text{ for some } a \in \Gamma_i(v_j \downarrow X_i))\}\ .\end{aligned}$$

Note that during turns of one process P_i, the values of the private variables $X \setminus X_i$ of the other processes remain unchanged. We define the projection of traces to moves as follows:

$$\begin{aligned}(v_0, v_1, v_2, \ldots) \downarrow \mathit{Moves} = \{(a_0, a_1, a_2, \ldots) \in \mathit{Moves}^\omega \mid \forall j \geq 0.\, \mathrm{Sc}(v_0, \ldots, v_j) = i;\\ v_{j+1} \downarrow X_i \in \delta_i(v_j \downarrow X_i, a_j); a_j \in \Gamma_i(v_j \downarrow X_i)\}\ .\end{aligned}$$

Specifications. A *specification* φ_i for process P_i is a set of traces on X; that is, $\varphi_i \subseteq \mathcal{F}[X]^\omega$. We consider only ω-regular specifications [Tho97]. We define boolean operations on specifications using logical operators such as $\wedge$ (conjunction) and $\Rightarrow$ (implication).

The input to the co-synthesis problem is given as follows: for $i \in \{1,2,3\}$, processes $P_i = (X_i, \Gamma_i, \delta_i)$, specifications φ_i for process i, and a start valuation $v_0 \in \mathcal{F}$.

Weak co-synthesis. The *weak co-synthesis* problem is defined as follows: do there exist refinements $P_i' = (X_i', \Gamma_i', \delta_i')$ and a valuation $v_0' \in \mathcal{F}'$, such that,

1. $P_i' \preceq P_i$ and $v_0' \downarrow X = v_0$, and

2. For all fair schedulers Sc for P_i' we have,

   $[\![(P_1' \parallel P_2' \parallel P_3' \parallel \mathrm{Sc})(v_0')]\!] \downarrow X \subseteq (\varphi_1 \wedge \varphi_2 \wedge \varphi_3)$.

Classical co-synthesis. The *classical co-synthesis* problem is defined as follows: do there exist refinements $P_i' = (X_i', \Gamma_i', \delta_i')$ and a valuation $v_0' \in \mathcal{F}'$, such that,

1. $P_i' \preceq P_i$ and $v_0' \downarrow X = v_0$, and

2. For all fair schedulers Sc for P_i' we have,

   (a) $[\![(P_1' \parallel P_2 \parallel P_3 \parallel \mathrm{Sc})(v_0')]\!] \downarrow X \subseteq \varphi_1$;

   (b) $[\![(P_1 \parallel P_2' \parallel P_3 \parallel \mathrm{Sc})(v_0')]\!] \downarrow X \subseteq \varphi_2$;

   (c) $[\![(P_1 \parallel P_2 \parallel P_3' \parallel \mathrm{Sc})(v_0')]\!] \downarrow X \subseteq \varphi_3$.

Assume-Guarantee synthesis. The *assume-guarantee synthesis* problem is defined as follows: do there exist refinements $P_i' = (X_i', \Gamma_i', \delta_i')$ and a valuation $v_0' \in \mathcal{F}'$, such that,

1. $P_i' \preceq P_i$ and $v_0' \downarrow X = v_0$, and

2. For all fair schedulers Sc for P_i' we have,

   (a) $[\![(P_1' \parallel P_2 \parallel P_3 \parallel \mathrm{Sc})(v_0')]\!] \downarrow X \subseteq (\varphi_2 \wedge \varphi_3) \Rightarrow \varphi_1$;

   (b) $[\![(P_1 \parallel P_2' \parallel P_3 \parallel \mathrm{Sc})(v_0')]\!] \downarrow X \subseteq (\varphi_1 \wedge \varphi_3) \Rightarrow \varphi_2$;

   (c) $[\![(P_1 \parallel P_2 \parallel P_3' \parallel \mathrm{Sc})(v_0')]\!] \downarrow X \subseteq (\varphi_1 \wedge \varphi_2) \Rightarrow \varphi_3$;

   (d) $[\![(P_1' \parallel P_2' \parallel P_3' \parallel \mathrm{Sc})(v_0')]\!] \downarrow X \subseteq (\varphi_1 \wedge \varphi_2 \wedge \varphi_3)$.

7.4 Protocol Co-synthesis

In this section, we present our results on synthesizing fair non-repudiation protocols. We first introduce the protocol synthesis model and show that classical co-synthesis fails and weak co-synthesis generates unacceptable solutions. We provide a game theoretic justification of the need for a TTP by showing that without the TTP neither classical co-synthesis nor assume-guarantee synthesis can be used to synthesize fair non-repudiation protocols. We define the set P_{AGS} of assume-guarantee refinements and prove that the refinements are attack-free. We then present an alternate characterization of the set P_{AGS} and show that the Kremer-Markowitch (KM) non-repudiation protocol with offline TTP, proposed in [MK01, KMZ02, KR03], is included in P_{AGS} whereas the ASW certified mail protocol and the GJM protocol are not. Finally, we systematically analyze refinements of the most general agents and the TTP with respect to their membership in P_{AGS} and show the KM protocol can be automatically generated.

The protocol synthesis model. Our protocol model defined in Section 7.1 leads to a protocol synthesis model based on processes defined in Section 7.3. We take the originator O, the recipient R, the trusted third party TTP as processes with variables X_{O}, X_{R} and X_{TTP} and with objectives φ_{O}, φ_{R} and φ_{TTP} respectively. We do not model the set of channels explicitly but reason against all possible behaviors of unreliable channels. The set of traces $[\![\mathrm{O} \parallel \mathrm{R} \parallel \mathrm{TTP} \parallel \mathrm{Sc}]\!]$, given Sc is a fair scheduler, is then the joint behavior of the most general agents and the most general TTP, subject to the constraint that they can only send messages based on what they know at every valuation of their variables. A protocol is a refinement $\mathrm{O}' \preceq \mathrm{O}$, $\mathrm{R}' \preceq \mathrm{R}$ and $\mathrm{TTP}' \preceq \mathrm{TTP}$, where each participant has a restricted set

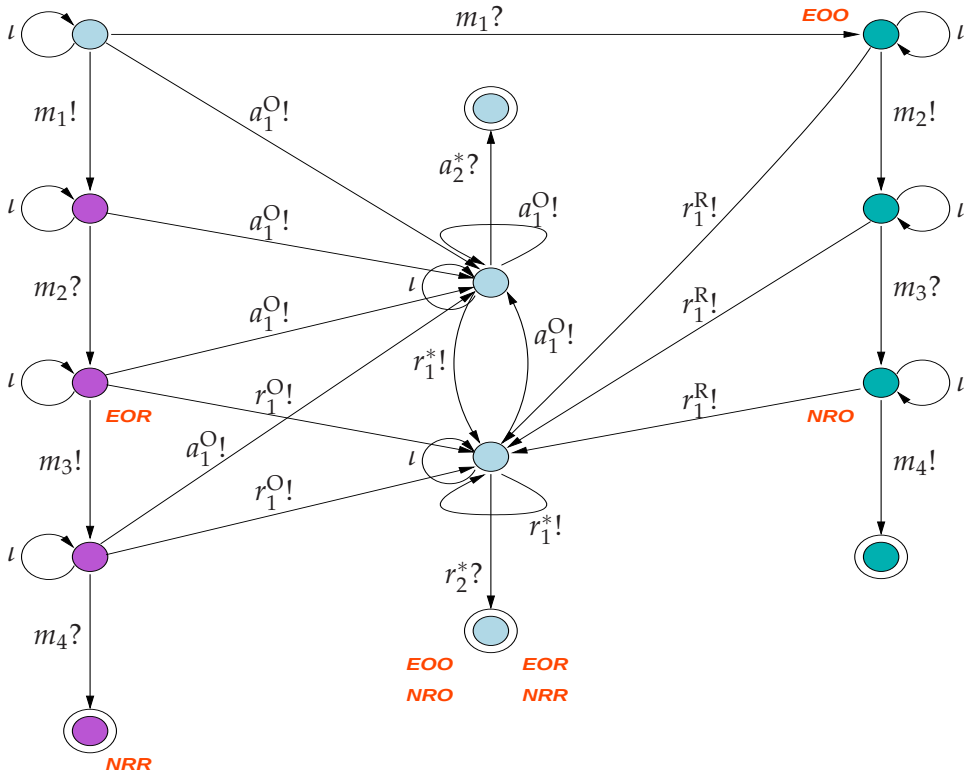


Figure 7.1: An interface automaton that shows the states and enabled moves of the agents O (on the left) and R (on the right). Move ι is the idle move. The states with no outgoing edges are terminal. We consider the most liberal behaviors of the agents wherein the abort and resolve messages can be sent from all states where the agents have the data they need to send those messages. The predicates are monotonic and are shown in the first state at which they hold. In states that can be either agent state, we use the * in the messages a_2^*, r_1^*, r_2^* to denote one of O or R. Abort or resolve requests can be sent from the states marked terminal, but they have no bearing on the outcome of the protocol and hence we omit them.

of moves at every valuation of the process variables; the restrictions constituting the rules of the protocol. We take a protocol state as a valuation over the process variables. By an abuse of notation, we take the set of all messages that can be sent by each process as moves of that process. We get the following moves for the agents and the trusted third party:

1. $\mathit{Moves}_{\mathrm{O}} = \{\iota, m_1, m_3, a_1^{\mathrm{O}}, r_1^{\mathrm{O}}\}$;

2. $\mathit{Moves}_{\mathrm{R}} = \{\iota, m_2, m_4, r_1^{\mathrm{R}}\}$; and

3. $Moves_{\text{TTP}} = \{\iota, a_2^{\text{O}}, a_2^{\text{R}}, [a_2^{\text{O}}, a_2^{\text{R}}], r_2^{\text{O}}, r_2^{\text{R}}, [r_2^{\text{O}}, r_2^{\text{R}}]\}$.

We assume that at every state, the agents and the TTP may choose the *idle* action ι which does nothing. The TTP move $[a_2^{\text{O}}, a_2^{\text{R}}]$ results in the TTP sending messages a_2^{O} to O and a_2^{R} to R. The TTP can choose to send them in any order; all that is guaranteed is that both messages will be sent by the TTP. Similarly for the TTP move $[r_2^{\text{O}}, r_2^{\text{R}}]$. In Figure 7.1 we show an interface automaton for an agent. Since an agent can act either as an originator or a recipient, we show the actions available to the agent in both roles in the figure. The moves for the trusted third party are shown in Table 7.1; these include the moves for the TTP in the ASW certified mail protocol [ASW98], the GJM protocol [GJM99] and the KM protocol [MK01]. We show moves for the TTP with and without a persistent database; it is trivially the case that TTP accountability cannot be satisfied without a persistent database. We therefore do not consider the absence of a persistent database in the rest of this paper. We assume that every message at least includes the name of the sender, is signed with the private key of the sender and encrypted with the public key of the recipient. We take $X = X_{\text{O}} \cup X_{\text{R}} \cup X_{\text{TTP}}$ as a set of boolean variables, where,

- $X_{\text{O}} = \{M_1, \text{EOR}, M_3, \text{EOR}_k^{\text{R}}, \text{EOR}_k^{\text{TTP}}, \text{ABR}^{\text{O}}, \text{RES}^{\text{O}}, \text{AO}\}$,
- $X_{\text{R}} = \{\text{EOO}, M_2, \text{EOO}_k^{\text{O}}, M_4, \text{EOO}_k^{\text{TTP}}, \text{RES}^{\text{R}}, \text{AR}\}$ and
- $X_{\text{TTP}} = \{\text{ABR}, \text{RES}, A_2^{\text{O}}, A_2^{\text{R}}, R_2^{\text{O}}, R_2^{\text{R}}\}$.

The variables M_1, M_3, ABR^{O} and RES^{O} are set by O when she sends m_1, m_3, a_1^{O} and r_1^{O} respectively. The variables M_2, M_4 and RES^{R} are set by R when he sends m_2, m_4 and r_1^{R} respectively. The variables A_2^{O}, A_2^{R}, R_2^{O} and R_2^{R} are set by the TTP when she sends a_2^{O}, a_2^{R},

Agent moves	**Enabled TTP moves**				
	Without DB		**With a persistent DB**		
			ASW	**GJM**	**KM**
O sends $a_1^{\mathbf{O}}$	a_2^{O}	$[a_2^{\mathrm{O}}, a_2^{\mathrm{R}}]$	If R has recovered, invite O to recover else a_2^{O}	If recovered, then r_2^{O} else a_2^{O}	If aborted or recovered, then ι else $[a_2^{\mathrm{O}}, a_2^{\mathrm{R}}]$
O sends $r_1^{\mathbf{O}}$	r_2^{O}	$[r_2^{\mathrm{O}}, r_2^{\mathrm{R}}]$	If aborted, else r_2^{O}	If aborted, then a_2^{O} else r_2^{O}	If aborted or recovered, then ι else $[r_2^{\mathrm{O}}, r_2^{\mathrm{R}}]$
R sends $r_1^{\mathbf{R}}$	r_2^{R}	$[r_2^{\mathrm{O}}, r_2^{\mathrm{R}}]$	If aborted, else r_2^{R}	If aborted, then a_2^{R} else r_2^{R}	If aborted or recovered, then ι else $[r_2^{\mathrm{O}}, r_2^{\mathrm{R}}]$

Table 7.1: In this table we list the choices of moves available to the trusted third party. Each row begins with a message sent by an agent to the TTP followed by the choices available to the TTP in all subsequent states. The TTP moves for the ASW, GJM and KM protocols are shown.

r_2^{O} and r_2^{R} respectively; the variables A_2^{O} and A_2^{R} are both set by the TTP when she sends $[a_2^{\mathrm{O}}, a_2^{\mathrm{R}}]$ and similarly the variables R_2^{O} and R_2^{R} are both set by the TTP when she sends $[r_2^{\mathrm{O}}, r_2^{\mathrm{R}}]$. The remaining variables of the agents and the TTP are set when messages are received as defined in Section 7.1. By an abuse of notation, we represent every state of the protocol by the set of variables that are set to true in that state; for example a valuation $f = \{M_1, \mathrm{EOO}, M_2, \mathrm{EOR}\}$ corresponds to the state of the protocol after messages m_1 and m_2 have been received. $f \downarrow X_{\mathrm{R}} = \{\mathrm{EOO}, M_2\}$ corresponds to the restriction of the valuation f

to the variables of process R; all that R knows in this state is that he has received m_1 and has sent m_2. We take v_0 as the initial valuation where all variables are false. We assume that the refinements $\mathrm{O}' \preceq \mathrm{O}$, $\mathrm{R}' \preceq \mathrm{R}$ and $\mathrm{TTP}' \preceq \mathrm{TTP}$ have the same variables as the processes O, R and TTP respectively, and assume all traces begin with the initial valuation v_0.

7.4.1 Failure of Classical and Weak Co-Synthesis

In this subsection we show that classical co-synthesis fails while weak co-synthesis generates solutions that are not attack-free and are hence unacceptable. We first tackle classical co-synthesis. In order to show failure of classical co-synthesis we need to show that one of the following conditions:

1. $[\![(\mathrm{O}' \parallel \mathrm{R} \parallel \mathrm{TTP} \parallel \mathrm{Sc})]\!] \subseteq \varphi_{\mathrm{O}}$;
2. $[\![(\mathrm{O} \parallel \mathrm{R}' \parallel \mathrm{TTP} \parallel \mathrm{Sc})]\!] \subseteq \varphi_{\mathrm{R}}$;
3. $[\![(\mathrm{O} \parallel \mathrm{R} \parallel \mathrm{TTP}' \parallel \mathrm{Sc})]\!] \subseteq \varphi_{\mathrm{TTP}}$,

can be violated. We show that for all refinements R' of the recipient R, that is, for every sequence of moves ending in a move chosen by R', there exist moves for the other processes O, TTP and Sc, and a behavior of the the channels, to extend that sequence such that the objective φ_{R} is violated. Since R should satisfy his objective against all possible behaviors of the channels, to show failure of classical co-synthesis it suffices to fix the behavior of all channels. We assume the channels eventually deliver all messages.

Theorem 26 (Classical co-synthesis fails for R) *For all refinements $R' \preceq R$, the following assertion holds:*

$$[\![O \parallel R' \parallel TTP \parallel Sc]\!] \not\subseteq \varphi_R.$$

Proof. We consider every valuation of the process variables and the set of all possible moves that can be selected by R at each valuation. This defines all possible refinements of R. Since every valuation is the result of a finite sequence of moves (messages) chosen (sent) by the agents and the TTP, it suffices to consider all possible finite sequences of messages received, ending in a message chosen by R. Let $\tau = (v_0, v_1, \ldots, v_n)$ be a finite sequence of valuations seen in a partial protocol run, where v_0 is the starting valuation. Let $\sigma = \tau \downarrow \mathit{Moves} = \langle a_0, a_1, \ldots, a_{n-1} \rangle$ be the corresponding sequence of n moves seen in the run. At the beginning of a protocol run, we have $\sigma = \emptyset$. In the following, on a case by case basis, we show the sequence of moves seen in a partial protocol run, ending in a move chosen by R, followed by moves for O, TTP and Sc that leads to a violation of φ_{R}.

- R1: $\langle m_1, \iota \rangle$
  - Whenever *Sc* schedules O, she chooses the idle action ι. Since EOO is true, as long as O does not abort the protocol but chooses to remain idle, φ_{R} is violated.
  - φ_{R} and φ_{O} are violated but φ_{TTP} is satisfied.
- R2: $\langle m_1, m_2 \rangle$
  - Sc schedules O; O sends r_1^{O} to TTP;
  - Sc schedules TTP; TTP resolves the protocol for O and sends r_2^{O};
  - Sc schedules O; O aborts the protocol by sending a_1^{O};
  - Sc schedules TTP; TTP sends $[a_2^{\mathrm{O}}, a_2^{\mathrm{R}}]$ with R having no option of obtaining O's signature;
  - φ_{R} and φ_{TTP} are violated but φ_{O} is satisfied.

– R3: $\langle m_1, r_1^{\mathrm{R}} \rangle$

  ◦ Sc schedules TTP; TTP resolves and sends $[r_2^{\mathrm{O}}, r_2^{\mathrm{R}}]$;

  ◦ Sc schedules O; O sends a_1^{O} to TTP;

  ◦ Sc schedules TTP; TTP sends $[a_2^{\mathrm{O}}, a_2^{\mathrm{R}}]$;

  ◦ φ_{R}, φ_{TTP} and φ_{O} are violated.

– R4: $\langle m_1, m_2, r_1^{\mathrm{R}} \rangle$

  ◦ Sc schedules TTP; TTP resolves and sends $[r_2^{\mathrm{O}}, r_2^{\mathrm{R}}]$;

  ◦ Sc schedules O; O sends a_1^{O} to TTP;

  ◦ Sc schedules TTP; TTP sends a_2^{R};

  ◦ φ_{R} and φ_{TTP} are violated but φ_{O} is satisfied.

■

It is easy to verify that the sequences in the proof are exhaustive. From the agent interface automaton shown in Figure 7.1 we can extract all the partial sequences of moves ending in a move of R and similarly for O. In all of the above cases, φ_{R} is violated. In all of the above cases $\varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}}$ is also violated. This shows that for all counter moves of O and the TTP, violation of the specification of R also violates the specification of O or the TTP. Since O and the TTP co-operate, O never sends m_3, instead choosing to use the TTP to get her non-repudiation evidence while denying R the ability to get his evidence.

The following example illustrates that given our objectives, given a reasonable TTP as defined in Section 7.1, weak co-synthesis yields solutions that are not attack-free and are hence unacceptable.

Example. **(Weak co-synthesis generates unacceptable solutions)** Consider a refinement O′, R′ and TTP′, that generates the following sequence of messages: $\langle m_1, m_2, r_1^{\mathrm{O}}, r_2^{\mathrm{O}}, r_1^{\mathrm{R}}, r_2^{\mathrm{R}} \rangle$; the agents send m_1 and m_2 and then resolve the protocol individually. We assume that TTP′ needs both m_1 and m_2 to resolve the protocol for either O or R. The trace corresponding to this sequence satisfies weak co-synthesis. But then this behavior of the TTP, that assumes co-operative agent behavior, is not attack-free. There exists a Y-attack for $Y = \{\mathrm{O}, \mathrm{R}\}$ as follows: after resolving the protocol, if O decides to send m_3 and R responds with m_4, we get the following message sequence: $\langle m_1, m_2, r_1^{\mathrm{O}}, r_2^{\mathrm{O}}, m_3, m_4 \rangle$. In this case, the objectives φ_{O} and φ_{R} are satisfied but the objective φ_{TTP} is violated; a reasonable TTP will only send messages in response to abort and resolve requests and thus needs r_1^{R} to satisfy φ_{TTP}. Similarly, taking $Y = \{\mathrm{R}\}$, consider the following Y-attack where R exploits the fact that a reasonable TTP responds with r_2^{R} when she receives r_1^{R}. If R sends a resolve request immediately after receiving m_1 we get the message sequence $\langle m_1, r_1^{\mathrm{R}}, r_2^{\mathrm{R}} \rangle$. In this case φ_{R} is satisfied, but φ_{O} and φ_{TTP} are violated. The only way to satisfy φ_{O} and φ_{TTP} is if O′ sends r_1^{O}, which she cannot do, as she does not know the contents of m_2. This is an attack on the ASW certified mail protocol that compromises fairness for O [KR03]. Therefore, we see that while there exist solutions that satisfy weak co-synthesis, they may not be attack-free. ■

7.4.2 The Need for a TTP

We now provide a justification of the need for a TTP in fair non-repudiation protocols, given our synthesis objective. While this follows from [EY80, PG99], our proof gives an alternative game-theoretic proof through synthesis. We present the following theorem which shows that if we remove the TTP, then both classical and assume-guarantee

synthesis fail to synthesize a fair non-repudiation protocol.

Theorem 27 (Classical and assume-guarantee synthesis fail without the TTP) *For all refinements $O' \preceq O$, the following assertions hold:*

1. Classical co-synthesis fails*:* $[\![O' \parallel R \parallel Sc]\!] \not\subseteq \varphi_O$.

2. Assume-Guarantee synthesis fails*:*

   (a) $[\![O' \parallel R \parallel Sc]\!] \not\subseteq (\varphi_R \Rightarrow \varphi_O)$ *or,*

   (b) $[\![O' \parallel R \parallel Sc]\!] \subseteq (\varphi_R \Rightarrow \varphi_O)$*;* $[\![R' \parallel O \parallel Sc]\!] \subseteq (\varphi_O \Rightarrow \varphi_R)$*; and* $[\![O' \parallel R' \parallel Sc]\!] \not\subseteq (\varphi_O \wedge \varphi_R)$.

Proof. We note that as the TTP is not involved, AO, AR, $\text{EOO}_k^{\text{TTP}}$ and $\text{EOR}_k^{\text{TTP}}$ are always false. The agent objectives then simplify to,

$$\varphi_{\text{O}} = \Diamond M_1 \wedge \Diamond \text{EOR}_k^{\text{R}}; \qquad \varphi_{\text{R}} = \Box(\text{EOO} \Rightarrow \Diamond \text{EOO}_k^{\text{O}}) \ .$$

For assertion 1, consider an arbitrary refinement $\text{O}' \preceq \text{O}$. We show a witness trace in $[\![\text{O}' \parallel \text{R} \parallel \text{Sc}]\!]$ that violates φ_{O}. If O' does not send m_1 in the initial protocol state v_0, then we have a witness trace that trivially violates φ_{O} and hence $[\![\text{O}' \parallel \text{R} \parallel \text{Sc}]\!] \not\subseteq \varphi_{\text{O}}$. Assume O' sends m_1 and the channel between O and R eventually delivers all messages. Consider a partial trace ending in protocol state $\{M_1, \text{EOO}, M_2, \text{EOR}\}$; messages m_1 and m_2 have been received. The only choice of moves for O' in this state of the protocol are ι or m_3. If O' chooses ι, then the trace does not satisfy φ_{O} and hence $[\![\text{O}' \parallel \text{R} \parallel \text{Sc}]\!] \not\subseteq \varphi_{\text{O}}$. If O' chooses m_3 and upon receiving m_3 if R decides to stop participating in the protocol by choosing ι, then the trace satisfies φ_{R} but violates φ_{O} and hence $[\![\text{O}' \parallel \text{R} \parallel \text{Sc}]\!] \not\subseteq \varphi_{\text{O}}$.

For assertion 2, consider an arbitrary refinement $\mathrm{O}' \preceq \mathrm{O}$. If O' does not send m_1 in the initial protocol state v_0, we have a witness trace that trivially violates φ_{O} but satisfies φ_{R}. Therefore, the trace does not satisfy $\varphi_{\mathrm{R}} \Rightarrow \varphi_{\mathrm{O}}$ and $[\![\mathrm{O}' \parallel \mathrm{R} \parallel \mathrm{Sc}]\!] \not\subseteq (\varphi_{\mathrm{R}} \Rightarrow \varphi_{\mathrm{O}})$. Assume the channels eventually deliver all messages and as in the proof of assertion 1, consider a partial trace ending in protocol state $\{M_1, \mathrm{EOO}, M_2, \mathrm{EOR}\}$. To produce a witness trace we have the following cases based on the move chosen by O':

- *Case 1. O' chooses ι.* Since O' chooses ι, she does not send her signature $\mathrm{EOO}_k^{\mathrm{O}}$. Therefore, the trace does not satisfy φ_{R}. Since R sends m_4 only in response to m_3, O does not get $\mathrm{EOR}_k^{\mathrm{R}}$ from R in this case. Therefore, the trace does not satisfy φ_{O} either and hence satisfies $\varphi_{\mathrm{O}} \Rightarrow \varphi_{\mathrm{R}}$ and $\varphi_{\mathrm{R}} \Rightarrow \varphi_{\mathrm{O}}$ but does not satisfy $\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}}$. This leads to, $[\![\mathrm{O}' \parallel \mathrm{R} \parallel \mathrm{Sc}]\!] \subseteq (\varphi_{\mathrm{O}} \Rightarrow \varphi_{\mathrm{R}})$ and $[\![\mathrm{O}' \parallel \mathrm{R} \parallel \mathrm{Sc}]\!] \subseteq (\varphi_{\mathrm{R}} \Rightarrow \varphi_{\mathrm{O}})$ but $[\![\mathrm{O}' \parallel \mathrm{R} \parallel \mathrm{Sc}]\!] \not\subseteq (\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}})$
- *Case 2. O' chooses m_3.* Since m_3 is eventually delivered, R gets his non-repudiation evidence and the trace satisfies φ_{R}. If R now stops participating in the protocol and chooses the idle move ι instead of sending m_4, then O does not get her non-repudiation evidence and the trace does not satisfy φ_{O}. We therefore have a witness trace that does not satisfy $\varphi_{\mathrm{R}} \Rightarrow \varphi_{\mathrm{O}}$. This leads to, $[\![\mathrm{O}' \parallel \mathrm{R} \parallel \mathrm{Sc}]\!] \not\subseteq (\varphi_{\mathrm{R}} \Rightarrow \varphi_{\mathrm{O}})$

■

If the agents co-operate, then a refinement $\mathrm{O}' \preceq \mathrm{O}$ that sends m_1 and then m_3 upon receiving m_2 and similarly a refinement $\mathrm{R}' \preceq \mathrm{R}$ that sends m_2 and m_4 upon receiving m_1 and m_3 respectively, is a solution to the weak co-synthesis problem. The sequence of messages in this case is precisely $\langle m_1, m_2, m_3, m_4 \rangle$ which is the main protocol in all the fair

exchange protocols we have studied. The problem arises when either O or R are dishonest and try to cheat the other agent.

7.4.3 Assume-Guarantee Solutions are Attack-Free

In this subsection we show that assume-guarantee solutions are attack free; no coalition of participants can violate the objective of at least one of the other participants while satisfying their own objectives. Let $P' = (\mathrm{O}', \mathrm{R}', \mathrm{TTP}')$ be a tuple of refinements of the agents and the TTP. For two refinements $P' = (\mathrm{O}', \mathrm{R}', \mathrm{TTP}')$ and $P'' = (\mathrm{O}'', \mathrm{R}'', \mathrm{TTP}'')$, we write $P' \preceq P''$ if $\mathrm{O}' \preceq \mathrm{O}''$, $\mathrm{R}' \preceq \mathrm{R}''$ and $\mathrm{TTP}' \preceq \mathrm{TTP}''$. Given $P = (\mathrm{O}, \mathrm{R}, \mathrm{TTP})$, the most general behaviors of the agents and the TTP, let P_{AGS} be the set of all possible refinements $P' \preceq P$ that satisfy the conditions of assume-guarantee synthesis. For a refinement $P' = (\mathrm{O}', \mathrm{R}', \mathrm{TTP}')$ to be in P_{AGS}, we require that the refinements $\mathrm{O}' \preceq \mathrm{O}$, $\mathrm{R}' \preceq \mathrm{R}$ and $\mathrm{TTP}' \preceq \mathrm{TTP}$ satisfy the following conditions:

For all fair schedulers Sc, for all possible behaviors of the channels,

1. $[\![(\mathrm{O}' \parallel \mathrm{R} \parallel \mathrm{TTP} \parallel \mathrm{Sc})]\!] \subseteq (\varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{O}}$;
2. $[\![(\mathrm{O} \parallel \mathrm{R}' \parallel \mathrm{TTP} \parallel \mathrm{Sc})]\!] \subseteq (\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{R}}$;
3. $[\![(\mathrm{O} \parallel \mathrm{R} \parallel \mathrm{TTP}' \parallel \mathrm{Sc})]\!] \subseteq (\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}}) \Rightarrow \varphi_{\mathrm{TTP}}$;
4. $[\![(\mathrm{O}' \parallel \mathrm{R}' \parallel \mathrm{TTP}' \parallel \mathrm{Sc})]\!] \subseteq (\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}})$.

We now characterize the smallest restriction on the refinements $\mathrm{TTP}' \preceq \mathrm{TTP}$ that satisfy the implication condition,

$$[\![(\mathrm{O} \parallel \mathrm{R} \parallel \mathrm{TTP}' \parallel \mathrm{Sc})]\!] \subseteq (\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}}) \Rightarrow \varphi_{\mathrm{TTP}} \,. \tag{7.7}$$

In order to characterize the smallest restriction on TTP′ we first define the following constraints on the TTP and prove that they are both necessary and sufficient to satisfy (7.7).

AGS constraints on the TTP. We say that a refinement TTP′ $\preceq$ TTP satisfies the *AGS constraints on the TTP*, if TTP′ satisfies the the following constraints:

1. *Abort constraint.* If the first request received by the TTP is an abort request, then her response to that request should be $[a_2^{\mathrm{O}}, a_2^{\mathrm{R}}]$;

2. *Resolve constraint.* If the first request received by the TTP is a resolve request, then her response to that request should be $[r_2^{\mathrm{O}}, r_2^{\mathrm{R}}]$;

3. *Accountability constraint.* If the first response from the TTP is $[x, y]$, then for all subsequent abort or resolve requests her response should be in the set $\{\iota, x, y, [x, y]\}$.

We assume a reasonable TTP, as defined in Section 7.1; in particular she only responds to abort or resolve requests. In the following lemma, in assertion 1 we show that for all refinements TTP′ $\preceq$ TTP that satisfy the AGS constraints on the TTP, we have TTP′ is inviolable , i.e., neither agent can violate the objective φ_{TTP}, and hence satisfies the implication condition (7.7); in assertion (2) we show that if TTP′ does not satisfy the AGS constraints on the TTP, the implication condition (7.7) is not satisfied.

Lemma 11 *For all refinements TTP′ $\preceq$ TTP, the following assertions hold:*

1. *if TTP′ satisfies the AGS constraints on the TTP, then*

$$[\![O \parallel R \parallel TTP' \parallel Sc]\!] \subseteq \varphi_{TTP} \subseteq (\varphi_O \wedge \varphi_R) \Rightarrow \varphi_{TTP}.$$

2. *if TTP′ does not satisfy the AGS constraints on the TTP, then*

$$[\![O \parallel R \parallel TTP' \parallel Sc]\!] \not\subseteq (\varphi_O \wedge \varphi_R) \Rightarrow \varphi_{TTP}.$$

Proof. For assertion 1, consider an arbitrary TTP′ $\preceq$ TTP that satisfies the AGS constraints on the TTP. We consider the following cases of sets of traces of $[\![\mathrm{O} \parallel \mathrm{R} \parallel \mathrm{TTP}' \parallel \mathrm{Sc}]\!]$ for the proof:

– *Case 1. Neither agent aborts nor resolves the protocol.* In these traces, since the TTP is neither sent an abort nor a resolve request, φ_{TTP} is satisfied trivially. Therefore, all these traces satisfy $(\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}}) \Rightarrow \varphi_{\mathrm{TTP}}$.

– *Case 2. The first request to the TTP is an abort request.* For the set of traces where the first request to the TTP is an abort request, given TTP′ satisfies the AGS constraints on the TTP, by the abort constraint, the response of the TTP to this request is $[a_2^{\mathrm{O}}, a_2^{\mathrm{R}}]$. For all subsequent abort or resolve requests, by the accountability constraint, the TTP responds with a move in set $\{\iota, a_2^{\mathrm{O}}, a_2^{\mathrm{R}}, [a_2^{\mathrm{O}}, a_2^{\mathrm{R}}]\}$. This implies that both agents get the abort token and neither agent gets non-repudiation evidences. Therefore, φ_{TTP} is satisfied for all these traces and hence $(\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}}) \Rightarrow \varphi_{\mathrm{TTP}}$ is also satisfied.

– *Case 3. The first request to the TTP is a resolve request.* Similar to the proof of Case 2, in the set of traces where the first request to the TTP is a resolve request, by the resolve constraint, the TTP responds to this request with move $[r_2^{\mathrm{O}}, r_2^{\mathrm{R}}]$. Since the response of the TTP to all subsequent abort or resolve requests is in the set $\{\iota, r_2^{\mathrm{O}}, r_2^{\mathrm{R}}, [r_2^{\mathrm{O}}, r_2^{\mathrm{R}}]\}$, by the accountability constraint, the agents get their non-repudiation evidences and neither gets the abort token. Therefore, φ_{TTP} is satisfied for all these traces and hence

$(\varphi_O \wedge \varphi_R) \Rightarrow \varphi_{TTP}$ is also satisfied and the result follows.

For assertion 2, consider an arbitrary TTP$'$ $\preceq$ TTP that does not satisfy the AGS constraints on the TTP. We assume a reasonable TTP and consider violation of the AGS constraints on the TTP on a case by case basis. For each case we produce a witness trace that violates the implication condition $(\varphi_O \wedge \varphi_R) \Rightarrow \varphi_{TTP}$. We proceed as follows:

- *Case 1. The abort constraint is violated.* To produce a witness trace we consider a partial trace that ends in protocol state $\{M_1, \text{ABR}^O\}$; O requests the TTP to abort the protocol after sending message m_1 but before it is received. Since TTP$'$ violates the abort constraint, the only choice of moves for TTP$'$ are ι or a_2^O. This leads to the following cases:
  - *Case (a). TTP$'$ chooses ι.* It is trivially the case that φ_{TTP} is violated for this trace as φ_{TTP}^1 is violated. At this stage in the protocol, there exists a behavior of O, R and the channel between O and R, where the channel delivers all messages and the agents co-operate and complete the protocol by exchanging their signatures. Therefore, $\varphi_O \wedge \varphi_R$ is satisfied but φ_{TTP} is violated. Therefore, the trace does not satisfy $(\varphi_O \wedge \varphi_R) \Rightarrow \varphi_{TTP}$.
  - *Case (b). TTP$'$ chooses a_2^O.* Since the channel between the agents and the TTP is resilient, O eventually receives her abort token AO. At this stage in the protocol, there exists a behavior of O, R and the channel between O and R such that the channel delivers all messages and the agents exchange their signatures, leading to the satisfaction of $\varphi_O \wedge \varphi_R$ but a violation of φ_{TTP}^4 and hence φ_{TTP}. Therefore, the trace does not satisfy $(\varphi_O \wedge \varphi_R) \Rightarrow \varphi_{TTP}$.

– *Case 2. The resolve constraint is violated.* To produce a witness trace we consider a partial trace that ends in protocol state $\{M_1, \text{EOO}, M_2, \text{EOR}, \text{RES}^{\text{O}}\}$; O resolves the protocol after messages m_1 and m_2 have been received. Since TTP′ violates the resolve constraint, the only choice of moves for TTP′ are ι or r_2^{O}. An argument similar to the argument for cases 1(a) and 1(b) again leads to the satisfaction of $\varphi_{\text{O}} \wedge \varphi_{\text{R}}$ but a violation of φ_{TTP}.

– *Case 3. The accountability constraint is violated.* To produce a witness trace we consider a partial trace that ends in protocol state $\{M_1, \text{EOO}, M_2, \text{EOR}, \text{ABR}^{\text{O}}, \text{RES}^{\text{R}}, A_2^{\text{O}}, A_2^{\text{R}}, \text{AO}, \text{AR}\}$; O aborts the protocol and R resolves the protocol after messages m_1 and m_2 have been received. The TTP receives the abort request before the resolve request and aborts the protocol by sending $[a_2^{\text{O}}, a_2^{\text{R}}]$. Since TTP′ violates the accountability constraint, the only choice of moves for TTP′ to the resolve request from R are r_2^{R} or $[r_2^{\text{O}}, r_2^{\text{R}}]$. The leads to the following cases:

  ◦ *Case (a). TTP′ chooses* r_2^R. This violates φ_{TTP}^4 and φ_{TTP}^5 and hence violates φ_{TTP}. At this stage in the protocol, there exists a behavior of O, R and the channel between O and R such that the agents exchange their signatures and complete the protocol thus satisfying $\varphi_{\text{O}} \wedge \varphi_{\text{R}}$. Therefore, this trace does not satisfy the implication condition $(\varphi_{\text{O}} \wedge \varphi_{\text{R}}) \Rightarrow \varphi_{\text{TTP}}$.

  ◦ *Case (b). TTP′ chooses* $[r_2^O, r_2^R]$. This violates φ_{TTP}^4 and φ_{TTP}^5 and hence violates φ_{TTP}. An argument similar to Case 2(a) leads to a violation of $(\varphi_{\text{O}} \wedge \varphi_{\text{R}}) \Rightarrow \varphi_{\text{TTP}}$ for this trace.

As we have shown witness traces that do not satisfy the implication condition $(\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}}) \Rightarrow \varphi_{\mathrm{TTP}}$ when TTP' violates any of the AGS constraints on the TTP, the result follows.

■

In the following theorem we show that all refinements $P' \in P_{AGS}$ are attack-free; no subset of participants can violate the objective of at least one of the other participants while satisfying their own objectives.

Theorem 28 *All refinements $P' \in P_{AGS}$ are attack-free.*

Proof. We show that for all refinements $P' \in P_{AGS}$ there exists no Y-attack for all $Y \subseteq \{\mathrm{O}, \mathrm{R}, \mathrm{TTP}\}$. Let $P' = (\mathrm{O}', \mathrm{R}', \mathrm{TTP}')$ and $A = \{\mathrm{O}, \mathrm{R}, \mathrm{TTP}\}$ be the set of participants. We have the following cases:

- *Case 1.* $|Y| = 0$. In this case $Y = \emptyset$ and $(A \setminus Y)' = \{\mathrm{O}', \mathrm{R}', \mathrm{TTP}'\}$. Since $(A \setminus Y)'$ are the refinements in P' which is in P_{AGS}, by the weak co-synthesis condition, the objectives φ_{O}, φ_{R} and φ_{TTP} are satisfied. Therefore there is no Y-attack in this case.

- *Case 2.* $|Y| = 1$. We first show that there is no Y-attack for $Y = \{\mathrm{O}\}$. The case of $Y = \{\mathrm{R}\}$ is similar. By Lemma 11 (assertion 2), for all refinements $P' \in P_{AGS}$, the refinement TTP' must satisfy the AGS constraints on the TTP. This implies, by Lemma 11 (assertion 1), neither O nor R can violate φ_{TTP}. Since φ_{TTP} cannot be violated, a Y-attack in this case must generate a trace where φ_{R} is violated but φ_{O} is satisfied. But this violates the implication condition, $\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{TTP}} \Rightarrow \varphi_{\mathrm{R}}$, contradicting the assumption that $P' \in P_{AGS}$. We now show that there is no Y-attack for $Y = \{\mathrm{TTP}\}$.

Since we assume the TTP is reasonable, in all traces where neither agent sends an abort nor a resolve request to the TTP, the TTP cannot violate the agent objectives. In all traces where the first request from the agents is an abort request, given a reasonable TTP, since the trace satisfies φ_{TTP}, it must be the case that the response to that request is $[a_2^{\text{O}}, a_2^{\text{R}}]$. Similarly, for resolve requests. If the first response of the TTP is $[x, y]$, then the only responses that satisfy φ_{TTP}, to all subsequent abort and resolve requests, are in the set $\{\iota, x, y, [x, y]\}$. This implies that either the agents get abort tokens or non-repudiation evidences but never both, which implies φ_{O} and φ_{R} are satisfied in all these traces. Therefore there is no Y-attack in this case as well.

– *Case 3.* $|Y| = 2$. Since $P' \in P_{AGS}$, by the implication conditions of assume-guarantee synthesis, there cannot be a Y-attack where $|Y| = 2$.

– *Case 4.* $|Y| = 3$. It is trivially the case that there is no Y-attack as $(A \setminus Y)' = \emptyset$.

Since we have shown that for all refinements $P' \in P_{AGS}$, for all $Y \subseteq A$, there is no Y-attack in P', we conclude that all refinements in P_{AGS} are attack-free. ■

We now present the following theorem that establishes conditions for any refinement in P_{AGS} to be an attack-free fair non-repudiation protocol.

Theorem 29 (Fair non-repudiation protocols) *For all refinements $P' \in P_{AGS}$, if $[\![O' \parallel R' \parallel TTP' \parallel Sc]\!] \cap (\Diamond NRO \wedge \Diamond NRR) \neq \emptyset$, then P' is an attack-free fair non-repudiation protocol.*

Proof. Consider an arbitrary refinement $P' = (\text{O}', \text{R}', \text{TTP}') \in P_{AGS}$. Since $P' \in P_{AGS}$, by Theorem 28, it is attack-free. Further, by the weak co-synthesis condition, we have $[\![\text{O}' \parallel \text{R}' \parallel \text{TTP}' \parallel \text{Sc}]\!] \subseteq (\varphi_{\text{O}} \wedge \varphi_{\text{R}} \wedge \varphi_{\text{TTP}})$ and hence by Theorem 25, we have $[\![\text{O}' \parallel \text{R}' \parallel$

$\text{TTP}' \parallel \text{Sc}]\!] \subseteq \varphi_f \cap \varphi_b$. Thus P' satisfies fairness and balance and hence is fair and abuse-free. Since $[\![\text{O}' \parallel \text{R}' \parallel \text{TTP}' \parallel \text{Sc}]\!] \cap (\Diamond\text{NRO} \wedge \Diamond\text{NRR}) \neq \emptyset$, the refinement P' enables an exchange of signatures and hence is an exchange protocol. Given NRO and NRR are non-repudiation evidences for R and O respectively, we conclude that P' is an attack-free fair non-repudiation protocol. ■

7.4.4 Analysis of Existing Fair Non-Repudiation Protocols as P_{AGS} Solutions

In this subsection we analyze existing fair non-repudiation protocols and check if they are solutions to assume-guarantee synthesis. To facilitate the analysis, we first present an alternate characterization of the set P_{AGS} of assume-guarantee refinements. We then show that the KM non-repudiation protocol with offline TTP is in P_{AGS} whereas the ASW certified mail protocol and the GJM protocol are not. Finally, we present a systematic exploration of refinements leading to the KM protocol. Towards an alternate characterization of P_{AGS}, we begin by defining constraints on O, similar to the AGS constraints on the TTP that ensure satisfaction of the implication condition for O. We then define maximal and minimal refinements that satisfy all the implication conditions of assume-guarantee synthesis and introduce a *bounded idle time* requirement to ensure satisfaction of weak co-synthesis.

AGS constraints on O. Given $P = (\text{O}, \text{R}, \text{TTP})$, the most general behaviors of the agents and the TTP, we say a refinement $P' \preceq P$ satisfies the *AGS constraints on O*, if the following conditions hold:

1. $a_1^{\text{O}} \notin \Gamma_{\text{O}'}(v_0)$;

2. $\text{EOO}_k^{\text{O}} \notin \Gamma_{\text{O}'}(\{M_1, \text{EOR}, \text{ABR}^{\text{O}}\})$; and

3. $a_1^{\text{O}} \notin \Gamma_{\text{O}'}(\{M_1, \text{EOR}, M_3\})$.

In the Appendix, we show that these constraints are both necessary and sufficient restrictions on the moves of O that satisfy the implication condition $(\varphi_{\text{R}} \wedge \varphi_{\text{TTP}}) \Rightarrow \varphi_{\text{O}}$ of assume-guarantee synthesis. We also show that all refinements $\text{R}' \preceq \text{R}$ satisfy the implication condition $(\varphi_{\text{O}} \wedge \varphi_{\text{TTP}}) \Rightarrow \varphi_{\text{R}}$ of assume-guarantee synthesis.

The maximal refinement P^*. We define the maximal refinement $\text{P}^* = (\text{O}^*, \text{R}^*, \text{TTP}^*)$ as follows:

1. $\text{O}^* \preceq \text{O}$ satisfies the AGS constraints on O and for all O' that satisfy the constraints, we have $O' \preceq \text{O}^*$;

2. $\text{R}^* = \text{R}$; and

3. $\text{TTP}^* \preceq \text{TTP}$ satisfies the AGS constraints on the TTP and for all TTP' that satisfy the constraints, we have $\text{TTP}' \preceq \text{TTP}^*$.

We show in the Appendix the correspondence between P^* and the smallest restriction on the moves of O and the TTP so that P^* is a witness to P_{AGS}. While there are restrictions on O and the TTP, there are no restrictions on R.

The minimal refinement P_*. We present the smallest refinement $\text{P}_* = (\text{O}_*, \text{R}_*, \text{TTP}_*)$ in P_{AGS}, as the largest restriction on the moves of O, R and the TTP, as follows:

1. $\text{P}_* \preceq \text{P}^*$;

2. $Moves_{\text{O}_*} = \{m_1, a_1^{\text{O}}\}$;

3. $\mathit{Moves}_{\mathrm{R}_*} = \{\iota\}$;

4. O_* satisfies the AGS constraints on O; and

5. TTP_* satisfies the AGS constraints on the TTP.

If $m_1 \notin \mathit{Moves}_{\mathrm{O}_*}$, then φ_{O} cannot be satisfied as O_* does have the ability to initiate a protocol instance. If $a_1^{\mathrm{O}} \notin \mathit{Moves}_{\mathrm{O}_*}$, then φ_{O} cannot be satisfied whether or not m_1 is delivered, as R_* has no choice of moves other than ι. If O_* does not satisfy the AGS constraints on O and sends a_1^{O} in the initial state of the protocol v_0, then the resulting trace trivially violates φ_{O} while satisfying $\varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}}$.

The bounded idle time requirement. We say that a refinement P' satisfies *bounded idle time* if O and the TTP in P' choose the idle move ι, when scheduled by Sc, at most b times for a finite $b \in \mathbb{N}$. We prove that satisfaction of the bounded idle time requirement is both necessary and sufficient to ensure satisfaction of the weak co-synthesis condition of assume-guarantee synthesis, for all refinements that satisfy the AGS constraints on the TTP and the AGS constraints on O, in the Appendix.

Alternate characterization of P_{AGS}. We now use P_* and P^* to provide an alternate characterization of the set P_{AGS}. We first define the following set of refinements $\overline{\mathrm{P}}$:

$$\overline{\mathrm{P}} = \{P' = (\mathrm{O}', \mathrm{R}', \mathrm{TTP}') \mid P' \text{ satisfies bounded idle time; } \mathrm{P}_* \preceq P' \preceq \mathrm{P}^*;$$
$$\mathrm{TTP}' \text{ satisfies the AGS constraints on the TTP}\} .$$

The following lemma states that the set $\overline{\mathrm{P}}$ and the set P_{AGS} coincide. We present the lemma here and prove it in the Appendix.

Protocol 1: THE KM, ASW AND GJM MAIN PROTOCOL

```
1  O sends m_1 to R;
2  R sends m_2 to O;
3  if (R does not send m_2 on time) then
4      O sends a_1^O to the TTP;
5  else
6      O sends m_3 to R;
7      if (O does not send m_3 on time) then
8          R sends r_1^R to the TTP;
9      else
10         R sends m_4 to O;
11         if (R does not send m_4 on time) then
12             O sends r_1^O to the TTP;
```

Figure 7.2: The main protocol in the KM, ASW and GJM protocols.

Lemma 12 (Alternate characterization of P_{AGS}) *We have $\overline{P} = P_{AGS}$.*

The KM non-repudiation protocol. The KM protocol, like the ASW and GJM protocols consists of a main protocol, an abort subprotocol and a resolve subprotocol. The main protocol is the same as in the ASW and GJM protocols and is defined in terms of messages in Protocol 1. The abort subprotocol and the resolve subprotocol are defined in Table 7.1. Let $P_{KM} = (\mathrm{O}_{KM}, \mathrm{R}_{KM}, \mathrm{TTP}_{KM})$ correspond to the agent and TTP refinements in the KM protocol. Since O does not abort the protocol in state v_0 and in state $\{M_1, \mathrm{EOR}, M_3\}$ in O_{KM}, it follows that $\mathrm{O}_* \preceq \mathrm{O}_{KM} \preceq \mathrm{O}^*$. It is easy to verify that $\mathrm{R}_* \preceq \mathrm{R}_{KM} \preceq \mathrm{R}^*$ and $\mathrm{TTP}_* \preceq \mathrm{TTP}_{KM} \preceq \mathrm{TTP}^*$. Moreover, TTP_{KM} satisfies the AGS constraints on the TTP and P_{KM} satisfies bounded idle time. Therefore $P_{KM} \in \overline{\mathrm{P}}$ and hence by Lemma 12, $P_{KM} \in P_{AGS}$.

The ASW certified mail protocol. The ASW certified mail protocol differs from the KM protocol in its abort and resolve sequences. To define the abort protocol, the TTP needs a move req^O that can be used to request O to resolve a protocol instance if R has already resolved it. The abort and resolve subprotocols are defined in Table 7.1. Let $P_{ASW} = (\mathrm{O}_{ASW}, \mathrm{R}_{ASW}, \mathrm{TTP}_{ASW})$ correspond to the agent and TTP refinements in the ASW certified mail protocol. Since TTP_{ASW} neither has move $[a_2^{\mathrm{O}}, a_2^{\mathrm{R}}]$ nor $[r_2^{\mathrm{O}}, r_2^{\mathrm{R}}]$, TTP_{ASW} does not satisfy the AGS constraints on the TTP and hence by Lemma 11 (assertion 2), we have $P_{ASW} \notin P_{AGS}$. Moreover, the ASW certified mail protocol is not attack-free as shown by the following attacks [KR03]: Consider a behavior of the channels that deliver all messages and the sequence of messages $\langle m_1, r_1^{\mathrm{R}}, r_2^{\mathrm{R}}, a_1^{\mathrm{O}}, req^O \rangle$. This is a valid sequence in the ASW protocol. In this sequence a malicious R decides to resolve the protocol after receiving m_1 and thus succeeds in getting $\mathrm{EOO}_k^{\mathrm{TTP}}$. When O_{ASW} attempts to abort the protocol,

TTP_{ASW} expects her to resolve the protocol as R has already resolved it, but O_{ASW} cannot do so as she does not have m_2. Therefore, φ_{O} is violated; O_{ASW} cannot abort or resolve the protocol, neither can she get R's signature. Consider the sequence of messages $\langle m_1, m_2, r_1^{\text{O}}, r_2^{\text{O}}, a_1^{\text{O}}, a_2^{\text{O}} \rangle$. This is an attack that compromises fairness for R; in the words of [KR03] the protocol designers did not foresee that O could resolve the protocol and then abort it. This violates φ_{R} and TTP accountability, violating φ_{TTP}, while satisfying φ_{O}.

The GJM protocol. The GJM protocol differs in the abort and resolve sequences as shown in Table 7.1. Garay et al., introduced the notion of abuse-freeness and invented *private contract signatures or PCS*, a cryptographic primitive that ensures abuse-freeness and optionally TTP accountability [GJM99]. Further, the GJM protocol is faithful to the informal definition of fairness in that, when a protocol instance is aborted, neither agent gets partial information that can be used to negotiate a contract with a third party. This is ensured by the use of PCS which provides the *designated verifier property*; only R can verify the authenticity of a message signed by O and vice versa. The use of PCS in addition to the fixes to the original protocol proposed in [SM02] ensure that the protocol is free from replay attacks, is fair and abuse-free. Let $P_{GJM} = (\text{O}_{GJM}, \text{R}_{GJM}, \text{TTP}_{GJM})$ correspond to the agent and TTP refinements in the GJM protocol. Since TTP_{GJM} neither has move $[a_2^{\text{O}}, a_2^{\text{R}}]$ nor $[r_2^{\text{O}}, r_2^{\text{R}}]$, TTP_{GJM} does not satisfy the AGS constraints on the TTP and hence by Lemma 11 (assertion 2), we have $P_{GJM} \notin P_{AGS}$. P_{GJM} does not provide TTP inviolability and is not attack-free by our definition. Consider the message sequence $g = \langle m_1, m_2, m_3, r_1^{\text{O}}, r_2^{\text{O}} \rangle$; agent R does not send his final signature but goes idle and stops participating in the protocol after receiving O's signature. O_{GJM} resolves the protocol by sending r_1^{O} and gets $\text{EOR}_k^{\text{TTP}}$. In this case,

while the objectives of O and R are satisfied, the TTP cannot satisfy φ_{TTP} unless R_{GJM} co-operates and sends a resolve request r_1^{R} after having satisfied his objective, which he may never do; it is rather unrealistic to expect that he will. Precisely, $g \in [\![\text{O} \parallel \text{R} \parallel \text{TTP}_{GJM} \parallel \text{Sc}]\!]$ and $g \notin (\varphi_{\text{O}} \wedge \varphi_{\text{R}}) \Rightarrow \varphi_{\text{TTP}}$.

Theorem 30 *The refinement corresponding to the KM non-repudiation protocol is in P_{AGS} and the refinements corresponding to the ASW certified mail protocol and the GJM protocol are not in P_{AGS}.*

Computation. We can obtain the solution of assume-guarantee synthesis by solving graph games with *secure equilibria* [CHJ06]. In fact, the refinements that satisfy assume-guarantee synthesis precisely correspond to secure equilibrium strategies of players in the game. This result was presented in [CH07]. All the objectives we consider in this paper are boolean combinations of Büchi ($\Box\Diamond$) and co-Büchi ($\Diamond\Box$) objectives. It follows from [CH07] that secure equilibria with combinations of Büchi and co-Büchi objectives can be solved in polynomial time. This gives us a polynomial time algorithm for the assume-guarantee synthesis of fair exchange protocols.

From P_{AGS} to P_{KM}. We now first present a systematic exploration of the refinements of $P = (\text{O}, \text{R}, \text{TTP})$, the most general behavior of the agents and the TTP, leading to the KM protocol. We consider the following refinements, that we assume satisfy bounded idle time and the AGS constraints on the TTP, and study their properties:

1. $\text{P}_* = (\text{O}_*, \text{R}_*, \text{TTP}_*)$; the minimal refinement.

2. $P_1 = (\text{O}_1, \text{R}_1, \text{TTP}_1)$ with

$Moves_{\mathrm{O}_1} = Moves_{\mathrm{O}_*} \cup \{\iota, m_3\}$, $Moves_{\mathrm{R}_1} = Moves_{\mathrm{R}_*} \cup \{m_2, m_4\}$ and $\mathrm{TTP}_1 = \mathrm{TTP}^*$.

3. $P_2 = (\mathrm{O}_2, \mathrm{R}_2, \mathrm{TTP}_2)$ with

   $Moves_{\mathrm{O}_2} = Moves_{\mathrm{O}_1} \cup \{r_1^{\mathrm{O}}\}$, $Moves_{\mathrm{R}_2} = Moves_{\mathrm{R}_1}$ and $\mathrm{TTP}_2 = \mathrm{TTP}^*$.

4. $P_3 = (\mathrm{O}_3, \mathrm{R}_3, \mathrm{TTP}_3)$ with

   $Moves_{\mathrm{O}_3} = Moves_{\mathrm{O}_2} \setminus \{a_1^{\mathrm{O}}\}$, $Moves_{\mathrm{R}_3} = Moves_{\mathrm{R}_1} \cup \{r_1^{\mathrm{R}}\}$ and $\mathrm{TTP}_3 = \mathrm{TTP}^*$.

5. $\mathrm{P}^* = (\mathrm{O}^*, \mathrm{R}^*, \mathrm{TTP}^*)$; the maximal refinement.

Analysis of the refinement P_*. It is easy to check that while $\mathrm{P}_* \in P_{AGS}$, it always ends aborted as a_1^{O} is the only choice of moves for O_* after m_1 is sent. It is not an exchange protocol as it does not enable an exchange of signatures.

Analysis of the refinement P_1. In this case, the agents do not have the ability to resolve the protocol. The objectives of the agent and the TTP then reduce to,

$$\varphi_{\mathrm{O}} = \Diamond M_1 \wedge \Box(\Diamond \mathrm{EOR}_k^{\mathrm{R}} \vee (\Diamond \mathrm{AO} \wedge \Box \neg \mathrm{EOO}_k^{\mathrm{O}})),$$

$$\varphi_{\mathrm{R}} = \Box(\mathrm{EOO} \Rightarrow (\Diamond \mathrm{EOO}_k^{\mathrm{O}} \vee (\Diamond \mathrm{AR} \wedge \Box \neg \mathrm{EOR}_k^{\mathrm{R}})),$$

$$\varphi_{\mathrm{TTP}} = \Box(\mathrm{ABR} \Rightarrow (\Diamond \mathrm{AO} \vee \Diamond \mathrm{AR})) \wedge \Box(\mathrm{AO} \Rightarrow \Diamond \mathrm{AR}) \wedge \Box(\mathrm{AR} \Rightarrow \Diamond \mathrm{AO})\ .$$

The agent moves that extend partial protocol runs such that the implication conditions of assume-guarantee synthesis are satisfied in all resulting traces is shown in Table 7.2. Each row in the table corresponds to a protocol state and the moves available to O_1 and R_1 at that state, such that the implication conditions of assume-guarantee synthesis are satisfied in all resulting traces. For example, in the row corresponding to $\langle m_1 \rangle$, we have two move choices for O_1, one that selects ι and the other that selects a_1^{O}; O_1 can choose to wait for

Delivered message sequences	**Moves for O_1 and R_1**				
	Choices for O_1		**Choices for R_1**		
$\langle\rangle$	m_1	m_1	ι	ι	ι
$\langle \mathbf{m_1} \rangle$	ι	a_1^{O}	ι	m_2	either ι or m_2
$\langle \mathbf{m_1}, \mathbf{m_2} \rangle$	a_1^{O}	a_1^{O}	ι	ι	ι
$\langle \mathbf{m_1}, \mathbf{m_2}, \mathbf{m_3} \rangle$	$\emptyset$	$\emptyset$	ι	m_4	ι

Table 7.2: The moves that satisfy the objectives of assume-guarantee synthesis for O_1 and R_1 are shown in this table at relevant protocol states represented by message sequences, when the agents have no ability to resolve the protocol.

R to send m_2 or choose a_1^{O}. A similar interpretation is attached to the moves of R_1. We have $\mathrm{P}_* \preceq P_1 \preceq \mathrm{P}^*$. As P_1 satisfies bounded idle time and the AGS constraints on the TTP, $P_1 \in \overline{\mathrm{P}}$ and hence, by Lemma 12, $P_1 \in P_{AGS}$. The refinement P_1, while attack-free, is not a fair non-repudiation protocol as it does not enable an exchange of non-repudiation evidences. The protocol always ends up aborted as a_1^{O} is the only move that satisfies φ_{O} for O in state $\{M_1, \mathrm{EOO}\}$ against all behaviors of R and the TTP; once O_1 sends her signature in m_3, there is no move available to O_1 such that satisfaction of $\varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}}$ is guaranteed to satisfy φ_{O}, as R may decide to stop participating in the protocol.

Analysis of the refinement $\mathbf{P_2}$. In this case, R has no ability to resolve the protocol. It is easy to verify that $\mathrm{P}_* \preceq P_2 \preceq \mathrm{P}^*$. Therefore, $P_2 \in \overline{\mathrm{P}}$ and hence, by Lemma 12, $P_2 \in P_{AGS}$. This protocol is a fair non-repudiation protocol that satisfies fairness, balance and timeliness. If O does not send m_3, then R_2 has no choice of moves. But since P_2 satisfies bounded idle time, O_2 will eventually either abort or resolve the protocol. As TTP_2 satisfies the AGS

Delivered message sequences	**Moves for O_3 and R_3**				
	Choices for O_3		**Choices for R_3**		
$\langle\rangle$	m_1	m_1	ι	ι	ι
$\langle \mathbf{m_1} \rangle$	ι	ι	m_2	r_1^{R}	either m_2 or r_1^{R}
$\langle \mathbf{m_1}, \mathbf{m_2} \rangle$	m_3	r_1^{O}	ι	r_1^{R}	either ι or r_1^{R}
$\langle \mathbf{m_1}, \mathbf{m_2}, \mathbf{m_3} \rangle$	ι	r_1^{O}	ι	m_4	r_1^{R}

Table 7.3: The moves that satisfy the objectives of assume-guarantee synthesis for O_3 and R_3 are shown in this table at relevant protocol states represented by message sequences, when the agents have no ability to abort the protocol.

constraints on the TTP, either both agents get abort tokens or they get their respective non-repudiation evidences eventually.

Analysis of the refinement P_3. Since O has no ability to abort the protocol, while both agents have the ability to resolve it, the predicates AO and AR are always false. The agent and TTP objectives then reduce to,

$$\varphi_{\mathrm{O}} = \Diamond M_1 \wedge \Box(\Diamond \mathrm{EOR}_k^{\mathrm{R}} \vee \Diamond \mathrm{EOR}_k^{\mathrm{TTP}}),$$

$$\varphi_{\mathrm{R}} = \Box(\mathrm{EOO} \Rightarrow (\Diamond \mathrm{EOO}_k^{\mathrm{O}} \vee \Diamond \mathrm{EOO}_k^{\mathrm{TTP}})),$$

$$\varphi_{\mathrm{TTP}} = \Box(\mathrm{RES} \Rightarrow (\Diamond \mathrm{EOO}_k^{\mathrm{TTP}} \vee \Diamond \mathrm{EOR}_k^{\mathrm{TTP}})) \wedge \Box(\mathrm{EOO}_k^{\mathrm{TTP}} \Rightarrow \Diamond \mathrm{EOR}_k^{\mathrm{TTP}}) \wedge$$

$$\Box(\mathrm{EOR}_k^{\mathrm{TTP}} \Rightarrow \Diamond \mathrm{EOO}_k^{\mathrm{TTP}}) \ .$$

The moves of the agents that satisfy the objectives of assume-guarantee synthesis at select protocol valuations represented by message sequences are shown in Table 7.3. It is easy to verify that as $\mathrm{P}_* \not\preceq P_3 \preceq \mathrm{P}^*$, $P_3 \notin \overline{\mathrm{P}}$ and hence by Lemma 12, $P_3 \notin P_{AGS}$. Since TTP_3

satisfies the AGS constraints on the TTP, P_3 is a fair non-repudiation protocol similar to the ZG optimistic non-repudiation protocol, but it does not satisfy timeliness [KMZ02] as O does have the ability to abort the protocol. If message m_1 is not delivered, then O has no choice of moves to satisfy φ_{O}, while $\varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}}$ are satisfied trivially. Balance does not apply in this case as there are no abort moves.

Analysis of the refinement P^*. In the maximal refinement $\mathrm{P}^* = (\mathrm{O}^*, \mathrm{R}^*, \mathrm{TTP}^*)$, since TTP^* satisfies the AGS constraints on the TTP, if her first response to an abort or resolve request is $[x, y]$, she can choose any move in $\{\iota, x, y, [x, y]\}$ for all subsequent abort or resolve requests. Consider a refinement $P_{KM} = (\mathrm{O}_{KM}, \mathrm{R}_{KM}, \mathrm{TTP}_{KM}) \preceq \mathrm{P}^*$, where O_{KM} and R_{KM} correspond to O^* and R^* and $\mathrm{TTP}_{KM} \preceq \mathrm{TTP}^*$ such that TTP_{KM} goes idle after her first response to an abort or resolve request. P_{KM} is then the KM protocol. We remark that given the choices of moves for the TTP after her first response as suggested by assume-guarantee synthesis, choosing ι satisfies the informal notion of efficiency. This refinement ensures fairness, balance and timeliness.

7.5 A Symmetric Fair Non-Repudiation Protocol

In this section we present a symmetric fair non-repudiation protocol that gives R the ability to abort the protocol, assuming that the channels between the agents and the TTP are operational. If we enhance the ability of R by including an abort move a_1^{R} without enhancing O and the TTP, then assume-guarantee synthesis fails. By enhancing both O and the TTP, using assume-guarantee analysis, we design a new fair non-repudiation protocol that (a) has no Y-attack for all $Y \subseteq \{\mathrm{O}, \mathrm{R}\}$; and (b) that provides R the ability to abort. We

show that if the TTP does not change her behavior, while satisfying her objective, then the protocol is attack-free.

In the KM, ASW and GJM protocols, R cannot abort the protocol. While the ability of O to abort the protocol after sending m_1 is required in the event m_1 is not delivered or if R does not send m_2, it can be used to abort the protocol even if all channels are resilient or if O decides not to sign the contract after receiving m_2. The protocols give O the ability to postpone abort decisions but deny R a similar ability. While this does not violate fairness or abuse-freeness as per prevailing definitions, it is not equitable to both agents. If R does not want to participate in a protocol instance, then the only choice of moves for R is ι and not m_2; O will then eventually abort the protocol. Once m_2 has been sent, if R decides not to participate in the protocol and not be held responsible for signing the contract, he has no choice of moves. If he decides to ignore m_3, then O will resolve the protocol resulting in non-repudiation evidences being issued to O, using which she can claim R is obligated by the contract.

Consider the following refinement $\mathrm{P}_s = (\mathrm{O}_s, \mathrm{R}_s, \mathrm{TTP}_s)$ with $\mathrm{P}^* \preceq \mathrm{P}_s$ defined as follows:

- $Moves_{\mathrm{O}_s} = Moves_{\mathrm{O}^*} \cup \{res^{\mathrm{O}}\}$;
- $Moves_{\mathrm{R}_s} = Moves_{\mathrm{R}^*} \cup \{a_1^{\mathrm{R}}\}$; and
- $Moves_{\mathrm{TTP}_s} = Moves_{\mathrm{TTP}^*} \cup \{req^{\mathrm{O}}\}$.

The move req^{O} may be sent by TTP_s only after receiving an abort request from R. The move res^{O} may be sent by O_s only after receiving req^{O}. We present the main protocol and the abort subprotocol for our symmetric fair non-repudiation protocol in Protocol 2 and

Protocol 3; the resolve subprotocol is identical to the one in the KM protocol and shown in Protocol 4.

To facilitate the assume-guarantee analysis of P_s, we present the following *enhanced AGS constraints on the TTP* that is both necessary and sufficient to ensure TTP inviolability (neither agent can violate φ_{TTP}):

1. *Abort constraint.* If the first request received by the TTP is a_1^{O}, then her response to that request should be $[a_2^{\text{O}}, a_2^{\text{R}}]$; If the first request received by the TTP is a_1^{R}, then her response to that request should be req^{O};

2. *Resolve constraint.* If the first request received by the TTP is a resolve request, then her response to that request should be $[r_2^{\text{O}}, r_2^{\text{R}}]$; If the TTP receives res^{O} in response to req^{O} within bounded idle time, then her response should be $[r_2^{\text{O}}, r_2^{\text{R}}]$, otherwise it should be $[a_2^{\text{O}}, a_2^{\text{R}}]$.

3. *Accountability constraint.* If the first response from the TTP is $[x, y]$ or the first response from the TTP is req^{O} and the next response is $[x, y]$, then for all subsequent abort or resolve requests her response should be in the set $\{\iota, x, y, [x, y]\}$.

The enhanced AGS constraints on the TTP are required both to satisfy the implication condition $(\varphi_{\text{O}} \wedge \varphi_{\text{R}}) \Rightarrow \varphi_{\text{TTP}}$ and the condition for weak co-synthesis, $(\varphi_{\text{O}} \wedge \varphi_{\text{R}} \wedge \varphi_{\text{TTP}})$. Since TTP_s waits for a bounded number of turns before sending abort tokens to both agents after sending req^{O}, we require that (a) the channels between the agents and the TTP are operational, and (b) the time taken to deliver messages req^{O} and res^{O} be subsumed by the bound on idle time chosen by the TTP between sending req^{O} and abort tokens. As there is no bound on the time taken to deliver messages on resilient channels, the above

Protocol 2: MAIN PROTOCOL OF OUR SYMMETRIC NON-REPUDIATION PROTOCOL

```
1  O sends m_1 to R;
2  if (R does not want to participate) then
3      R sends a_1^R to the TTP;
4  else
5      R sends m_2 to O;
6      if (R does not send m_2 on time) then
7          O sends a_1^O to the TTP;
8      else
9          O sends m_3 to R;
10         if (O does not send m_3 on time) then
11             if (R does not want to participate) then
12                 R sends a_1^R to the TTP;
13             else
14                 R sends r_1^R to the TTP;
15         else
16             R sends m_4 to O;
17             if (R does not send m_4 on time) then
18                 O sends r_1^O to the TTP;
```

Figure 7.3: Main protocol of our Symmetric Non-Repudiation Protocol

Protocol 3: ABORT SUBPROTOCOL. $X \in \{O, R\}$

1 X sends a_1^X to TTP;
2 **if** *(the protocol has been aborted or resolved)* **then**
3 TTP goes idle;
4 **else**
5 **if** $(X = R)$ **then**
6 TTP sends req^O to O;
7 **if** *(O sends* res^O *on time)* **then**
8 TTP marks this protocol instance as resolved in its persistent DB;
9 TTP sends $[r_2^O, r_2^R]$ to O and R;
10 **else**
11 TTP marks this protocol instance as aborted in its persistent DB;
12 TTP sends $[a_2^O, a_2^R]$ to O and R;
13 **else**
14 TTP marks this protocol instance as aborted in its persistent DB;
15 TTP sends $[a_2^O, a_2^R]$ to O and R;

Figure 7.4: The abort sub-protocol of our symmetric fair non-repudiation protocol.

Protocol 4: RESOLVE SUBPROTOCOL. $X \in \{O, R\}$

1 X sends r_1^X to TTP;
2 **if** *(the protocol has been aborted or resolved)* **then**
3 TTP goes idle;
4 **else**
5 TTP marks this protocol instance as resolved in its persistent DB;
6 TTP sends $[r_2^O, r_2^R]$ to O and R;

Figure 7.5: The resolve sub-protocol of our symmetric fair non-repudiation protocol.

AGS constraints on the TTP cannot be enforced without operational channels. Consider a partial trace that ends in protocol state $\{M_1, \text{EOO}, M_2, \text{EOR}, M_3\}$; messages m_1 and m_2 have been received and m_3 has been sent. If R now aborts the protocol and the TTP sends req^O to O, then resilient channels can delay delivering either req^O or res^O sufficiently for the TTP to abort the protocol. In this case if m_3 is eventually delivered, φ_O is violated whereas $\varphi_R \wedge \varphi_{TTP}$ is satisfied. In the following lemma we show that in P_s, O cannot violate φ_R while satisfying φ_O, R cannot violate φ_O while satisfying φ_R and O and R cannot violate φ_{TTP} while satisfying their objectives. That is, in the refinement P_s we have $[\![O \parallel R \parallel TTP_s \parallel Sc]\!] \subseteq (\varphi_O \wedge \varphi_R) \Rightarrow \varphi_{TTP}$, and $[\![O \parallel R_s \parallel TTP \parallel Sc]\!] \subseteq (\varphi_O \wedge \varphi_{TTP}) \Rightarrow \varphi_R$. However, it is not the case that $[\![O_s \parallel R \parallel TTP \parallel Sc]\!] \subseteq (\varphi_R \wedge \varphi_{TTP}) \Rightarrow \varphi_O$. But if the TTP is fixed then the implication condition holds, i.e., $[\![O_s \parallel R \parallel TTP_s \parallel Sc]\!] \subseteq \varphi_R \Rightarrow \varphi_O \subseteq (\varphi_R \wedge \varphi_{TTP}) \Rightarrow \varphi_O$. It follows that under the assumption that the TTP does not change her behavior, while satisfying her objective, the symmetric protocol is attack-free. We present the following

lemma and prove it in the Appendix.

Lemma 13 *For the refinement $P_s = (O_s, R_s, TTP_s)$, if the channels between the agents and the TTP are operational, then there exists no Y-attack for all $Y \subseteq \{O, R\}$.*

The assumption that the bound on idle time of the TTP between sending req^{O} and abort tokens subsume the time taken for the delivery of messages req^{O} and res^{O} can easily be enforced before the beginning of a protocol; O agrees to participate in the protocol with a given TTP, only if the bound chosen by the TTP is satisfactory. We point out that in state $\{\mathrm{EOO}, M_2\}$, if R sends an abort request, he still needs O's co-operation to abort the protocol. Since she has m_2, she can launch recovery if she so desires by composing res^{O} when she receives req^{O}. But this is identical to the ability of O in aborting the protocol after she sends m_1. R can resolve the protocol as soon as he receives m_1 and thus hold O as a signatory to the contract even if she decided to abort the protocol after sending m_1. The protocol is therefore symmetrical to both O and R. In addition, we claim that this version of the protocol provides better quality of service in terms of timeliness; O does not have to wait after sending m_1 for R to send m_2, in protocol instances where R has no desire to sign the contract. The following theorem states that if the TTP does not change her behavior, then the refinement P_s is an attack-free fair non-repudiation protocol. The proof is in the Appendix.

Theorem 31 *If the TTP does not change her behavior, then the refinement $P_s = (O_s, R_s, TTP_s)$ is an attack-free fair non-repudiation protocol.*

From P_{AGS} to P_s. We can systematically analyze refinements leading to P_s. Similar to the case of synthesizing the KM non-repudiation protocol, we now present the steps that

explore refinements leading to P_s. We assume the TTP satisfies the AGS constraints on the TTP and all refinements satisfy bounded idle time. The analyzed refinements are as follows:

1. $\mathrm{P}_* = (\mathrm{O}_*, \mathrm{R}_*, \mathrm{TTP}_*)$; the minimal refinement.

2. $P_1 = (\mathrm{O}_1, \mathrm{R}_1, \mathrm{TTP}_1)$ with

   $Moves_{\mathrm{O}_1} = Moves_{\mathrm{O}_*} \cup \{\iota, m_3\}$, $Moves_{\mathrm{R}_1} = Moves_{\mathrm{R}_*} \cup \{m_2, m_4\}$ and $\mathrm{TTP}_1 = \mathrm{TTP}^*$.

3. $P_2 = (\mathrm{O}_2, \mathrm{R}_2, \mathrm{TTP}_2)$ with

   $Moves_{\mathrm{O}_2} = Moves_{\mathrm{O}_1} \cup \{r_1^{\mathrm{O}}\}$, $Moves_{\mathrm{R}_2} = Moves_{\mathrm{R}_1}$ and $\mathrm{TTP}_2 = \mathrm{TTP}^*$.

4. $P_3 = (\mathrm{O}_3, \mathrm{R}_3, \mathrm{TTP}_3)$ with

   $Moves_{\mathrm{O}_3} = Moves_{\mathrm{O}_2} \setminus \{a_1^{\mathrm{O}}\}$, $Moves_{\mathrm{R}_3} = Moves_{\mathrm{R}_1} \cup \{r_1^{\mathrm{R}}\}$ and $\mathrm{TTP}_3 = \mathrm{TTP}^*$.

5. $\mathrm{P}^* = (\mathrm{O}^*, \mathrm{R}^*, \mathrm{TTP}^*)$; the maximal refinement.

6. $\mathrm{P}_s = (\mathrm{O}_s, \mathrm{R}_s, \mathrm{TTP}_s)$ with

   $Moves_{\mathrm{O}_s} = Moves_{\mathrm{O}^*} \cup \{res^{\mathrm{O}}\}$, $Moves_{\mathrm{R}_s} = Moves_{\mathrm{R}^*} \cup \{a_1^{\mathrm{R}}\}$ and

   $Moves_{\mathrm{TTP}_s} = Moves_{\mathrm{TTP}^*} \cup \{req^{\mathrm{O}}\}$.

Chapter 8

Resource Manager Synthesis

In this chapter, we present our second application of games for synthesis. We show the automatic synthesis of resource managers, that manage the runtime allocation of mutexes and semaphores in multi-threaded C programs, such that there are no deadlocks and all threads make progress. We begin by illustrating the advantages of code-aware managers. Consider the threads in Figure 8.1. Thread 1 and Thread 2 can lead to a deadlock under a standard, most liberal resource manager. On the other hand, the code-aware manager we construct is able to differentiate, in Thread 1, between the requests for the mutex a occurring on the **then** and **else** branches of the **if** statement (during code analysis, information about the location of resource manager calls is added to the calls themselves). When Thread 1 holds mutex a, and Thread 2 requests mutex b, the request is granted if Thread 1 is in the **else** branch, and denied otherwise. Similarly, when Thread 2 holds the mutex b, and Thread 1 requests the mutex a, the request is granted if Thread 1 is in the **else** branch, and denied otherwise. In all cases, the code-aware manager guarantees deadlock

```
while (1) {
  if (exp) {
    mutex_lock(a);
    mutex_lock(b);
    // critical region
    mutex_unlock(b);
    mutex_unlock(a);
  } else {
    mutex_lock(a);
    mutex_lock(c);
    // critical region
    mutex_unlock(c);
    mutex_unlock(a);
  }
}
```

(a) Thread 1

```
while (1) {
  mutex_lock(b);
  mutex_lock(a);
  // critical region
  mutex_unlock(a);
  mutex_unlock(b);
}
```

(b) Thread 2

Figure 8.1: Two fragments of C code.

freedom while managing resources in a fair and liberal manner.

8.1 Thread Resource Interfaces

8.1.1 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. Intuitively,

these actions correspond to communications between the threads that manipulate the resource and the resource manager, and have the following meaning:

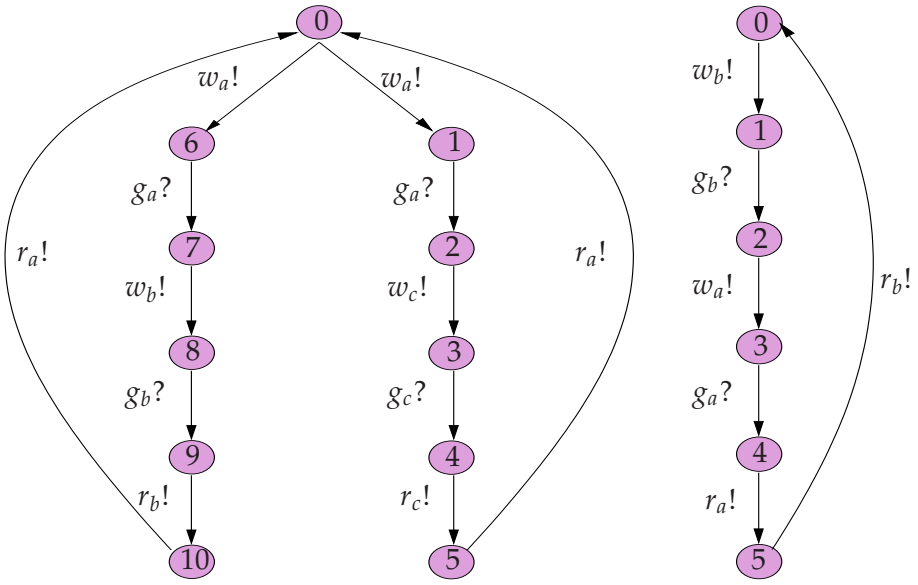
- $w_x!$: a thread requests the resource x ("want x").
- $g_x?$: the resource manager grants the resource x to a thread ("get x").
- $r_x!$: the thread releases the resource x ("release x").

Given a set R of resources, the set of *actions on* R is $Acts[R] = \{w_x!, g_x?, r_x! \mid x \in R\} \cup \{\varepsilon\}$. The *output actions* over R are given by $Acts^O[R] = \{w_x!, r_x! \mid x \in R\} \cup \{\varepsilon\}$, and correspond to communication from the thread to the resource manager. In addition, we have a special action ε which is needed in Definition 4 below. The *input actions* over R are given by $Acts^I[R] = \{g_x? \mid x \in R\}$, and correspond to communication from the resource manager to the thread. We consider two types of resources: *mutexes* and (counting) *semaphores.* A mutex is a resource that takes value in $\{0, 1\}$ and starts from the initial value 1; a mutex can only be released by the same thread that acquired it (as in POSIX). A semaphore, on the other hand, can be initialized to any integer, and can be released and acquired without constraints, except that its value can never become negative.

8.1.2 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.

Definition 2 A *thread interface* $I = (R, S, E, s^{\text{init}}, \lambda)$ consists of a set R of resources, a finite control-flow graph (S, E) with $E \subseteq S \times S$, an initial state $s^{\text{init}} \in S$, and an *action label* $\lambda : E \to Acts[R] \setminus \{\varepsilon\}$ mapping each edge to a resource action, such that



(a) Thread interface 1

(b) Thread interface 2

Figure 8.2: The thread interfaces corresponding to the code in Figure 8.1.

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

Example 12 Consider the POSIX interface for mutexes with functions `mutex_lock(x)` and

`mutex_unlock(x)`. Each call `mutex_lock(x)` is represented by the pair of actions $w_x!$ and $g_x?$; a (nonblocking) call `mutex_unlock(x)` is represented by the action $r_x!$. Similarly, for a counting semaphore y, the function `sem_wait(y)` corresponds to the two actions $w_y!$ and $g_y?$, and the function `sem_post(y)` corresponds to the release action $r_y!$. For example, our tool extracts the resource interfaces of Figure 8.2 from the code in Figure 8.1. ■

8.1.3 Systems

Syntax

Given a set R of resources, a *resource valuation* is a function $\nu : R \mapsto \mathbb{N}$ mapping each resource to a natural number value. For a valuation ν and $x \in R$, we denote by $\nu[x := k]$ the valuation obtained from ν by assigning the value $k \in \mathbb{N}$ to x. A *system* is a set of resources, an initial resource valuation of the resources, and a tuple of (a fixed number of) thread interfaces.

Definition 3 A *system* is a tuple $\mathcal{I} = (R, \nu^0, (I_1, \ldots, I_n))$, consisting of a set R of resources, a mapping $\nu^0 : R \mapsto \mathbb{N}$ assigning an initial value to each resource, and of $n > 0$ thread interfaces $I_1, \ldots, I_n$. We require that $R_i \subseteq R$, for $1 \leq i \leq n$, and that if $x \in R$ is a mutex, $\nu^0(x) = 1$.

Semantics

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.

Definition 4 Given a system $\mathcal{I} = (R, \nu^0, (I_1, \ldots, I_n))$, its *joint interface* is a tuple $M_{\mathcal{I}} = (R, S, E, s^{\text{init}}, \lambda, \theta)$, where R is as in $\mathcal{I}$, and:

- $S = (\prod_i S_i) \times (R \mapsto \mathbb{N})$;
- $s^{\text{init}} = (s_1^{\text{init}}, \ldots, s_n^{\text{init}}, \nu^0)$;
- $E \subseteq S \times S$, and $\lambda : E \mapsto \mathit{Acts}[R]$, $\theta : E \mapsto \{0, \ldots, n\}$ are defined as follows. Let $s = (s_1, \ldots, s_n, \nu) \in S$; 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 $t = (s_1, \ldots, s_{i-1}, s_i', s_{i+1}, \ldots, s_n, \nu')$ we have:

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

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

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

Moreover, let s be a state that has no successors according to the above rules. Then, we add a self-loop $(s, s) \in E$ and we set $\lambda(s, s) = \varepsilon$ and $\theta(s, s) = 0$.

Let $s \in S$ and $s = (s_1, \ldots, s_n, \nu)$; for all $i = 1, \ldots, n$, we set $\mathit{loc}_i(s) = s_i$. We let *Osucc*, *Isucc* refer to $M_{\mathcal{I}}$, and for $1 \leq i \leq n$, we let Osucc_i, Isucc_i refer to I_i.

In $M_{\mathcal{I}}$, edges labeled with the special action ε are a technical addition, used to ensure that all finite paths can be extended to infinite ones.

The portion of the joint interface $M_{\mathcal{I}}$ that is reachable from its initial state s^{init} may not be finite, as the value of resources could grow beyond bounds. Of course, if all resources are

mutexes (which take values 0 and 1), the state space is finite. In general, a coverability tree algorithm for Petri nets can check for boundedness, but this check is expensive.

Theorem 32 *Let $M_{\mathcal{I}} = (R, S, E, s^{\text{init}}, \lambda, \theta)$ be the joint interface of a system $\mathcal{I}$. The problem of deciding whether the portion of S that is reachable in (S, E) is finite is EXPSPACE-hard.*

Proof. If all resources in R are mutexes, then clearly they can assume only the values 0, 1 at any state reachable from s^{init}. The second result is by a reduction to the boundedness problem for Petri Nets. ■

In the following, we only consider systems $\mathcal{I}$ such that the reachable portion of $M_{\mathcal{I}}$ is finite. In our tool CYNTHESIS we avoid solving the question of whether the portion of the joint interface reachable from the initial state is finite. Rather, we simply take as input the maximum value to consider for any semaphore; this value is usually well known to the programmer. If we find a reachable state where the value of a semaphore is greater than this maximum, we stop and report the problem.

8.2 The Scheduling Game

In this section, unless otherwise noted, we consider a fixed system $\mathcal{I} = (R, \nu^0, (I_1, \ldots, I_n))$, 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 is to synthesize a resource manager that ensures that all threads make 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. Notice that for all thread interfaces, the set of final states is absorbing. Therefore, $final_i$ being true over an edge $(s,t) \in E$, implies that it remains true along all paths that originate at s; this means that $\Box final_i$ holds on all paths that originate at s. Using

temporal logic notation, the goal can therefore 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 the goal $\phi_{\mathcal{I}}^{goal}$. We first describe the two sources of non-determinism that the resource manager plays against.

Inter-thread non-determinism.

This non-determinism is due to the scheduler. If there are two or more threads that are waiting to issue output actions, then which one gets to do so depends on the underlying OS scheduler. If two threads want to get a resource, which thread gets to call the OS primitive to acquire the resource is decided by the scheduler. Similarly, if a thread wants to release a resource, whether or not it does so again depends on the scheduler.

Intra-thread non-determinism.

This non-determinism has two origins. The first is the *environment:* often, the behavior of a thread in an embedded system reacts to inputs (input timings, or input values) received from the environment. The second is *abstraction:* our thread interface is an abstraction of the actual thread behavior that disregards variable values. In particular, the outcome of control-flow statements such as loop tests, and if-then-else, is modeled as intra-thread non-determinism. Assuming that intra-thread non-determinism is resolved in an arbitrary way may easily lead to declaring the manager synthesis problem to be infeasible.[1] In fact, whenever a thread can execute a loop while holding a resource, the arbitrary

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

resolution of intra-thread non-determinism introduces the possibility that the loop never terminates.

The synthesis objective.

```
while (1) {
  if (exp) {
    mutex_lock(a);
    x = 1;
    // critical region
    mutex_unlock(a);
  } else {
    mutex_lock(b);
    mutex_lock(c);
    x = 1;
    // critical region
    mutex_unlock(b);
    while (x != 2)
      ;
    mutex_unlock(c);
  }
}
```

(a) Thread 1

```
while (1) {
  if (!exp) {
    mutex_lock(b);
    mutex_lock(c);
    x = 2;
    // critical region
    mutex_unlock(b);
    while (x != 1)
      ;
    mutex_unlock(c);
  } else {
    mutex_lock(d);
    x = 2;
    // critical region
    mutex_unlock(d);
  }
}
```

(b) Thread 2

```
while (1) {
  mutex_lock(b);
  mutex_lock(c);
  // critical region
  mutex_unlock(c);
  mutex_unlock(b);
}
```

(c) Thread 3

Figure 8.3: A system of three threads to illustrate the assumptions on the sources of non-determinacy and the goal of the resource manager.

We first show that the automatic synthesis of a resource manager given the

goal $\phi_{\mathcal{I}}^{goal}$ will fail against arbitrary resolution of the inter-thread and intra-thread non-determinism. We then use fairness assumptions on inter-thread and intra-thread non-determinism and derive a synthesis objective that satisfies $\phi_{\mathcal{I}}^{goal}$, given these fairness assumptions. Consider the system of three threads in Figure 8.3. If we assume that the scheduler never schedules Thread 3, then the $w_b!$ action from Thread 3 never takes place. In this case, irrespective of the resource manager policy, $\phi_{\mathcal{I}}^{goal}$ is not satisfied. We need to restrict the inter-thread non-determinism and we do so by placing a fairness assumption on the underlying operating system scheduler: more precisely, 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:

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

We now show that even if we assume that inter-thread non-determinism is resolved satisfying $\phi_{\mathcal{I}}^{inter}$, the goal of the resource manager can still be violated: more precisely, the resource manager cannot ensure that $\phi_{\mathcal{I}}^{inter} \Rightarrow \phi_{\mathcal{I}}^{goal}$. Assume that the conditional expression *exp* in Thread 1 and Thread 2 is always false. Thread 1 releases resource b and waits till Thread 2 acquires it before releasing resource c. Similarly, Thread 2 releases resource b and waits till Thread 1 acquires it before releasing resource c. There is no way to resolve intra-thread non-determinism in this case to ensure that Thread 3 can make progress

for any policy followed by the resource manager. Notice that even if we assume that the scheduler is fair and that Thread 3 is scheduled infinitely often, it cannot make progress because resource b is always held by either Thread 1 or Thread 2. Therefore the resource manager cannot ensure that $\phi_{\mathcal{I}}^{inter} \Rightarrow \phi_{\mathcal{I}}^{goal}$.

We need to restrict intra-thread non-determinism and we do so by placing a fairness constraint on intra-thread non-determinism: if each choice is presented infinitely often, then each choice outcome is followed infinitely often. 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 $take_i^{u,v}(s,t) \stackrel{\text{def}}{=} \big((loc_i(s) = u) \wedge (loc_i(t) = v)\big)$. The fairness assumption for intra-thread non-determinism can then be written as

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

This entails that the conditional expression *exp* takes both values infinitely often. With this assumption, Thread 1 (Thread 2) will enter the *then* (*else*) branch of the conditional statement infinitely often. This implies that either Thread 1 is holding resource a or Thread 2 is holding resource d infinitely often. The resource manager strategy is as follows:

- If both Thread 1 and Thread 2 are holding resources a and d, then a winning strategy for the resource manager would be to assign resources b and c to Thread 3.

- If Thread 1 is holding resource a and Thread 2 is holding resources b and c. Then a winning strategy for the resource manager would be to wait till Thread 2 releases both b and c and then allocate these resources to Thread 3, thus ensuring that Thread 3 enters its critical region.

– If Thread 2 is holding resource d and Thread 1 is holding resources b and c, then a strategy similar to the one above will ensure that Thread 3 enters its critical region.

Therefore, the objective for resource manager synthesis requires fairness assumptions on both inter-thread and intra-thread non-determinism. Formally, the objective for the resource manager is:

$$\phi^2 = (\phi_{\mathcal{I}}^{inter} \wedge \phi_{\mathcal{I}}^{intra} \Rightarrow \phi_{\mathcal{I}}^{goal}) \,. \tag{8.1}$$

8.2.1 Stochastic Games

We base the synthesis of the resource manager on *stochastic games*. As we will see in detail later, 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 [CdAH04]. Given a finite set A, we denote by $\mathrm{Dist}(A)$ the set of probability distributions over A. For $d \in \mathrm{Dist}(A)$ we let $\mathrm{Supp}(d) = \{a \in A \mid d(a) > 0\}$. Given $a \in A$ we denote by $\delta(a) \in \mathrm{Dist}(A)$ the probability distribution that associates probability 1 with a, and 0 to all other elements of A. We also denote by $\mathit{Uniform}(A)$ the probability distribution that associates probability $1/|A|$ to every element of A.

Definition 5 A *two-player game* $G = (S, \mathit{Moves}, \Gamma_1, \Gamma_2, \delta, \phi)$ consists of a set of states S, of a set of moves Moves, of two mappings $\Gamma_1, \Gamma_2 : S \mapsto 2^{\mathit{Moves}} \setminus \emptyset$ associating to each state s and player $i \in \{1, 2\}$ the set of moves $\Gamma_i(s)$ that player i can play at s, a (probabilistic) destination function $\delta : S \times \mathit{Moves}^2 \mapsto \mathrm{Dist}(S)$, which associates with each $s \in S$ and $m_1 \in \Gamma_1(s)$, $m_2 \in \Gamma_2(s)$, a probability distribution $\delta(s, m_1, m_2)$ over the successor state. Finally, the winning condition ϕ is a measurable subset of S^ω.

For $i \in \{1,2\}$, we say that G is an *i-Markov decision process* (i-MDP) [Der70] if $|\Gamma_{3-i}(s)| = 1$ at all $s \in S$; 1-MDPs are also called simply MDPs. A *strategy* for player $i \in \{1,2\}$ in a game $G = (S, Moves, \Gamma_1, \Gamma_2, \delta)$ is a mapping $\pi_i : S^+ \mapsto \mathrm{Dist}(Moves)$, such that for all $\sigma \in S^*$ and $s \in S$, we have $\pi_i(\sigma s)(m) > 0$ implies $m \in \Gamma_i(s)$. We denote by Π_1, Π_2 the set of strategies for players 1 and 2 respectively. Once the strategies π_1 and π_2 are fixed, the game is reduced to an ordinary stochastic process, and the probabilities of all measurable events (which include all ω-regular properties [Tho90]) are defined (see e.g. [FV97]). 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 $\Pr_s^{\pi_1,\pi_2}(\phi) = 1$. The *size* of a game is defined by $|G| = \sum_{s \in S} \sum_{m_1 \in \Gamma_1(s)} \sum_{m_2 \in \Gamma_2(s)} |\,\mathrm{Supp}(\delta(s, m_1, m_2))|$.

8.2.2 The Scheduling Game

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 $\mathcal{I} = (R, \nu^0, (I_1, \ldots, I_n))$ which gives rise to a joint interface $M_{\mathcal{I}} = (R, S, E, s^{\mathrm{init}}, \lambda, \theta)$.

Definition 6 The *two-player game* corresponding to a system $\mathcal{I}$ consists of a tuple $G^2 = (S, Moves, \Gamma_1, \Gamma_2, \delta, \phi^2)$, where $Moves = S \cup \{\bot\}$ 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 $\Gamma_1(s) = Isucc(s) \cup \{\bot\}$ and $\Gamma_2(s) = Osucc(s)$.

• If $Osucc(s) = \emptyset$, then $\Gamma_1(s) = Isucc(s)$ and $\Gamma_2(s) = \{\bot\}$.

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

• For $t \in Isucc(s)$, we have $\delta(s, t, *) = \delta(t)$;

• for $t \in Osucc(s)$, we have $\delta(s, \bot, t) = \delta(t)$.

The manager synthesis problem can thus be phrased as the problem of finding a winning strategy in G^2. We say that the system is *schedulable* if $s^{\text{init}} \in Win(G^2)$. One can see that this goal is *upward-closed*, so that memoryless, but randomized, strategies suffice to win the game [CdAH04].

8.2.3 Practical Solution of the Scheduling Game

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 $\mathcal{I}$ [Tho95]. Thus, this approach leads to an inefficient algorithm. Instead, we show 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 show that 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 $G^{2.5}$. We show that this winning strategy, under many cases of practical importance, is also a winning strategy for the original game G^2. In all cases, we show that 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.

Definition 7 Given the game $G^2 = (S, \mathit{Moves}, \Gamma_1, \Gamma_2, \delta, \phi^2)$, the games $G^{2.5} = (S, \mathit{Moves}', \Gamma_1, \Gamma_2', \delta', \phi^{2.5})$ and $G^{1.5} = (S, \mathit{Moves}, \Gamma_1, \Gamma_2'', \delta'', \phi^{1.5})$ are obtained as follows. We have $\mathit{Moves}' = \mathit{Moves} \cup \{1, \ldots, n\}$, $\phi^{2.5} = \phi_{\mathcal{I}}^{\mathit{inter}} \Rightarrow \phi_{\mathcal{I}}^{\mathit{goal}}$, and $\phi^{1.5} = \phi_{\mathcal{I}}^{\mathit{goal}}$. The functions Γ_2', δ' and Γ_2'', δ'' coincide with Γ_2, δ, except that:

- For all $s \in S$ such that $|\mathit{Osucc}(s)| > 1$, 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 $\delta'(s, \bot, i) = \mathit{Uniform}(\{t \in \Gamma_2(s) \mid \theta(s,t) = i\})$.
- For all $s \in S$, we let $\Gamma_2'' = \{\bot\}$, and we let $\delta''(s, \bot, \bot) = \mathit{Uniform}(\mathit{Osucc}(s))$.

First, we show 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 $E_\pi = \{(s,t) \mid \pi(s)(t) > 0$ or $\pi(s)(\bot) > 0$ and $\lambda(s,t) \in \mathit{Acts}^O[R]\}$. 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 $F = \{(s,t) \in E_\pi \mid s \in C\}$ [dA97]. We say that thread k is *finished* in a state s if $loc_k(s)$ 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 [CdAH04]; we denote by $Win(G^{1.5})$ the set of winning states of $G^{1.5}$. By the results of [dA97, dAHK98], 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 $s \in Win(G^{1.5})$, we let $\pi^*(s) = Uniform(\{m \in \Gamma_1(s) \mid \forall t \in S.(\delta''(s,m,\perp)(t) > 0 \Rightarrow t \in Win(G^{1.5}))\})$. π^* is arbitrarily defined on states $s \in S \setminus Win(G^{1.5})$.

Theorem 33 *The strategy π^* is winning in $G^{1.5}$, and can be computed in time $O(|G^{1.5}|^2)$.*

In the next section, we present some technical lemmas, that are later used to show the relationships between the different versions of the scheduling game.

8.2.4 Properties

In order to argue that π^* is winning not only in $G^{1.5}$, but also in $G^{2.5}$, we need to develop some properties of π^* and $M_{\mathcal{I}}$. First, we state a simple property of $M_{\mathcal{I}}$.

Lemma 14 *In $M_{\mathcal{I}}$, there is no loop made entirely of input edges, and there is no loop made entirely of output edges.*

Proof. The first statement is due to the fact that each input edge decreases the value of a resource. The second statement is due to the fact that resource requests ($w_x!$) are immediately followed by an input edge, and resource releases ($r_x!$) increase the value of a resource. ■

We now show that, in $M_{\mathcal{I}}$, input and output moves commute, as they are independent. In the following, we write $s \xrightarrow[i]{x} t$ to signify that $(s,t) \in E$, $\lambda(s,t) = x$ and $\theta(s,t) = i$.

Lemma 15 *For all $s, s_1, s_2 \in S$, if $s \xrightarrow[i]{\alpha!} s_1$ and $s \xrightarrow[j]{\beta?} s_2$, then there is $t \in S$ such that $s_2 \xrightarrow[i]{\alpha!} t$ and $s_1 \xrightarrow[j]{\beta?} t$.*

Proof. First, notice that $i \neq j$, as input edges have no siblings in their respective thread (see Definition 2). Second, the value of each resource in s_1 is at least as much as it is in s. Thus, there is a state t such that $s_1 \xrightarrow[j]{\beta?} t$. In s_2, the value of a certain resource is lower than it is s. However, output edges are not affected by the value of the resources, so there is a state t' such that $s_2 \xrightarrow[i]{\alpha!} t'$, and by construction of $M_{\mathcal{I}}$, we have $t = t'$. ■

The following lemma states an equivalent commutativity property for outputs belonging to different threads.

Lemma 16 *For all $s, s_1, s_2 \in S$, if $s \xrightarrow[i]{\alpha!} s_1$ and $s \xrightarrow[j]{\beta!} s_2$, with $i \neq j$, then there is $t \in S$ such that $s_2 \xrightarrow[i]{\alpha!} t$ and $s_1 \xrightarrow[j]{\beta!} t$.*

Proof. Since output edges can either decrease resource usage (in the case of resource release actions), or leave resource usage unchanged (in the case of resource request actions), $\alpha!$ will still be enabled from s_2, and $\beta!$ will be enabled from s_1; moreover, by construction of $M_{\mathcal{I}}$, we have $s_2 \xrightarrow[i]{\alpha!} t$ and $s_1 \xrightarrow[j]{\beta!} t$ for the same t. ■

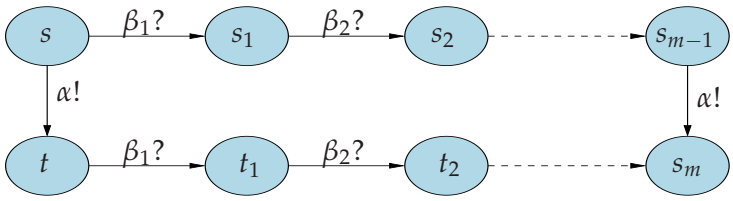


Figure 8.4: Outputs cannot link winning states to losing ones.

The following lemma shows that, in $G^{1.5}$, an edge labeled with an output cannot connect a winning state to a losing state.

Lemma 17 *Let $s \in Win(G^{1.5})$ and $s \xrightarrow[i]{\alpha!} t$. Then, $t \in Win(G^{1.5})$.*

Proof. Suppose that, starting from s, we keep following winning inputs, as long as there is a winning input in the current state. By Lemma 14, we must eventually reach a state s_{m-1} that has no winning inputs. By repeated applications of Lemma 15, the output $\alpha!$ is still enabled in s_{m-1}.

Summarizing, as illustrated in Figure 8.4, we can find a path $\sigma = ss_1 \dots s_m$ such that *(i)* all states in σ are winning, *(ii)* all edges in σ except the last one are labeled with inputs, and *(iii)* the last edge (s_{m-1}, s_m) is labeled with $\alpha!$.

Again by repeated applications of Lemma 15, from t we can mimic the path σ, by taking similar input edges, finally reaching s_m. We obtain the conclusion that t can reach the winning state s_m by means of input edges only. So, t itself is a winning state. ■

In the following, we say that a path is *in* $Win(G^{1.5})$ to mean that it is a path in $G^{1.5}$ made entirely of winning states. We now introduce a binary relation "$\sqsubseteq$" over the set of winning states of $G^{1.5}$. For all $s, s' \in Win(G^{1.5})$, let $s \sqsubseteq s'$ if and only if there is a path σ in $Win(G^{1.5})$

that goes from s to s' using only output edges. The following lemma shows that if $s \sqsubseteq s'$ and an input edge is winning from s, the corresponding input edge from s' is also winning.

Lemma 18 *Let $s \sqsubseteq s'$. For all $t \in Win(G^{1.5})$ such that $s \xrightarrow[i]{\alpha?} t$ there is $t' \in Win(G^{1.5})$ such that $s' \xrightarrow[i]{\alpha?} t'$ and $t \sqsubseteq t'$.*

Proof. Let σ be a path from s to s' in $Win(G^{1.5})$ that contains only outputs edges. By repeated applications of Lemma 15, we can take a similar path σ' from t, leading to a state t' such that $t \sqsubseteq t'$. Moreover, by construction $s' \xrightarrow[i]{\alpha?} t'$. By applying Lemma 17 to all edges in σ' we obtain that, since t is winning, t' is also winning. ■

The following lemma will be instrumental in showing that π^* is a winning strategy also in $G^{2.5}$.

Lemma 19 *There is $p > 0$ such that, for all $s \in Win(G^{1.5})$, if in $Win(G^{1.5})$ there is an acyclic path from s to a state s', then using π^* in $G^{2.5}$, for all player 2 strategies, with probability at least p, starting from s the game reaches a state t' such that $s' \sqsubseteq t'$.*

Proof. Let ρ be the path from s to s'; the proof is by induction on the length of ρ. Fix an arbitrary strategy of player 2. For $|\rho| = 0$, the result trivially holds. As induction hypothesis, assume that there is a path ρ from s to s' in $Win(G^{1.5})$, and assume that using π^* in $G^{2.5}$ we can reach from s a state t' such that $s' \sqsubseteq t'$ with positive probability. Let σ be the sequence of output actions leading from s' to t', and let θ be the path from s to t'. We will show that, if we prolong ρ by one step, reaching s'', then we can prolong θ by 0 or more steps, obtaining a path θ'' to t'', such that $s'' \sqsubseteq t''$, and such that θ'' is followed with positive bounded probability in $G^{2.5}$. Notice that, due to Lemma 16, outputs of different threads

commute. Hence, we can consider the ordering in σ restricted to outputs belonging to the same thread. Equivalently, rather than σ, we can reason about the collection of sequences of output actions $\{\sigma_i\}_{i=1..n}$, where σ_i represents the sequence of actions of thread i along σ. There are then three cases, depending on the step $s's''$:

- Assume that $s' \xrightarrow[i]{\alpha?} s''$, for some α and $i \in \{1, \ldots, n\}$. By Lemma 18, there is also a winning step $t' \xrightarrow[i]{\alpha?} t''$, and a path from s'' to t'' that uses the sequence of output actions σ. As π^* takes this step with positive probability, this leads to the result.

- Assume that $s' \xrightarrow[i]{\alpha!} s''$, for some α and $i \in \{1, \ldots, n\}$; assume also that α does not appear in σ_i. By Lemma 16, from t', the same output α is enabled, so that π^* will play with positive probability action $\perp$, and in $G^{2.5}$ some output β will occur. If β belongs to thread i, then with positive probability (according to the randomized resolution of intra-thread nondeterminism) it must be $\beta = \alpha$, and the destination state t'' will be related to s'' again by σ. If β does not belong to thread i, we add β to σ. By Lemma 16 we have that output α is still enabled from the destination state after β, so that π^* will again play $\perp$ from the destination with positive probability. Eventually, an output belonging to thread i will occur, as by Lemma 14 there cannot be an infinite path consisting entirely of output actions.

- Assume that $s' \xrightarrow[i]{\alpha!} s''$, for some α and $i \in \{1, \ldots, n\}$; assume also that α appears in σ_i. Then, with positive probability (due to the resolution of inter-thread nondeterminism), α will be the first action of σ_i. We remove α from σ_i, obtaining a shorter σ'; we have that $s'' \sqsubseteq t'$, and s'' and t' are related by σ'.

The existence of a constant bound $p > 0$ derives from the fact that the length of ρ, and the size of σ, are bounded, as is the number of ways in which intra-thread nondeterminism can be resolved. ■

8.2.5 Comparing Games

We now proceed to prove that the strategy π^* is also a winning strategy for $G^{2.5}$.

Theorem 34 *The strategy π^* is winning in game $G^{2.5}$, and $Win(G^{1.5}) = Win(G^{2.5})$.*

Proof. For $i \in \{1, \ldots, n\}$ and $s \in Win(G^{1.5})$, we say that thread i is enabled in s if there is an edge $(s, t) \in E$ such that $\theta(s, t) = i$ and $t \in Win(G^{1.5})$. Note that this definition is correct, as by Lemma 17 output edges are always winning.

For $i \in \{1, \ldots, n\}$ and $s^* \in Win(G^{1.5})$, we have to prove that, using π^* in $G^{2.5}$ and starting from s^*, with positive probability a state is reached where thread i is enabled. Since this is true of every winning state s^*, and since the game stays forever in the set of winning states, it follows that the probability of enabling thread i infinitely often, ensuring that it is also taken infinitely often, is in fact 1.

If in s^* the next action of thread i is an output, then by Lemma 17 it is available directly from s^*. Thus, assume in the following that the next action of thread i in s^* is an input. Since s^* is winning in $G^{1.5}$, there is a path in $Win(G^{1.5})$ from s^* to a state t^* where thread i is enabled. By applying Lemma 19 to states $s = s^*$ and $s' = t^*$, we obtain that in $G^{2.5}$ from s^* with positive probability a state t' is reached such that $t^* \sqsubseteq t'$, and therefore thread i is enabled in t'. ■

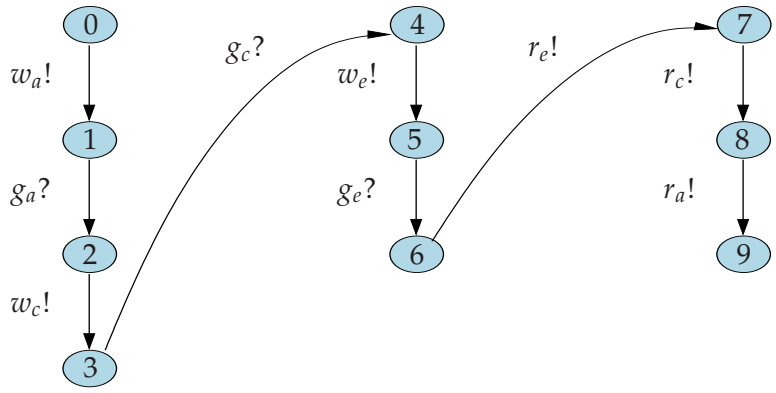


Figure 8.5: Thread interface from Example 13.

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 show 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 $\bigwedge_{u \in S_i} \bigwedge_{v \in Osucc_i(u)} \Box\Diamond from_i^u \Rightarrow \Box\Diamond take_i^{u,v}$. Moreover, for a finite path σ and a resource $x \in R$, let $decr(x, \sigma) = |\{(s,t) \in \sigma \mid \lambda_i(s,t) = g_x?\}|$, $incr(x, \sigma) = |\{(s,t) \in \sigma \mid \lambda_i(s,t) = r_x!\}|$, and $balance(x, \sigma) = incr(x, \sigma) - decr(x, \sigma)$. We say that I_i is *mutex-correct* if for all finite traces σ

and all mutexes $x \in R_i$, it holds $balance(x, \sigma) \in \{-1, 0\}$.

Definition 8 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 $balance(x, \sigma') = 0$ 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 35 *For PMF systems, π^* is winning in game G^2, and $Win(G^{1.5}) = Win(G^2)$.*

Proof. By Theorem 34, we have that π^* is winning in game $G^{2.5}$. We show that for all $i \in \{1, \ldots, n\}$ and $s^* \in Win(G^{2.5})$, using π^* in G^2 and starting from s^*, with positive probability a state is reached where thread i is enabled. Similar to the proof of Theorem 34, since this is true of every winning state s^*, and since the game stays forever in the set of winning states, it follows that the probability of enabling thread i infinitely often, ensuring that it is also taken infinitely often, is in fact 1.

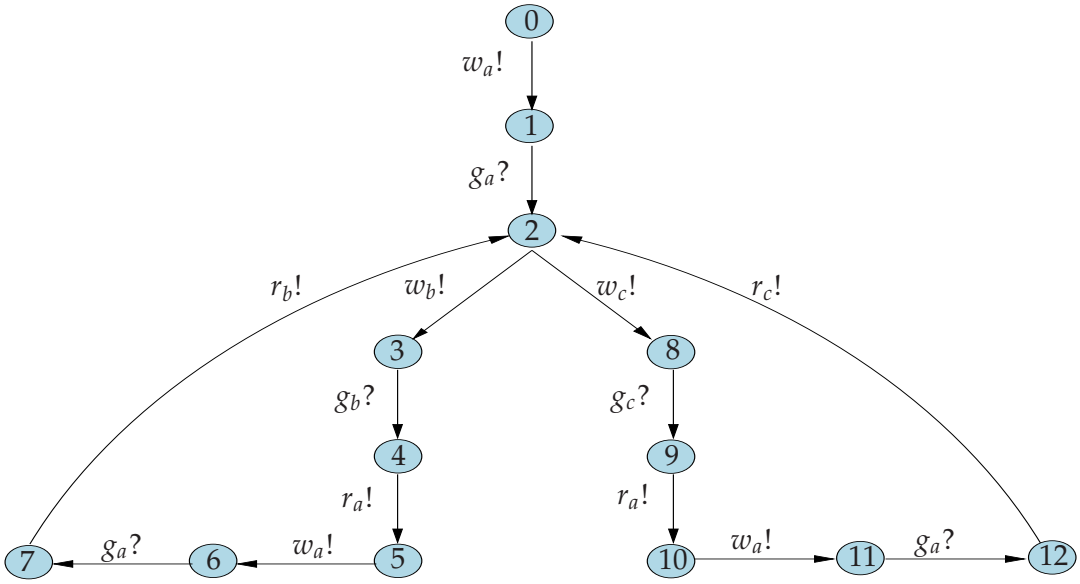
Consider an arbitrary strategy of player 2 such that $\phi_{\mathcal{I}}^{intra} \wedge \phi_{\mathcal{I}}^{inter}$ holds. By the definition of PMF systems, for all $i \in \{1, \ldots, n\}$, for all fair infinite paths σ, there exists infinitely many occurrences of a state where i is not holding any resource. In $G^{2.5}$, by the resolution of intra-thread non-determinism, for all paths σ starting at s^*, we have that they are fair and that for all threads $i \in \{1, \ldots, n\}$, since there exists infinitely many occurrences of a state where i is not holding any resource, we can reach a state t_i^*, where thread i is enabled with positive probability. This implies that in G^2, given $\phi_{\mathcal{I}}^{intra}$ holds,

ensuring all paths are fair, starting at state s^*, the state t_i^*, where thread i is enabled, is visited with positive probability as required. Therefore, for PMF systems, π^* is winning in G^2. Further, since $Win(G^{1.5}) = Win(G^{2.5})$ by Theorem 34, and we have shown that $Win(G^{2.5}) = Win(G^2)$, we conclude $Win(G^{1.5}) = Win(G^2)$. ■

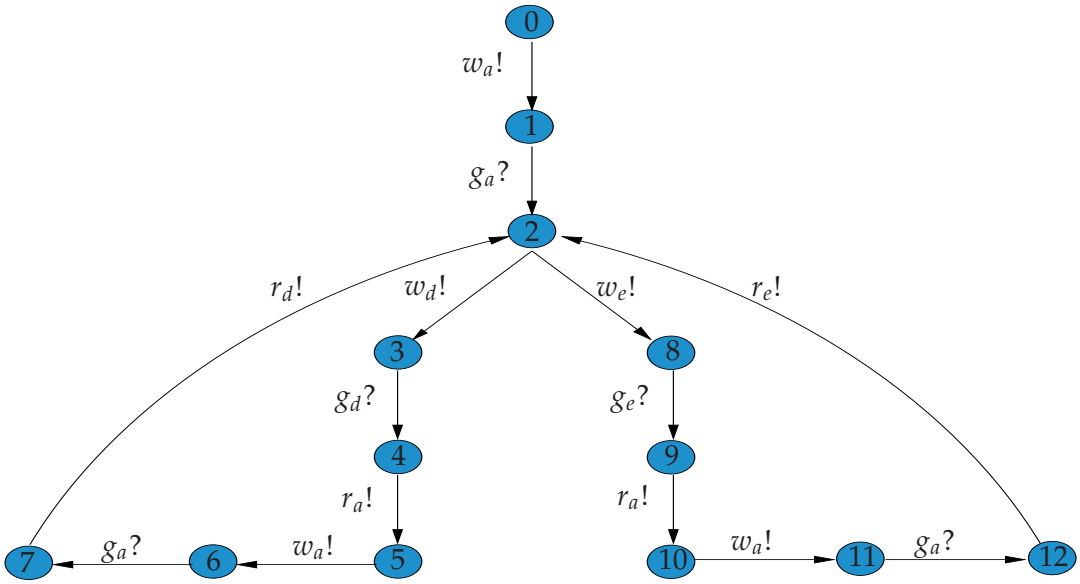
The next example shows that π^* may not be winning in G^2, when the system is not PMF. Notice that a rather special thread structure is required for this to happen.

Example 13 Consider the 5-mutex, 3-thread system $(\{a, b, c, d, e\}, \nu^0, (I_1, I_2, I_3))$ where I_1 is as in Figure 8.6(a), I_2 is as in Figure 8.6(b), and I_3 is as in Figure 8.5. First, at all times after thread 1 reaches state 2, it will always own at least one mutex among $\{a, b, c\}$. Similarly, thread 2 will always own at least one of $\{a, d, e\}$. For this reason, the system is not PMF. However, the initial state $(0, 0, 0, \nu^0)$ of $G^{1.5}$ is winning. Clearly, threads 1 and 2 can make infinite progress, since they only share mutex a, and they both release said mutex periodically. It remains to show that under the most general winning strategy π^*, thread 3 is allowed to perform its critical region (i.e. state 6) with probability 1. In $G^{1.5}$ (and $G^{2.5}$) the nondeterminism that threads 1 and 2 exhibit in state 2 is resolved by a uniform distribution. So, while making infinite progress, with probability 1 those threads will acquire mutexes b and d at the same time, thus leaving mutexes c and e free. At that point, as soon as mutex a is released, thread 3 can safely execute its critical region, by acquiring mutexes a, c, e.

On the other hand, in game G^2 threads 1 and 2 can cooperate in order to never release both c and e at the same time. When thread 1 is in state 2, thread 2 can only be in state 6 or 11 (because those are the only states where thread 2 does not hold a). So, player 2



(a)



(b)

Figure 8.6: Thread interfaces from Example 13.

can choose to acquire c when thread 2 is in 6 (thus holding d) and acquire b when thread 2 is in 11 (thus holding e). This ensures that c and e are never free at the same time. Now, consider a state where a is free. Giving a to thread 3 inevitably leads to a deadlock, because thread 3 needs c and e before releasing a, and either of them is currently owned and will not be released before a is. ■

Our next result, useful for threads that may use semaphores, enables us to establish whether the strategy π^* is winning also for G^2. To develop the result, note that the game G^2, once player 1 fixes strategy π^*, is a 2-MDP. For such 2-MDPs, we can compute in polynomial time the set of winning states for player 2 with respect to the complementary goal $\neg\phi^2$ using an algorithm that is a modified version of the algorithm proposed in [CdAH05] for Streett MDPs. This leads to the following result.

Theorem 36 *We can check in time* $O\left(|G^2|^2 \cdot n \cdot \sum_{i=1}^{n} |E_i|\right)$ *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 $|G^2|$.

8.3 Towards Efficient Resource Managers

The strategy π^*, even when winning, may not be an efficient strategy in practice. According to it, the resource manager would issue $\perp$ (wait for a resource request or release) with positive probability when there are input moves that are available and winning. First, this potentially reduces CPU utilization. In fact, other things being equal, it is better to grant immediately as many resource requests as possible: this ensures that the OS scheduler has the widest choice of threads to execute on the CPU, helping to avoid idle time when all available threads are blocked, e.g., waiting for I/O. More importantly, as a consequence of how we abstract thread interfaces, there is no guarantee that a thread whose next action is an output will issue that output within a short amount of time. For instance, the next resource request may be issued only after some user input has occurred.

In this section, we propose several improvements to π^*, aimed at reducing the number of times when the manager issues $\perp$ when input actions are available.

Maximal progress and critical progress strategies. The simplest idea consists in issuing $\perp$ only in the states $S^! = \{s \in S \mid \pi^*(s)(\perp) = 1\}$ where $\perp$ is the only winning move: this corresponds to waiting for output moves only when no resource can be granted. This idea leads to the *maximal progress strategy* π^{p}, defined by $\pi^{\mathrm{p}}(s) = \delta(\perp)$ for $s \in S^!$, and $\pi^{\mathrm{p}}(s) = \mathit{Uniform}(\mathrm{Supp}(\pi^*(s)) \setminus \{\perp\})$ otherwise. Unfortunately, the maximal progress strategy is not always winning, as the following example demonstrates.

Example 14 Consider the 3-thread system $(\{a, b\}, \{a \mapsto 1, b \mapsto 1\}, (I_1, I_2, I_3))$ where I_1 and I_2 are as in Figure 8.7(a), while I_3 is as in Figure 8.7(b). Figure 8.7(c) shows a fragment of the corresponding joint interface. Let us analyze this fragment as part of G^2, and assume

that player 1 employs π^{p}. One can check that, starting from the initial state $(0,0,0,\nu^0)$, player 2 can steer the game to state $(5,1,1,\nu)$, where $\nu = \{a \mapsto 0, b \mapsto 1\}$. At this point, all of the edges, except for the dashed ones, can be taken under π^{p}. The objective for the player 1 is to reach one of the states labeled as "good", as in those states thread 3 can make progress without risking a deadlock. However, player 2 can steer the game away from the two good states, thus reaching $(1,5,1,\nu)$ with certainty. Since $(1,5,1,\nu)$ is symmetrical w.r.t. $(5,1,1,\nu)$, this strategy enables player 2 to keep thread 3 starving forever. Thus, π^{p} is not a winning strategy in this game. The same applies to $G^{2.5}$, since the threads under consideration have no inter-thread non-determinism.

It should be noted that the situation is different in $G^{1.5}$. Since all output edges happen uniformly at random, π^{p} is winning in this case, as state $(0,0,1,\nu^0)$ is eventually reached with probability 1. ■

The example above suggests that sometimes, as in state $(5,1,1,\nu)$, it is necessary to wait for output actions, even when there are resources that are ready to be granted. The problem of waiting for outputs, as mentioned earlier, is that in general there is no guarantee that the outputs will be generated in a timely fashion. However, in mutex-only systems, we can assume that when a thread holds a mutex it will generate an output in a timely fashion, either to release the mutex, or to request another mutex. This captures the idea that, in well-written code, critical regions have short durations. Based on this idea, we let S^{c} be the set of states of a mutex-only system where there is some thread holding a mutex, and we propose a strategy that waits for outputs only in S^{c}. We define the *critical progress strategy* π^{c} by letting, for all $s \in S$, $\pi^{\mathrm{c}}(s) = \pi^*(s)$ if $s \in S^{\mathrm{c}}$ or $s \in S^{!}$, and $\pi^{\mathrm{c}}(s) =$

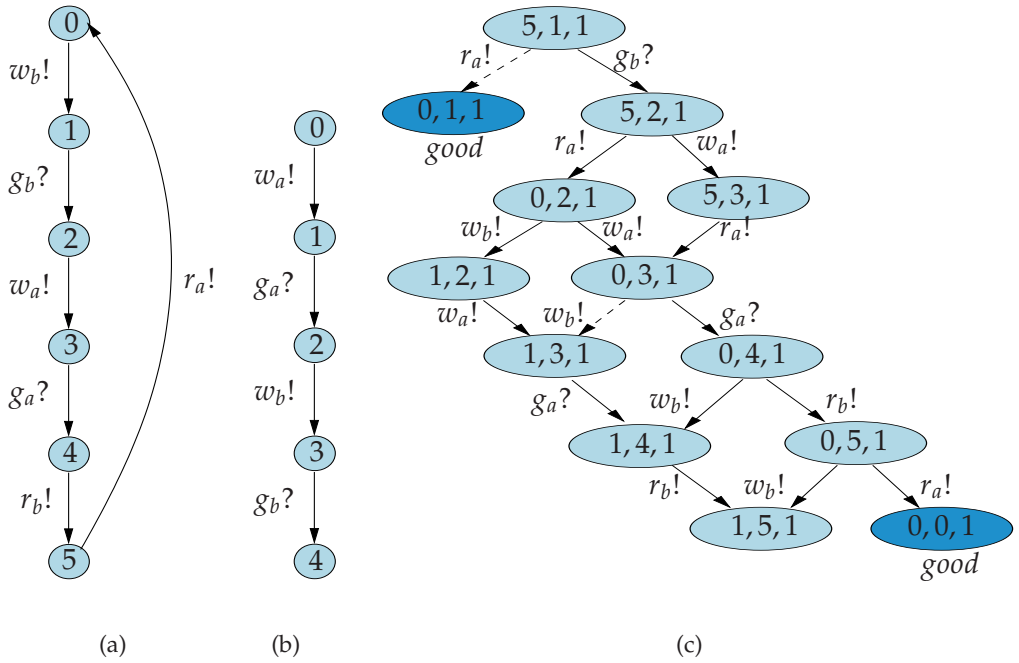


Figure 8.7: A system where the maximal progress strategy is not winning.

Uniform$(\text{Supp}(\pi^*(s)) \setminus \{\bot\})$ otherwise. The following result shows that, for PMF systems, π^c is an efficient resource manager strategy.

Theorem 37 *In a PMF system π^c is winning for G^2.*

Proof. For all states $s \in S^c \cup S^!$, since $\pi^c(s) = \pi^*(s)$, given π^* is winning in G^2 for PMF systems by Theorem 35, we have π^c is winning in G^2. For all states $s \in S \setminus (S^c \cup S^!)$, given π^* is winning in G^2, all moves in $\text{Supp}(\pi^*(s))$ are winning. As all threads are waiting for a resource at s by the definition of S^c, and choosing $\bot$ is necessary only when some thread is holding a resource, we have $\pi^c(s) = \textit{Uniform}(\text{Supp}(\pi^*(s)) \setminus \{\bot\})$ is winning in G^2. ■

Efficient strategies for systems with semaphores. A natural extension of π^c to systems with semaphores is a strategy that waits for outputs only when there is at least one thread

waiting for a resource that is not available (so that another thread must be holding a resource, and it may be reasonable to expect an output action in a timely manner). Unfortunately, there are examples showing that such an extension is not winning in general. We discuss two related strategies that are winning, and efficient, for systems with semaphores.

To obtain our first strategy, we reason as follows. Once a memoryless strategy $\pi \in \Pi_1$ is fixed, the game G^2 is equivalent to a 2-MDP $G^2(\pi)$. If an end-component in this 2-MDP is not fair, that is, if there is a thread k that is neither finished, nor progresses in the end component, then it can be seen that thread k must be stuck waiting for an input (a resource) at all states of the end component. This suggests to skip $\perp$ (waiting for outputs) only when no thread is blocked: in this way, if the strategy differs from π^* by cutting $\perp$, it can do so only in a winning component. Precisely, for $s \in S$ we let $Succ(s, \pi^*) = \{t \in S \mid \exists m_1 \in \Gamma_1(s).\exists m_2 \in \Gamma_2(s).(\pi^*(m_1) > 0 \wedge \delta(s, m_1, m_2)(t) > 0)\}$ be the set of possible successors of s according to π^*, and we let $S^{\mathrm{b}} = \{s \in S \mid \exists k \in [1..n].\forall t \in Succ(s, \pi^*).\theta(s, t) \neq k\}$ be the set of states where some thread is blocked. For $s \in S$, we then define π^{b} by $\pi^{\mathrm{b}}(s) = \pi^*(s)$ if $s \in S^{\mathrm{b}} \cup S^!$, and $\pi^{\mathrm{b}}(s) = \mathit{Uniform}(\mathrm{Supp}(\pi^*(s)) \setminus \{\perp\})$ otherwise.

Theorem 38 *The strategy π^{b} is winning in G^2 iff π^* is winning in G^2.*

Proof. In one direction, if π^{b} is winning in G^2, then not skipping $\perp$ in states where none of the threads are holding a resource is also winning in G^2 and hence π^* is winning in G^2. In the other direction, similar to the proof of Theorem 37, if π^* is winning in G^2, then cutting $\perp$ in states where no thread is holding a resource is winning in G^2. Hence π^{b} is winning in G^2. ■

Finally, we can obtain an efficient strategy *with memory* as follows. We say that a thread k is *bypassed* whenever it is waiting for an input, and the scheduling strategy does not give that input. Then, given a *bypass bound* $M \in \mathbb{N}$, we can construct a strategy π^{p}_M as follows. For each thread $k \in [1..n]$, π^{p}_M keeps track of the number b_k of times for which thread k has been consecutively bypassed. As long as $b_k \leq M$ for all $1 \leq k \leq n$, the strategy π^{p}_M behaves like π^{p}. When $b_k > M$ for some $k \in [1..n]$, on the other hand, π^{p}_M reverts to behave like π^*, thus sometimes waiting for outputs when there are input actions (resource grants) that could be taken. The idea, informally, is as follows: if a thread is bypassed for a large number of consecutive times, it means that some other threads may be holding the resources it needs to proceed. Favoring output actions (among which are resource releases) enables the system to reach a state where the bypassed thread can be finally granted the resource it needs.

Theorem 39 *For all $M \in \mathbb{N}$, we have that π^{p}_M is winning in G^2 iff π^* is winning in G^2.*

Proof. In one direction, consider an arbitrary fixed bound $M \in \mathbb{N}$ and the resulting strategy π^{p}_M that is winning in G^2 from a starting state s^*. We show that π^* is winning in G^2. For all states $s \in S^!$, where the only winning move is $\bot$, since π^{p}_M and π^* will choose $\bot$, π^* is winning at s. If $b_i \leq M$ for all threads $i \in \{1, \ldots, n\}$ for all paths starting at s^*, then given π^{p}_M is winning, we can always reach a state t^*_i where thread i is enabled with positive probability. This implies using π^*, which only differs from π^{p}_M by playing $\bot$ with positive probability, we can again reach t^* with positive probability. Therefore, we have π^* is winning. If $b_k > M$ for some thread $k \in \{1, \ldots, n\}$ for some path starting at s^*, then as π^{p}_M reverts to π^* and π^{p}_M is winning, π^* is winning as well.

In the other direction, given π^* is winning in G^2 from a starting state s^*, if $M = 0$, then as π^{p}_M is the same as π^*, we have π^{p}_M is winning in G^2. Consider an arbitrary fixed $M > 0$. The strategy π^{p}_M differs from π^* by cutting $\perp$ in states $Win(G^2) \setminus S^!$ as long as $b_i \leq M$ for all threads $i \in \{1, \ldots, n\}$. Since the game always remains in $Win(G^2)$ and π^{p}_M reverts to π^* when $b_k > M$ for some thread $k \in \{1, \ldots, n\}$, given π^* is winning, we have π^{p}_M is also winning in G^2. ■

8.4 The Tool

We have developed a prototype tool called CYNTHESIS that realizes the theory hereby presented. The tool takes as input a C program, and it either produces a warning that the system is not schedulable (according to the definition in Section 8.2.2), or it outputs a custom resource manager encoded as a C program that can be compiled and linked to the original program. The result is an executable that is deadlock-free whenever the OS scheduler is fair, and the threads do not block for reasons other than resources (such as infinite loops). The tool is currently tailored to the eCos embedded OS [eco], but it can be easily modified to work with another OS.

To extract thread interfaces, the tool uses the CIL library [NMRW02] to build a control-flow graph (CFG) for each thread. For the purpose of this graph, function calls are treated as inlined. While building the CFG, each time a synchronization primitive is detected, edges labeled with the appropriate action are added to the thread interface, as follows: *(i)* calls to `mutex_unlock(x)` and `sem_post(x)` are represented by an edge labeled $r_x!$, and *(ii)* calls to `mutex_lock(x)` and `sem_wait(x)` are represented by a sequence of two

edges labeled with $w_x!$ and $g_x?$ respectively. The original calls are also automatically annotated with location information, to allow the resource manager to distinguish them at run-time. The graph is then minimized to remove transitions that do not involve resources.

Currently, in order for the tool to correctly identify resources, they must be declared as global variables and then used by their original names; we are working to add alias analysis to the tool to overcome this limitation. Once the thread interfaces are extracted, the tool solves the game $G^{1.5}$ and it outputs a custom resource manager in the form of compilable C code. The resource manager behaves like the strategy π^*, or optionally like one of the other winning strategies discussed in Section 8.3. In order to simulate the behavior of a strategy, the custom manager needs to know which winning moves are available at any given decision point. In turn, this means that it has to know in which state of the joint interface the system currently is, and what are the winning moves from that state. Rather than keeping a copy of the joint interface, which can be of exponential size in the number of threads, the manager keeps separate copies of the individual thread interfaces, along with the value of the resources. With this information, the manager is aware of all moves; all that remains to encode are the moves that are *not* part of the winning strategy: to do this, it suffices to store the set of losing states. As the number of losing states can grow exponentially with the number of threads, we encode the losing states using a BDD [Bry86], leading to a very compact representation. In Table 1, we report the result of some experiments, all run on a 2.4GHz Pentium 4 machine with 512Mb of memory. The threads involved in the test give rise to thread interfaces having between 5 and 12 states; apart from the resource primitives, the size of the source code of the threads has a negligi-

Number of threads(n)	$\lvert M_{\mathcal{I}} \rvert$	Number of bad states	BDD nodes	Time (sec.)
2	37	3	15	0.05
3	171	18	30	0.07
6	17496	2592	62	39
6	33120	5490	211	334

Table 8.1: Experiments.

ble effect on the running time of the tool, and it is irrelevant to the size of the synthesized manager and the BDD. The second column reports the number of states in the joint interface, and the last column reports the total time needed to synthesize the manager.

A Case Study

We conducted a more extensive test, consisting in analyzing a multi-threaded program implementing an ad-hoc network protocol for Lego robots. As illustrated in Figure 8.8, the program is composed of five threads, represented by ovals in the figure, that manage four message queues, represented as boxes in the figure.

Threads *user* and *generator* add packets to the *input* queue. The *router* thread removes packets from the *input* queue, and dispatches them to the other queues. Packets in the *user* queue are intended for the local node, so they are consumed by the *user* thread. Packets in the *broadcast* queue are intended for broadcast, and they are moved to the *output* queue by the *delay* thread, after a random delay, intended to avoid packet collisions dur-

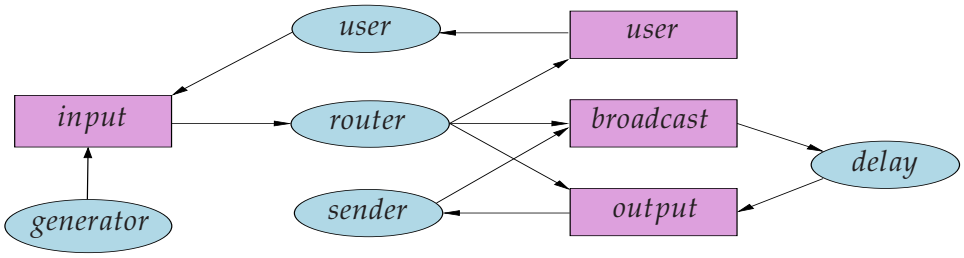


Figure 8.8: Scheme of an ad-hoc network protocol implementation.

ing broadcast propagations. Packets in the *output* queue are in transit to another node, so they are treated by the *sender* thread. Notice that if the *sender* fails to send a packet on the network, it puts it in the *broadcast* queue (even if it is not a broadcast packet), so that it will be re-sent after a delay.

Each queue is protected by a mutex, and two semaphores that count the number of empty and free slots, respectively. Altogether, the program employs 7 mutexes and 8 semaphores. By restricting all queues to having 1 slot, the resulting joint interface contains 400,000 states, and the tool terminates its analysis in about 7 minutes.

The tool found a deadlock that corresponds to the following situation. Suppose that queues *output* and *broadcast* are both full. Suppose also that the *sender* thread extracts a packet from *output* and tries to send it on the network. If the send fails, the thread will try to insert the packet in the *broadcast* queue. Since the latter is full, the *sender* thread will hang on a semaphore, waiting for an empty slot in *broadcast*. However, the only way a slot in *broadcast* can be emptied is for the *delay* thread to move a packet to *output*, which is still full. Therefore, the *sender* will hang forever, and the whole system will consequently block. Interestingly, the tool reports that there is a winning strategy in this situation. The strategy

consists in “slowing down” the router, preventing it from adding packets to *broadcast* if *output* is full, and vice versa.

Chapter 9

Discussion

In this chapter we present our conclusions and discuss possible future directions for the work presented in this thesis. We highlight possible applications of game relations and metrics and then cover extensions to our work on protocol synthesis and resource manager synthesis. We begin with game relations and metrics.

9.1 Game Relations and Metrics

We have shown theoretical applications of game metrics with respect to discounted and long-run average values of games. An interesting question regarding game metrics is related to their usefulness in real-world applications. We now discuss possible applications of game metrics.

State space reduction. The kernels of the metrics are the simulation and bisimulation relations. These relations have been well studied in the context of transition systems with applications in program analysis and verification. For example, in [KKZJ07] the authors

show that bisimulation based state space reduction is practical and may result in an enormous reduction in model size, speeding up model checking of probabilistic systems.

Security. Bisimulation plays a critical role in the formal analysis of security protocols. If two instances of a protocol, parameterized by a message m, are bisimilar for messages m and m', then the messages remain secret [CNP09]. The authors use bisimulation in probabilistic transition systems to analyze probabilistic anonymity in security protocols.

Computational Biology. In the emerging area of computational systems biology, the authors of [TK09] use the metrics defined in the context of probabilistic systems [DGJP99, vBW01a, vBW01b] to compare reduced models of *Stochastic Reaction Networks*. These reaction networks are used to study intra-cellular behavior in computational systems biology. The reduced models are Continuous Time Markov Chains (CTMCs), and the comparison of different reduced models is via the metric distance between their initial states. A central question in the study of intra-cellular behavior is estimating the sizes of populations of various species that cohabitate cells. The inter-cellular dynamics in this context is modeled as a stochastic process, representing the temporal evolution of the species' populations, represented by a family $(X(t))_{t\geq 0}$ of random vectors. For $0 \leq i < N$, N being the number of different species, $X_i(t)$ is the population of species S_i at time t. In [SW08], the authors show how CTMCs that model system dynamics can be reduced to Discrete Time Markov Chains (DTMCs) using a technique called uniformization or discrete-time conversion. The DTMCs are stochastically identical to the CTMCs and enable more efficient estimation of species' populations. An assumption that is made in these studies is that systems are spatially homogeneous and thermally equilibrated; the molecules are well stirred in a fixed

volume at a constant temperature. These assumptions enable the reduction of these systems to CTMCs and to DTMCs in some cases.

In the applications we have discussed, non-determinism is modeled probabilistically. In applications where non-determinism needs to be interpreted demonically, rather than probabilistically, MDPs or turn-based games would be the appropriate framework for analysis. If the interaction between various sources of non-determinism needs to be modeled simultaneously, then concurrent games would be the appropriate framework for analysis. For the analysis of these general models, our results and algorithms will be useful.

Open Problems. While we have shown polynomial time algorithms for the kernel of the simulation and bisimulation metrics for MDPs and turn-based games, the existence of a polynomial time algorithm for the kernel of both the simulation and bisimulation metrics for concurrent games is an open problem. We have introduced a metric in this thesis, which we call the total reward metric. While we show that the undiscounted total reward metric provides a bound for $q\mu$, $d\mu$, discounted, long-run average and total rewards, we are not aware of an extension to a quantitative calculus that would be a useful generalization of $q\mu$ and $d\mu$ and that would be logically characterized by the new metric. This is an open problem.

9.2 Protocol Synthesis

Fair non-repudiation protocols were invented to address the problem of digital contract signing over networks. It has been difficult to design these protocols and various

formal verification methods have been applied to verify their correctness. In this thesis we have approached the problem of designing these protocols as participant refinements, using automated synthesis, that satisfy fairness, abuse-freeness and are attack free. This provides the ability to separate designing the correct restrictions on participant behaviors from the cryptanalysis of the content of the messages, enabling a modular and structured approach to the design of protocols. We have shown that traditional co-operative or strictly adversarial notions of synthesis cannot be used to design these protocols. The right notion of synthesis is assume-guarantee synthesis based on the existence of secure equilibrium strategies in three-player turn-based games with a fair scheduler. We have shown that the solution set, P_{AGS}, of assume-guarantee synthesis includes the KM protocol and not the ASW or the GJM protocols and that the KM protocol could have been automatically synthesized from P_{AGS}. Using assume-guarantee analysis, we have presented a symmetric fair non-repudiation protocol, that is attack-free given a fixed TTP, assuming the channels between the agents and the TTP are operational. The symmetric non-repudiation protocol enables R to abort the protocol, thus engendering symmetrical abilities to both O and R. Using PCS, a cryptographic primitive which provides the designated verifier property, we can easily ensure that our symmetric protocol does not provide even partial information to either agent when the protocol is aborted. As an offshoot of our work we have shown that the TTP is needed for synthesis if we assume that the agents are not co-operative. Our work is the first application of the theory of controller synthesis to security protocols. The size of the state space and the efficiency of computing the assume-guarantee solution makes this approach attractive to the design of all protocols where the agents are assumed

to be rational, not deviating from the protocol if it hurts their payoff. Further, the subtleties in these protocols and the interplay between the protocol, participant and synthesis objectives can be effectively analyzed using techniques we present in this thesis. For future work, we would like to study the problem of automatic synthesis of multi-party fair exchange protocols and other security protocols. We would like to study the problem of synthesizing messages that are impervious to attacks. We would also like to extend the theory of assume-guarantee synthesis, as needed, to account for imperfect information.

9.3 Synthesis of Resource Managers

The progress objective $\phi_{\mathcal{I}}^{goal}$ states that each thread that is ready makes progress eventually, but the "eventual" time to make progress can be unbounded. A stronger and more desirable notion of progress is that of finitary progress, which states that each ready thread makes progress within bounded time. Let $\sigma \in S^{\omega}$ be an infinite path that can be taken in a joint interface $M_{\mathcal{I}}$; we take $\sigma[j]$ for $j = (0, 1, 2, \ldots)$ as the sequence of states in the path σ. Let $progress_i(s, t)$ be the predicate that is true for an edge (s, t) 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. The finitary progress goal $\phi_{\mathcal{I},f}^{goal}$ can be defined as follows:

$$\phi_{\mathcal{I},f}^{goal} = \bigcap_{i=1}^{n} (\Diamond final_i \cup \{\sigma \in S^{\omega} \mid \exists b \in \mathbb{N} \,.\, \forall j \geq 0 \,.\, \exists l \leq j \,.$$

$$(progress_i(\sigma[l], \sigma[l+1]) \wedge (j < l \leq (j+b)))\}) \,.$$

Intuitively, the winning set of paths for the resource manager is the set of paths such that in each path for every thread i, $progress_i(s, t)$ is true over edges that are never more than

b apart. We now show that the fairness assumption on inter-thread and intra-thread non-determinism is not sufficient to ensure finitary progress.

Consider again the example in Figure 8.3. From our earlier analysis of the example, the resource manager can give resources b and c to Thread 3 only when either Thread 1 is in its *then* branch or Thread 2 is in its *else* branch. As long as Thread 1 and Thread 2 are in their *else* and *then* branches respectively, the resource manager does not have a strategy to ensure that Thread 3 enters its critical region. A fair strategy to resolve intra-thread non-determinism is as follows. The strategy is played in rounds. In round i, Thread 1 and Thread 2 collude such that Thread 1 is in the *else* branch of its conditional statement and Thread 2 is in the *then* branch of its conditional statement for at least i executions of the while loop. Thread 1 then enters its *then* branch or Thread 2 enters its *else* branch once before proceeding to round $i+1$. For example, let the conditional expression exp be $power\_of\_2(y)$ where y is a variable shared by Thread 1 and Thread 2 that is initially 0 and is incremented by 1 in Thread 1 each time the while loop executes. The function $power\_of\_2(y)$ returns 1 if y is a power of 2 and 0 otherwise. For any bound $b > 0$, there exists a $y > 0$ with $2^{y-1} \leq b < 2^y$ such that the resource manager has no strategy to allocate resources b and c to Thread 3 for $2^y > b$ executions of the loop in Thread 1 and Thread 2. It follows that $\phi_{\mathcal{I}}^{inter} \wedge \phi_{\mathcal{I}}^{intra} \Rightarrow \phi_{\mathcal{I},f}^{goal}$ fails. On the other hand, if Thread 1 and Thread 2 satisfy the stronger notion of finitary fairness, where both branches of the conditional statement will be executed within a bound $b > 0$, then as soon as Thread 1 enters its *then* branch or Thread 2 enters its *else* branch, the resource manager can allocate b and c to Thread 3 and ensure that Thread 3 makes progress within the bound b thus satisfying its goal $\phi_{\mathcal{I},f}^{goal}$. We

now formulate the finitary fairness assumption on intra-thread non-determinism as:

$$\phi^{intra}_{\mathcal{I},f} = \bigcap_{i=1}^{n} \{\sigma \in S^{\omega} \mid \exists b \in \mathbb{N} \,.\, \forall j \geq 0 \,.\, \forall u \in S_i \,.\, \forall v \in Osucc_i(u) \,.\, \exists l \in \mathbb{N} \,.$$

$$from^{u}_{i}(\sigma[j], \sigma[j+1]) \Rightarrow take^{u,v}_{i}(\sigma[l], \sigma[l+1]) \wedge (j < l \leq (j+b))\} \,.$$

Intuitively, the finitary assumption $\phi_{\mathcal{I},f}$ is the set of paths such that in each path, if a thread visits a state where it has multiple output successors, then each output successor can be ignored at most k times. A similar definition applies to the finitary fairness assumption $\phi^{inter}_{\mathcal{I},f}$ on inter-thread non-determinism. The ammended objective for the automatic synthesis of resource managers is then:

$$\phi^{2}_{f} = (\phi^{inter}_{\mathcal{I},f} \wedge \phi^{intra}_{\mathcal{I},f} \Rightarrow \phi^{goal}_{\mathcal{I},f}) \,. \tag{9.1}$$

We would like to study finitary progress under finitary fairness conditions on the sources of non-determinism for future work.

Appendix

Appendix A

Completing Protocol Synthesis

A.1 Appendix

Translating protocol models to process models. We now present a translation from the protocol model introduced in Section 7.1 to the process model introduced in Section 7.3. We take $\mathit{Moves} = \mathcal{M}$, as the set of process moves, corresponding to the set of all messages in $\mathcal{M}$. For $1 \leq i \leq n$, we map each participant A_{i-1} to a process P_i as follows:

- $X_i = V_{i-1} \cup \{L_i\}$, is the set of variables of process P_i that includes all participant variables V_{i-1} and a special variable L_i corresponding to line numbers, taking finitely many values in $\mathbb{N}$,
- for all valuations $f \in \mathcal{F}_i[X_i]$, we have $\Gamma_i(f) = \Lambda_{i-1}(f \downarrow V_{i-1})$ and
- $\delta_i : \mathcal{F}_i[\{L_i\}] \times \mathcal{F}_i[X_i \setminus \{L_i\}] \times \mathit{Moves} \mapsto \mathcal{F}_i[\{L_i\}] \times \mathcal{F}_i[X_i \setminus \{L_i\}]$ is the process transition function that exactly corresponds to the participant transition function Λ_{i-1}.

The sets X_i form a partition of $X = \bigcup_{i=1}^{n} X_i$. The set of processes P_i, given all possible behaviors of a fair scheduler Sc, corresponds to the most general exchange program. The realization of a protocol corresponds to a refinement $P'_i \preceq P_i$ for $1 \leq i \leq n$, where each participant A'_{i-1} maps to the process P'_i as follows:

- $X'_i = X_i = V_{i-1} \cup \{L_i\}$ is the set of variables of process P'_i,
- for all valuations $f \in \mathcal{F}'_i[X'_i]$, we have $\Gamma'_i(f) = \Lambda'_{i-1}(f \downarrow V_{i-1})$ and
- for all valuations $f \in \mathcal{F}'_i[X'_i]$, for all moves $m \in \mathit{Moves}$, we have $\delta'_i(f, m) = \Lambda'_{i-1}(f(L_i), f \downarrow V_{i-1}, m)$.

A protocol instance (protocol run) is a trace in $[\![P'_1 \parallel P'_2 \ldots \parallel P'_n \parallel \mathrm{Sc}]\!](v_0)$ for an initial valuation $v_0 \in \mathcal{F}[X]$. The specifications of the participants, which were defined as a set of desired sequences of messages, are subsets of traces in $[\![P'_1 \parallel P'_2 \ldots \parallel P'_n \parallel \mathrm{Sc}]\!](v_0)$. Given specifications φ_i for process P_i, a Y-attack for $Y \subseteq \{P_1, P_2, \ldots, P_n\}$ satisfies φ_i for all $P_i \in Y$, while violating φ_j for at least one process $P'_j \in (\{P_1, P_2, \ldots, P_n\} \setminus Y)'$. There are three participants in a two party fair non-repudiation protocol, the originator O, the recipient R and the trusted third party TTP. We therefore take $n = 3$ in modeling two party fair exchange protocols in the above translation.

We now prove Lemma 12. Given a refinement $P' = (\mathrm{O}', \mathrm{R}', \mathrm{TTP}') \preceq P$, we first characterize the smallest restriction on O' and R' that satisfy the implication conditions:

$$[\![\mathrm{O} \parallel \mathrm{R}' \parallel \mathrm{TTP} \parallel \mathrm{Sc}]\!] \subseteq (\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{R}}; \text{ and} \tag{A.1}$$

$$[\![\mathrm{O}' \parallel \mathrm{R} \parallel \mathrm{TTP} \parallel \mathrm{Sc}]\!] \subseteq (\varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{O}} \,. \tag{A.2}$$

We show that for all refinements $\mathrm{R}' \preceq \mathrm{R}$, the implication condition (A.1) holds. In order to characterize the smallest restrictions on O that satisfies the implication condition (A.2), we recall the following constraints on O. We show that these constraints are both necessary and sufficient to satisfy (A.2).

AGS constraints on O. We say that a refinement $\mathrm{O}' \preceq \mathrm{O}$ satisfies the *AGS constraints on O* if O' satisfies the following constraints:

1. $a_1^{\mathrm{O}} \notin \Gamma_{\mathrm{O}'}(v_0)$;
2. $\mathrm{EOO}_k^{\mathrm{O}} \notin \Gamma_{\mathrm{O}'}(\{M_1, \mathrm{EOR}, \mathrm{ABR}^{\mathrm{O}}\})$; and
3. $a_1^{\mathrm{O}} \notin \Gamma_{\mathrm{O}'}(\{M_1, \mathrm{EOR}, M_3\})$.

The most flexible refinements $O' \preceq O$ and $R' \preceq R$. We now characterize the most flexible refinements $O' \preceq O$ and $R' \preceq R$ that satisfy the implication conditions $(\varphi_R \wedge \varphi_{TTP}) \Rightarrow \varphi_O$ and $(\varphi_O \wedge \varphi_{TTP}) \Rightarrow \varphi_R$.

Lemma 20 *For all refinements $R' \preceq R$, the following assertion holds:*

$$[\![O \parallel R' \parallel TTP \parallel Sc]\!] \subseteq (\varphi_O \wedge \varphi_{TTP}) \Rightarrow \varphi_R.$$

Proof. Consider an arbitrary refinement $R' \preceq R$. We have the following cases of sets of traces of $[\![O \parallel R' \parallel TTP \parallel Sc]\!]$ for the proof:

- *Case 1. Set of traces where m_3 has been received.* For all traces where m_3 has been received, φ_R is satisfied. Therefore all these traces satisfy the implication condition, $(\varphi_O \wedge \varphi_{TTP}) \Rightarrow \varphi_R$.
- *Case 2. Set of traces where m_3 has not been received.* For all traces where m_3 has not been received, the traces where either φ_O or φ_{TTP} is violated, satisfy the implication condition $(\varphi_O \wedge \varphi_{TTP}) \Rightarrow \varphi_R$ trivially. The interesting case are those traces that satisfy $\varphi_O \wedge \varphi_{TTP}$ but violate φ_R. These are exactly the traces where O does not have EOR_k^R, since m_4 is not sent before receiving m_3, and R does not have EOO_k^O, as otherwise φ_R would be satisfied. We have following cases that lead to a contradiction:
  - *Case (a). O aborts the protocol.* In these traces, since φ_{TTP} is satisfied, the abort token must have been sent to both agents, and since neither agent will be sent the other's signature and the channels between the agents and the TTP are resilient, the traces satisfy φ_R, leading to a contradiction.

- *Case (b). O or R' resolve the protocol.* In these traces, since φ_{TTP} is true, the TTP sends $\text{EOO}_k^{\text{TTP}}$ to R and $\text{EOR}_k^{\text{TTP}}$ to O and never sends either AO or AR. This implies, given the channel between the agents and the TTP is resilient, the traces satisfy φ_{R}, leading to a contradiction.
- *Case (c). R' chooses move ι.* In these traces, since φ_{O} is true, either O aborts the protocol after sending m_1 or she chooses to abort or resolve the protocol after receiving m_2. In either case, given the traces satisfy φ_{TTP}, by the above argument φ_{R} is satisfied as well, irrespective of the behavior of the channel between O and R. This again leads to a contradiction.

Since we have shown that for all traces, either φ_{R} is satisfied or satisfaction of $\varphi_{\text{O}} \wedge \varphi_{\text{TTP}}$ implies satisfaction of φ_{R}, we conclude that for all refinements $R' \preceq R$ the assertion holds.

■

It follows from Lemma 20, that as R' can always resolve the protocol in state $\{\text{EOO}\}$ and all successor states, such that the resulting trace satisfies $(\varphi_{\text{O}} \wedge \varphi_{\text{TTP}}) \Rightarrow \varphi_{\text{R}}$, we have $m_2 \in \Gamma_{\text{R}'}(\{\text{EOO}\})$. Similarly, $m_4 \in \Gamma_{\text{R}'}(\{\text{EOO}, M_2, \text{EOO}_k^{\text{O}}\})$ as φ_{R} is satisfied in all traces where m_3 has been received, thus satisfying $(\varphi_{\text{O}} \wedge \varphi_{\text{TTP}}) \Rightarrow \varphi_{\text{R}}$.

In the following lemma, in assertion 1 we show that for all refinements $\text{O}' \preceq \text{O}$ that satisfy the AGS constraints on O, the implication condition (A.2) is satisfied; in assertion 2 we show that if O' does not satisfy the AGS constraints on O, the implication condition (A.2) is violated.

Lemma 21 (The smallest restriction on $\text{O}' \preceq \text{O}$) *For all refinements $O' \preceq O$, the following assertions hold:*

1. *if O' satisfies the AGS constraints on O, then*

$$[\![O' \parallel R \parallel TTP \parallel Sc]\!] \subseteq (\varphi_R \wedge \varphi_{TTP}) \Rightarrow \varphi_O.$$

2. *if O' does not satisfy the AGS constraints on O, then*

$$[\![O' \parallel R \parallel TTP \parallel Sc]\!] \not\subseteq (\varphi_R \wedge \varphi_{TTP}) \Rightarrow \varphi_O.$$

Proof. Consider an arbitrary refinement $\mathrm{O}' \preceq \mathrm{O}$ that satisfies the AGS constraints on O. We have the following cases of sets of traces of $[\![\mathrm{O}' \parallel \mathrm{R} \parallel \mathrm{TTP} \parallel \mathrm{Sc}]\!]$ for the proof:

– *Case 1. Set of traces where m_4 has been received.* In the case of classical co-synthesis, an adversarial R will never send m_4 as that satisfies φ_{O} unconditionally, but in assume-guarantee synthesis, from Lemma 20, since all refinements of R satisfy the weaker condition of $(\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{R}}$, $m_4 \in \Gamma_{\mathrm{R}'}(\langle \mathrm{EOO}, M_2, \mathrm{EOO}_k^{\mathrm{O}} \rangle)$. For all traces where m_4 has been received, φ_{O} is satisfied. Therefore all these traces satisfy the implication condition $(\varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{O}}$.

– *Case 2. Set of traces where m_4 has not been received.* For all traces where m_4 has not been received, the traces where either φ_{R} or φ_{TTP} is violated, satisfy the implication condition $(\varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{O}}$ trivially. The interesting case are those traces that satisfy $\varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}}$ but violate φ_{O}. These are exactly the traces where O does not have $\mathrm{EOR}_k^{\mathrm{R}}$, since m_4 has not been received. We have the following cases that lead to a contradiction:

  ◦ *Case (a). O' has sent m_3.* In these traces, since O' satisfies the AGS constraints on O, the only choice of moves for O' are ι or r_1^{O}; she can wait for R to send m_4

or resolve the protocol. In the set of traces where she eventually receives m_4, by Case 1, the traces satisfy $(\varphi_R \wedge \varphi_{TTP}) \Rightarrow \varphi_O$. If she does not receive m_4, she will eventually resolve the protocol to satisfy φ_O. In the set of traces where she eventually resolves the protocol, since φ_{TTP} is satisfied, and R cannot abort the protocol, the TTP will eventually respond to her request by sending her non-repudiation evidence and not the abort token. These traces therefore satisfy φ_O, leading to a contradiction.

- *Case (b). O' aborts the protocol before sending m_3.* Since O' satisfies the AGS constraints on O, she cannot abort the protocol in the initial state v_0. Therefore, O' must have started the protocol by sending m_1. In all these traces, O' aborts the protocol after sending m_1 but before sending m_3 and since O' satisfies the AGS constraints on O, she will not send m_3 after sending the abort request. Since these traces satisfy φ_{TTP}, the abort token must have been sent to both agents, and since neither agent will be sent the other's signature and the channels between the agents and the TTP are resilient, the traces satisfy φ_O, leading to a contradiction.

- *Case (c). O' resolves the protocol before sending m_3.* In these traces, since φ_{TTP} is true, the TTP sends EOR_k^{TTP} to O and EOO_k^{TTP} to R and never sends either AO or AR. This implies, given the channel between the agents and the TTP is resilient, the traces satisfy φ_O, leading to a contradiction.

- *Case (d). O' chooses move ι instead of sending m_3.* In these traces, since φ_R is true, R must have resolved the protocol after receiving m_1. In this case, given the traces

satisfy φ_{TTP}, by the above argument φ_{O} is satisfied as well. This again leads to a contradiction.

- *Case (e). The channel between O and R is unreliable.* If either m_1 or m_2 are not delivered, then O$'$ can abort the protocol. If either m_3 or m_4 are not delivered, then O$'$ can resolve the protocol. In either case, by Case (a), Case (b) and Case (c), we have φ_{O} is satisfied even when the channel between O and R is unreliable, leading to a contradiction.

We conclude that for all O$'$ that satisfy the AGS constraints on O, we have $[\![\text{O}' \parallel \text{R} \parallel \text{TTP} \parallel \text{Sc}]\!] \subseteq (\varphi_{\text{R}} \wedge \varphi_{\text{TTP}}) \Rightarrow \varphi_{\text{O}}$.

For assertion 2, consider an arbitrary refinement O$'$ $\preceq$ O that does not satisfy the AGS constraints on O. We consider violation of the constraints on a case by case basis. For each case we produce a witness trace that violates the implication condition $(\varphi_{\text{R}} \wedge \varphi_{\text{TTP}}) \Rightarrow \varphi_{\text{O}}$. We proceed as follows:

- *Case 1.* $a_1^O \in \Gamma_{O'}(v_0)$. In a trace where O$'$ sends an abort request before sending message m_1 in the initial protocol state v_0, it is trivially the case that the trace does not satisfy φ_{O} but satisfies φ_{R}. If the TTP satisfies the AGS constraints on the TTP and sends $[a_2^{\text{O}}, a_2^{\text{R}}]$ in response, then the trace satisfies φ_{TTP}. Therefore, the trace violates $(\varphi_{\text{R}} \wedge \varphi_{\text{TTP}}) \Rightarrow \varphi_{\text{O}}$.

- *Case 2.* $EOO_k^O \in \Gamma_{O'}(M_1, EOO, ABR^O)$. To produce a witness trace we consider a partial trace that ends in protocol state $\{M_1, \text{EOO}, \text{ABR}^{\text{O}}\}$; messages m_1 and m_2 have been received and a_1^{O} has been sent. Since the channel between O and the TTP is resilient, the abort request is eventually processed by the TTP. If O$'$ sends message

m_3 in this state and the TTP responds with move $[a_2^{\mathrm{O}}, a_2^{\mathrm{R}}]$ to her abort request, then there exists a behavior of the channel between O and R such that m_3 is eventually delivered and the protocol is aborted. The trace therefore satisfies $\varphi_R \wedge \varphi_{\mathrm{TTP}}$ but violates φ_{O}; as O cannot get R's signature after the protocol is aborted and R has her signature.

– *Case 3.* $a_1^O \in \Gamma_{O'}(M_1, EOO, M_3)$. To produce a witness trace we consider a partial trace that ends in protocol state $\{M_1, \mathrm{EOO}, M_3\}$; messages m_1 and m_2 have been received and m_3 has been sent. If O' aborts the protocol in this state and the TTP satisfies the AGS constraints on the TTP and responds with move $[a_2^{\mathrm{O}}, a_2^{\mathrm{R}}]$, then there exists a behavior of the channel between O and R, where m_3 is eventually delivered to R. The trace satisfies $\varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}}$ but violates φ_{O}.

We conclude that if O' does not satisfy the AGS constraints on O, then $[\![\mathrm{O}' \parallel \mathrm{R} \parallel \mathrm{TTP} \parallel \mathrm{Sc}]\!] \not\subseteq (\varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{O}}$. ■

From Lemma 21, it is both necessary and sufficient that O satisfies the AGS constraints on O to ensure the implication condition (A.2).

The maximal refinement $\mathbf{P}^* = (\mathbf{O}^*, \mathbf{R}^*, \mathbf{TTP}^*)$. We recall the definition of the maximal refinement $\mathrm{P}^* = (\mathrm{O}^*, \mathrm{R}^*, \mathrm{TTP}^*)$ below:

1. $\mathrm{O}^* \preceq \mathrm{O}$ satisfies the AGS constraints on O and for all O' that satisfy the constraints, we have $O' \preceq \mathrm{O}^*$;

2. $\mathrm{R}^* = \mathrm{R}$; and

3. $\mathrm{TTP}^* \preceq \mathrm{TTP}$ satisfies the AGS constraints on the TTP and for all TTP' that satisfy the

constraints, we have $\text{TTP}' \preceq \text{TTP}^*$.

The weak co-synthesis requirement. Let $b \in \mathbb{N}$ be a bound on the number of times that O or the TTP may choose the idle move ι when scheduled by Sc. In the following lemma, for all refinements $P' \preceq \text{P}^*$ that satisfy the AGS constraints on the TTP, in assertion 1 we show that if b is finite, then the condition for weak co-synthesis is satisfied; in assertion 2 we show that if b is unbounded, then the condition for weak co-synthesis is violated.

Lemma 22 (Bounded idle time lemma) *For all refinements $P' = (O', R', TTP') \preceq P^*$ that satisfy the AGS constraints on the TTP, for all $b \in \mathbb{N}$ with O' and TTP' choosing at most b idle moves when scheduled by Sc, the following assertions hold:*

1. *if b is finite, then $[\![O' \parallel R' \parallel TTP' \parallel Sc]\!] \subseteq (\varphi_O \wedge \varphi_R \wedge \varphi_{TTP})$.*
2. *if b is unbounded, then $[\![O' \parallel R' \parallel TTP' \parallel Sc]\!] \not\subseteq (\varphi_O \wedge \varphi_R \wedge \varphi_{TTP})$.*

Proof. For the first assertion, we show that the condition for weak co-synthesis holds against all possible behaviors of the channel between O and R. We have the following cases:

– *Case 1. Agents abort or resolve the protocol.* In all traces where the agents abort or resolve the protocol, given b is finite and that TTP' satisfies the AGS constraints on the TTP, by Lemma 11 (assertion 1), TTP' will eventually respond to the first and all subsequent requests such that φ_{TTP} is satisfied. In all these traces, given the channels between the agents and the TTP are resilient, both agents get either the abort token or non-repudiation evidences but never both. This ensures φ_{O} and φ_{R} are satisfied.

– *Case 2. The channel between O and R is resilient.* In all traces where neither agent aborts nor resolves the protocol, φ_{TTP} is satisfied trivially. Further, the only refinements of the agents that neither abort nor resolve the protocol are those where $\{m_1, m_3\} \in \mathit{Moves}_{\mathrm{O}'}$ and $\{m_2, m_4\} \in \mathit{Moves}_{\mathrm{R}'}$. Since b is finite, the only choice of moves for O', since she does not abort or resolve the protocol, are m_1 in state v_0 and m_3 in state $\{M_1, \mathrm{EOR}\}$, after choosing at most b idle moves at each state. Similarly, the only choice of moves for R' are ι or m_2 in state $\{\mathrm{EOO}\}$ and ι or m_4 in state $\{\mathrm{EOO}, M_2, \mathrm{EOO}_k^{\mathrm{O}}\}$. If R' never sends m_2, then O' will eventually abort the protocol after bounded idle time and this case reduces to Case 1. If R' never sends m_4, then O' will eventually resolve the protocol after bounded idle time and this case reduces to Case 1. If R' sends m_2 and m_4 eventually, since the channel between O and R is assumed resilient, messages m_1, m_2, m_3 and m_4 are eventually delivered satisfying φ_{O} and φ_{R}.

– *Case 3. The channel between O and R is unreliable.* Since $\mathrm{O}' \preceq \mathrm{O}^*$, we have $a_1^{\mathrm{O}} \notin \Gamma_{\mathrm{O}'}(v_0)$ and $a_1^{\mathrm{O}} \notin \Gamma_{\mathrm{O}'}(\{M_1, \mathrm{EOR}, M_3\})$; O' can abort the protocol in all other states. Therefore, O' satisfies the AGS constraints on O. Since b is finite and O' cannot resolve the protocol before initiating it, the only choice of moves for O' in state v_0 is to send m_1 eventually. If the channel between O and R does not deliver either messages m_1 or m_2, the only choice of moves for O' is to abort the protocol. If either messages m_3 or m_4 are not delivered, then the only choice of moves for O' is to resolve the protocol. In both these cases, since O' chooses to abort or resolve the protocol, by Case 1 the result follows.

We conclude that irrespective of the behavior of the channel between O and R, if b is finite, we have $[\![\mathrm{O}' \parallel \mathrm{R}' \parallel \mathrm{TTP}' \parallel \mathrm{Sc}]\!] \subseteq (\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}})$.

For the second assertion, given an unbounded b, to show that weak co-synthesis fails, it suffices to show that there exists a behavior of the agents, the TTP and the channels that violates the condition for weak co-synthesis. Consider a partial trace ending in protocol state $\{M_1, \mathrm{EOO}, M_2, \mathrm{EOR}, M_3, \mathrm{EOO}_k^{\mathrm{O}}, \mathrm{RES}^{\mathrm{O}}\}$; messages m_1, m_2 and m_3 have been received, R' chooses to go idle, never sending m_4 and O' has sent r_1^{O}. Since b is unbounded, if TTP' chooses to remain idle forever, then φ_{O} and φ_{TTP} are violated leading to a violation of $(\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}})$. Therefore, given an unbounded b, we have $[\![\mathrm{O}^* \parallel \mathrm{R}^* \parallel \mathrm{TTP}^* \parallel Sc]\!] \not\subseteq (\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}})$. ■

From Lemma 22, it is both necessary and sufficient that the refinements $P' \preceq \mathrm{P}^*$ that satisfy the AGS constraints on the TTP, also satisfy bounded idle time to ensure weak co-synthesis. While O and the TTP should satisfy bounded idle time, there are no restrictions on R. Using Lemma 11, Lemma 20, Lemma 21 and Lemma 22 we now present a proof of Lemma 12.

Proof. **(Proof of Lemma 12).** In one direction, consider an arbitrary refinement $P' = (\mathrm{O}', \mathrm{R}', \mathrm{TTP}') \in \overline{\mathrm{P}}$. We show that the conditions of assume-guarantee synthesis are satisfied as follows:

- *The implication condition for O.* Since $P' \preceq \mathrm{P}^*$, we have $\mathrm{O}' \preceq \mathrm{O}^*$, $\mathrm{R}' \preceq \mathrm{R}^*$ and $\mathrm{TTP}' \preceq \mathrm{TTP}^*$. As $a_1^{\mathrm{O}} \notin \Gamma_{\mathrm{O}^*}(v_0)$ and $a_1^{\mathrm{O}} \notin \Gamma_{\mathrm{O}^*}(M_1, \mathrm{EOR}, M_3)$, the refinement P' satisfies the AGS constraints on O. Therefore, by Lemma 21 (assertion 1), we have $[\![\mathrm{O}' \parallel \mathrm{R} \parallel \mathrm{TTP} \parallel \mathrm{Sc}]\!] \subseteq (\varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{O}}$.

– *The implication condition for R.* By Lemma 20, we have $[\![\mathrm{O} \parallel \mathrm{R}' \parallel \mathrm{TTP} \parallel \mathrm{Sc}]\!] \subseteq (\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{R}}$.

– *The implication condition for the TTP.* Since $\mathrm{TTP}' \preceq \mathrm{TTP}^*$ and TTP' satisfies the AGS constraints on the TTP, by Lemma 11 (assertion 1), φ_{TTP} is satisfied irrespective of the behavior of O and R, which implies $[\![\mathrm{O} \parallel \mathrm{R} \parallel \mathrm{TTP}' \parallel \mathrm{Sc}]\!] \subseteq (\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}}) \Rightarrow \varphi_{\mathrm{TTP}}$.

– *The weak co-synthesis condition.* Given P' satisfies bounded idle time, by Lemma 22 we have $[\![\mathrm{O}' \parallel \mathrm{R}' \parallel \mathrm{TTP}' \parallel \mathrm{Sc}]\!] \subseteq (\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}})$; weak co-synthesis holds.

Since we have shown that the refinement P' satisfies all the implication conditions and the weak co-synthesis condition of assume-guarantee synthesis, we have $P' \in P_{AGS}$. Hence $\overline{\mathrm{P}} \subseteq P_{AGS}$.

In the other direction, consider an arbitrary refinement $P'' = (\mathrm{O}'', \mathrm{R}'', \mathrm{TTP}'') \in P_{AGS}$. We show that $P'' \in \overline{\mathrm{P}}$ as follows:

– *The AGS constraints on O.* By Lemma 21, since it is both necessary and sufficient that a refinement satisfy the AGS constraints on O to ensure the implication condition $(\varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{O}}$ is satisfied, given the implication condition holds, we conclude that P'' satisfies the AGS constraints on O. Therefore, $\mathrm{O}'' \preceq \mathrm{O}^*$.

– *The AGS constraints on the TTP.* By Lemma 11, since it is both necessary and sufficient that a refinement satisfy the AGS constraints on the TTP to ensure the implication condition $(\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}}) \Rightarrow \varphi_{\mathrm{TTP}}$ is satisfied, given the implication condition holds, we conclude that P'' satisfies the AGS constraints on the TTP and $\mathrm{TTP}'' \preceq \mathrm{TTP}^*$.

– *The bounded idle time condition.* By Lemma 22, since it is both necessary and sufficient that a refinement satisfy bounded idle time to ensure weak co-synthesis, since weak co-synthesis holds in this case, we conclude that P'' satisfies bounded idle time.

– $P'' \preceq P^*$. Since we have shown that $\mathrm{O}'' \preceq \mathrm{O}^*$ and $\mathrm{TTP}'' \preceq \mathrm{TTP}^*$, we have $P'' \preceq \mathrm{P}^*$.

– $P_* \preceq P''$. Since P_* is the smallest refinement in the set P_{AGS}, given $P'' \in P_{AGS}$, it must be the case that $\mathrm{P}_* \preceq P''$.

For $P'' \in P_{AGS}$, as we have shown that $\mathrm{P}_* \preceq P'' \preceq \mathrm{P}^*$, P'' satisfies the AGS constraints on the TTP and satisfies bounded idle time. Thus we have $P'' \in \overline{\mathrm{P}}$ and hence $P_{AGS} \subseteq \overline{\mathrm{P}}$. The result follows. ■

We now present a proof of Lemma 13. We recall the *enhanced AGS constraints on the TTP* below:

1. *Abort constraint.* If the first request received by the TTP is a_1^{O}, then her response to that request should be $[a_2^{\mathrm{O}}, a_2^{\mathrm{R}}]$; If the first request received by the TTP is a_1^{R}, then her response to that request should be req^{O};

2. *Resolve constraint.* If the first request received by the TTP is a resolve request, then her response to that request should be $[r_2^{\mathrm{O}}, r_2^{\mathrm{R}}]$; If the TTP receives res^{O} in response to req^{O} within bounded idle time, then her response should be $[r_2^{\mathrm{O}}, r_2^{\mathrm{R}}]$, otherwise it should be $[a_2^{\mathrm{O}}, a_2^{\mathrm{R}}]$.

3. *Accountability constraint.* If the first response from the TTP is $[x, y]$ or the first response from the TTP is req^{O} and the next response is $[x, y]$, then for all subsequent abort or resolve requests her response should be in the set $\{\iota, x, y, [x, y]\}$.

Proof. **(Proof of Lemma 13).** From Protocol 2, since the refinement O_s does not abort the protocol either in the initial state v_0 or after sending message m_3, we have O_s satisfies the AGS constraints on O. By our definition of the behavior of TTP_s, we have TTP_s satisfies the enhanced AGS constraints on the TTP. From the definition of the main protocol in Protocol 2 and the abort subprotocol in Protocol 3, since the resolve subprotocol is identical to the KM protocol, we have O_s and TTP_s satisfy the bounded idle time requirement. We take $A = \{O, R, TTP\}$ and show that there is no Y-attack for $Y \subseteq \{O, R\}$ through the following cases:

- *Case 1.* $|Y| = 2$. In this case $Y = \{O, R\}$. We show that $[\![O \parallel R \parallel TTP_s \parallel Sc]\!] \subseteq \varphi_{TTP}$. For all traces in $[\![O \parallel R \parallel TTP_s \parallel Sc]\!]$ where R does not abort the protocol, since TTP_s satisfies the enhanced AGS constraints on the TTP, by Lemma 11 (assertion 1), φ_{TTP} is satisfied. For all traces where R sends an abort request, the TTP sends req^O. If O responds with res^O within bounded idle time, then the TTP resolves the protocol for both O and R such that the AGS constraints on the TTP are satisfied. If O does not send res^O within bounded idle time, then the TTP aborts the protocol, such that the AGS constraints on the TTP are satisfied. For all subsequent abort requests from R, the TTP response satisfies the AGS constraints on the TTP. All traces therefore satisfy φ_{TTP}. Hence, there is no Y-attack in this case.

- *Case 2.* $|Y| = 1$. In this case, either $Y = \{O\}$ or $Y = \{R\}$. We have the following cases towards the proof:

  - *Case (a).* $Y = \{O\}$. We show that $[\![O \parallel R_s \parallel TTP \parallel Sc]\!] \subseteq (\varphi_O \wedge \varphi_{TTP}) \Rightarrow \varphi_R$; it will follow that $[\![O \parallel R_s \parallel TTP_s \parallel Sc]\!] \subseteq (\varphi_O \wedge \varphi_{TTP}) \Rightarrow \varphi_R$. Consider the

set of traces in $[\![\mathrm{O} \parallel \mathrm{R}_s \parallel \mathrm{TTP} \parallel \mathrm{Sc}]\!]$. For all traces where R does not abort the protocol, by Lemma 20, we have $\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{TTP}} \Rightarrow \varphi_{\mathrm{R}}$. For all traces where R aborts the protocol, if he has received m_3, then φ_{R} is satisfied. For all traces where R aborts the protocol and message m_3 has not been received, if φ_{TTP} is violated, then the implication holds and if φ_{TTP} is satisfied, then either both agents get abort tokens or their respective non-repudiation evidences, thus satisfying φ_{R}. We have shown that all traces satisfy the implication condition $\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{TTP}} \Rightarrow \varphi_{\mathrm{R}}$. Since we have a fixed TTP that satisfies the AGS constraints on the TTP, we have φ_{TTP} is satisfied in all traces by Case 1. As φ_{O} is satisfied by assumption, we conclude φ_{R} is satisfied as well. Therefore, there is no Y-attack in this case.

- *Case (b).* $Y = \{R\}$. It can be shown that $[\![\mathrm{O}_s \parallel \mathrm{R} \parallel \mathrm{TTP} \parallel \mathrm{Sc}]\!] \not\subseteq (\varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{O}}$. We show that, by fixing the TTP, we have $[\![\mathrm{O}_s \parallel \mathrm{R} \parallel \mathrm{TTP}_s \parallel \mathrm{Sc}]\!] \subseteq (\varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{O}}$. Consider the set of traces $[\![\mathrm{O}_s \parallel \mathrm{R} \parallel \mathrm{TTP}_s \parallel \mathrm{Sc}]\!]$. For all traces where R does not abort the protocol, since O satisfies the AGS constraints on O, by Lemma 21, we have $(\varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{O}}$. If R aborts the protocol, since the TTP satisfies the enhanced AGS constraints on the TTP, and the channel between O and the TTP is operational, req^{O} must have been received by O. At this stage, if O_s has sent message m_3, then the only choice of moves for O_s to satisfy φ_{O} is res^{O}; a request to resolve the protocol. Since the channels are operational, there exists a bound on the idle time of the TTP such that both req^{O} and res^{O} can be delivered within this bound. Moreover, as TTP_s satisfies the enhanced AGS constraints on the TTP, both O and R will be issued non-

repudiation evidences and never abort tokens, thus satisfying φ_{O}. If O_s has not sent message m_3, then the only choice of moves for O_s to satisfy φ_{O} are ι or res^{O}. In all these traces, since TTP_s satisfies bounded idle time and the AGS constraints on the TTP, either both agents get non-repudiation evidences or abort tokens but never both, thus satisfying φ_{O}. Therefore, all these traces satisfy $(\varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}}) \Rightarrow \varphi_{\mathrm{O}}$, which given φ_{R} is satisfied by assumption and φ_{TTP} is satisfied by Case 1, implies φ_{O} is satisfied as well. There is no Y-attack in this case.

– *Case 2.* $|Y| = 0$. In this case $Y = \emptyset$ and $(A \setminus Y)' = \{\mathrm{O}_s, \mathrm{R}_s, \mathrm{TTP}_s\}$. Since P_s satisfies bounded idle time, in all traces where R does not abort the protocol, by Lemma 22, the condition for weak co-synthesis is satisfied. In all traces where R aborts the protocol, as TTP_s satisfies the AGS constraints on the TTP, she sends req^{O}. In all these traces, since TTP_s and O_s satisfy bounded idle time, and the channels are operational, O_s chooses ι or sends res^{O} and TTP_s responds with either abort tokens or non-repudiation evidences but not both, leading to the satisfaction of φ_{O} and φ_{R}. Since φ_{TTP} is satisfied by Case 1, all these traces satisfy $(\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}})$. Therefore, there is no Y-attack in this case.

The result follows. ■

Proof. **(Proof of Theorem 31).** By Lemma 13, it follows that if the TTP does not change her behavior, then P_s is attack-free. Further, by the weak co-synthesis condition, we have $[\![\mathrm{O}_s \parallel \mathrm{R}_s \parallel \mathrm{TTP}_s \parallel \mathrm{Sc}]\!] \subseteq (\varphi_{\mathrm{O}} \wedge \varphi_{\mathrm{R}} \wedge \varphi_{\mathrm{TTP}})$ and hence by Theorem 25, we have $[\![\mathrm{O}_s \parallel \mathrm{R}_s \parallel \mathrm{TTP}_s \parallel \mathrm{Sc}]\!] \subseteq \varphi_f \cap \varphi_b$. Thus P_s satisfies fairness and balance and hence is fair and abuse-

free. Since $[\![O_s \parallel R_s \parallel TTP_s \parallel Sc]\!] \cap (\Diamond NRO \wedge \Diamond NRR) \neq \emptyset$, the refinement P_s enables an exchange of signatures and hence is an exchange protocol. We conclude that if the TTP does not change her behavior, then P_s is an attack-free fair non-repudiation protocol. ■

Bibliography

[ACC07] Alessandro Armando, Roberto Carbone, and Luca Compagna. Ltl model checking for security protocols. In *CSF*, pages 385–396, 2007.

[AETCR08] Almudena Alcaide, Juan M. Estévez-Tapiador, Julio César Hernández Castro, and Arturo Ribagorda. Nature-inspired synthesis of rational protocols. In *PPSN*, pages 981–990, 2008.

[AHK97] R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-time temporal logic. In *Proc. 38th IEEE Symp. Found. of Comp. Sci.*, pages 100–109. IEEE Computer Society Press, 1997.

[AHK02] R. Alur, T.A. Henzinger, and O. Kupferman. Alternating time temporal logic. *J. ACM*, 49:672–713, 2002.

[AHKV98] 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.

[AHM+98] Rajeev Alur, Thomas A. Henzinger, Freddy Y. C. Mang, Shaz Qadeer, Sri-

ram K. Rajamani, and Serdar Tasiran. Mocha: Modularity in model checking. In *CAV*, pages 521–525, 1998.

[AN01] André Arnold and Damian Niwiński. *Rudiments of μ-Calculus*. Elsevier, Amsterdam, The Netherlands, 2001.

[ASB+95] A. Aziz, V. Singhal, F. Balarin, R.K. Brayton, and A.L. Sangiovanni-Vincentelli. It usually works: The temporal logic of stochastic systems. In *Computer Aided Verification*, volume 939 of *Lect. Notes in Comp. Sci.* Springer-Verlag, 1995.

[ASW98] N. Asokan, Victor Shoup, and Michael Waidner. Asynchronous protocols for optimistic fair exchange. In *IEEE Symposium on Security and Privacy*, pages 86–99, 1998.

[BAN90] Michael Burrows, Martín Abadi, and Roger M. Needham. A logic of authentication. *ACM Trans. Comput. Syst.*, 8(1):18–36, 1990.

[Bas99] S. Basu. New results on quantifier elimination over real closed fields and applications to constraint databases. *J. ACM*, 46(4):537–555, 1999.

[Ber95] D.P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 1995. Volumes I and II.

[BF68] David. Blackwell and T. S. Ferguson. The big match. *The Annals of Mathematical Statistics*, 39(1):159–163, 1968.

[BK90] Zbignew 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.

[Bry86] Randal E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 35:677–691, 1986.

[But04] Giorgio C. Buttazzo. *Hard Real-time Computing Systems: Predictable Scheduling Algorithms And Applications (Real-Time Systems Series)*. Springer-Verlag TELOS, Santa Clara, CA, USA, 2004.

[Can88] J. F. Canny. Some algebraic and geometric computations in pspace. In *STOC*, pages 460–467. ACM Press, 1988.

[CdAH04] 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.

[CdAH05] K. Chatterjee, L. de Alfaro, and T.A. Henzinger. The complexity of stochastic rabin and streett games. In *Proc. 32nd Int. Colloq. Aut. Lang. Prog.*, volume 3580 of *Lect. Notes in Comp. Sci.*, pages 878–890. Springer-Verlag, 2005.

[CdAMR08] Krishnendu Chatterjee, Luca de Alfaro, Rupak Majumdar, and Vishwanath Raman. Algorithms for game metrics. In *FSTTCS*, pages 107–118, 2008.

[CE81] 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.

[CE82] Edmund M. Clarke and E. Allen Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Logic of Programs, Workshop*, pages 52–71, London, UK, 1982. Springer-Verlag.

[CES83] 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.

[CH07] Krishnendu Chatterjee and Thomas A. Henzinger. Assume-guarantee synthesis. In *TACAS*, pages 261–275, 2007.

[CHJ06] Krishnendu Chatterjee, Thomas A. Henzinger, and Marcin Jurdzinski. Games with secure equilibria. *Theor. Comput. Sci.*, 365(1-2):67–82, 2006.

[CK73] C. C. Chang and H. J. Keisler. *Model theory. Studies in Logic and the Foundations of Mathematics.* North-Holland, Amsterdam., 1973.

[CKS01] R. Chadha, M. Kanovich, and A. Scedrov. Inductive methods and contract-signing protocols. In *CCS '01: Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 176–185, New York, NY, USA, 2001. ACM.

[CKS06] Rohit Chadha, Steve Kremer, and Andre Scedrov. Formal analysis of multi-party contract signing. *J. Autom. Reasoning*, 36(1-2):39–83, 2006.

[CNP09] K. Chatzikokolakis, G. Norman, and D. Parker. Bisimulation for demonic schedulers. In L. de Alfaro, editor, *12th International Conference on Foundations of Software Science and Computation Structures (FOSSACS'09)*, volume 5504 of *LNCS*, pages 318–332. Springer, 2009.

[CR10] Krishnendu Chatterjee and Vishwanath Raman. Assume-guarantee synthesis for digital contract signing. *CoRR*, abs/1004.2697, 2010.

[dA97] L. de Alfaro. *Formal Verification of Probabilistic Systems*. PhD thesis, Stanford University, 1997. Technical Report STAN-CS-TR-98-1601.

[dAFMR05] Luca de Alfaro, Marco Faella, Rupak Majumdar, and Vishwanath Raman. Code aware resource management. In *EMSOFT*, pages 191–202, 2005.

[dAH01] 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.

[dAHK98] L. de Alfaro, T. A. Henzinger, and O. Kupferman. Concurrent reachability games. In *FOCS*, pages 564–575, 1998.

[dAHK07] L. de Alfaro, T.A. Henzinger, and O. Kupferman. Concurrent reachability games. *Theoretical Computer Science*, 386(3):188–217, 2007.

[dAHM03] 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.

[dAM04] L. de Alfaro and R. Majumdar. Quantitative solution of omega-regular games. *Journal of Computer and System Sciences*, 68:374–397, 2004.

[dAMRS07] L. de Alfaro, R. Majumdar, V. Raman, and M. Stoelinga. Game relations and metrics. In *LICS*, pages 99–108. IEEE Computer Society Press, 2007.

[dAMRS08] Luca de Alfaro, Rupak Majumdar, Vishwanath Raman, and Mariëlle Stoelinga. Game refinement relations and metrics. *Logical Methods in Computer Science*, 4(3), 2008.

[dAS03] 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.

[Der70] C. Derman. *Finite State Markovian Decision Processes*. Academic Press, 1970.

[Dev77] R. Devillers. Game interpretation of the deadlock avoidance problem. *Commun. ACM*, 20(10):741–745, 1977.

[DGJP99] J. Desharnais, V. Gupta, R. Jagadeesan, and P. Panangaden. Metrics for labelled Markov systems. In *CONCUR*, pages 258–273. Springer-Verlag, 1999.

[DGJP02] J. Desharnais, V. Gupta, R. Jagadeesan, and P. Panangaden. The metric analogue of weak bisimulation for probabilistic processes. In *LICS*, pages 413–422. ACM Press, 2002.

[DGJP03] Josée Desharnais, Vineet Gupta, Radha Jagadeesan, and Prakash Panan-

gaden. Approximating labelled markov processes. *Inf. Comput.*, 184(1):160–200, 2003.

[EA03] D.R. Engler and K. Ashcraft. RacerX: effective, static detection of race conditions and deadlocks. In *SOSP 03: Symposium on Operating Systems Principles*, pages 237–252. ACM, 2003.

[ECM95] 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.

[eco] ecos homepage. http://ecos.sourceware.org/.

[End01] Herbert Enderton. *A Mathematical Introduction to Logic*. Academic Press, 2001.

[EY80] S. Even and Y. Yacobi. Relations among public key signature systems, technical report 175. Technical report, Technion, Haifa, Israel, 1980.

[EY06] K. Etessami and M. Yannakakis. Recursive concurrent stochastic games. In *ICALP (2)*, pages 324–335. Springer-Verlag, 2006.

[EY07] K. Etessami and M. Yannakakis. On the complexity of Nash equilibria and other fixed points (extended abstract). In *FOCS*, pages 113–123. IEEE Computer Society Press, 2007.

[FV97] J. Filar and K. Vrieze. *Competitive Markov Decision Processes*. Springer-Verlag, 1997.

[GGJ76] M. R. Garey, R. L. Graham, and D. S. Johnson. Some NP-complete geometric problems. In *STOC*, pages 10–22. ACM Press, 1976.

[GJM99] Juan A. Garay, Markus Jakobsson, and Philip D. MacKenzie. Abuse-free optimistic contract signing. In *CRYPTO*, pages 449–466, 1999.

[HC92] 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.

[HJ94] H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Asp. Comput.*, 6(5):512–535, 1994.

[HM85] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32(1):137–161, January 1985.

[IMA02] 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.

[JS90] 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.

[KfR02] Steve Kremer and Jean francois Raskin. Game analysis of abuse-free contract signing. In *In Proc. 15th IEEE Computer Security Foundations Workshop*, pages 206–220. IEEE Computer Society, 2002.

[KKZJ07] Joost-Pieter Katoen, Tim Kemna, Ivan S. Zapreev, and David N. Jansen. Bisimulation minimisation mostly speeds up probabilistic model checking. In *TACAS*, pages 87–101, 2007.

[KMZ02] Steve Kremer, Olivier Markowitch, and Jianying Zhou. An intensive survey of fair non-repudiation protocols. *Computer Communications*, 25(17):1606–1621, 2002.

[KNY03] 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.

[Koz83a] D. Kozen. A probabilistic PDL. In *Proc. 15th ACM Symp. Theory of Comp.*, pages 291–297, 1983.

[Koz83b] D. Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27(3):333–354, 1983.

[KR03] Steve Kremer and Jean-François Raskin. A game-based verification of non-repudiation and fair exchange protocols. *Journal of Computer Security*, 11(3):399–430, 2003.

[KSK66] J.G. Kemeny, J.L. Snell, and A.W. Knapp. *Denumerable Markov Chains*. D. Van Nostrand Company, 1966.

[KY03] C. Kloukinas and S. Yovine. Synthesis of safe, qos extendible, application specific schedulers for heterogeneous real-time systems. In *ECRTS*, 2003.

[Lou00] Panagiotis Louridas. Some guidelines for non-repudiation protocols. *SIGCOMM Comput. Commun. Rev.*, 30(5):29–38, 2000.

[Low95] Gavin Lowe. An attack on the needham-schroeder public-key authentication protocol. *Inf. Process. Lett.*, 56(3):131–133, 1995.

[LS92] 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.

[Maj03] R. Majumdar. *Symbolic algorithms for verification and control*. PhD thesis, University of California, Berkeley, 2003.

[MGK02] Olivier Markowitch, Dieter Gollmann, and Steve Kremer. On fairness in exchange protocols. In *ICISC*, pages 451–464, 2002.

[Mil80] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lect. Notes in Comp. Sci.* Springer-Verlag, 1980.

[Mil90] R. Milner. Operational and algebraic semantics of concurrent processes. In *Handbook of Theoretical Computer Science*, volume B, pages 1202–1242. Elsevier Science Publishers, 1990.

[Min82] Toshimi Minoura. Deadlock avoidance revisited. *J. ACM*, 29(4):1023–1048, 1982.

[MK01] Olivier Markowitch and Steve Kremer. An optimistic non-repudiation protocol with transparent trusted third party. In *ISC*, pages 363–378, 2001.

[MM04] A. McIver and C. Morgan. *Abstraction, Refinement, and Proof for Probabilistic Systems*. Monographs in Computer Science. Springer-Verlag, 2004.

[MN81] J.F. Mertens and A. Neyman. Stochastic games. *International Journal of Game Theory*, 10:53–66, 1981.

[MP91] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer-Verlag, New York, 1991.

[MP95] Z. Manna and A. Pnueli. *Temporal Verification of Reactive Systems: Safety*. Springer-Verlag, New York, 1995.

[NMRW02] G.C. Necula, S. McPeak, S.P. Rahul, and W. Weimer. Intermediate language and tools for analysis and transformation of C programs. In *Proceedings of Conference on Compiler Construction (CC)*, 2002.

[PG99] Henning Pagnia and Felix C. Grtner. On the impossibility of fair exchange without a trusted third party. Technical report, Darmstadt University of Technology, 1999.

[Pnu77] A. Pnueli. The temporal logic of programs. In *Proc. 18th IEEE Symp. Found. of Comp. Sci.*, pages 46–57. IEEE Computer Society Press, 1977.

[PR89] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proceedings of the 16th Annual Symposium on Principles of Programming Languages*, pages 179–190. ACM Press, 1989.

[PS85] J.L. Peterson and A. Silberschatz. *Operating System Concepts*. Addison-Wesley, 1985.

[PT87] R. Paige and R. E. Tarjan. Three partition refinement algorithms. *SIAM Journal on Computing*, 16(6):973–989, 1987.

[QS81] 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.

[RW87] P. J. Ramadge and W. M. Wonham. Supervisory control of a class of discrete event processes. *SIAM J. Control Optim.*, 25(1):206–230, 1987.

[SBN+97] S. Savage, M. Burrows, C.G. Nelson, P. Sobalvarro, and T.A. An derson. Eraser: A dynamic data race detector for multithreaded programs. *ACM Transactions on Computer Systems*, 15(4):391–411, 1997.

[SE89] Robert S. Streett and E. Allen Emerson. An automata theoretic decision procedure for the propositional mu-calculus. *Inf. Comput.*, 81(3):249–264, 1989.

[Sha53] L.S. Shapley. Stochastic games. *Proc. Nat. Acad. Sci. USA*, 39:1095–1100, 1953.

[Sio58] M. Sion. On general minimax theorems. *Pacific Journal of Mathematics.*, 8:171–176, 1958.

[SL94] R. Segala and N.A. Lynch. Probabilistic simulations for probabilistic processes. In *CONCUR*, pages 481–496. Springer-Verlag, 1994.

[SL95] R. Segala and N.A. Lynch. Probabilistic simulations for probabilistic processes. *Nordic Journal of Computing*, 2(2):250–273, 1995.

[SM02] Vitaly Shmatikov and John C. Mitchell. Finite-state analysis of two contract signing protocols. *Theor. Comput. Sci.*, 283(2):419–450, 2002.

[SW08] Werner Sandmann and Verena Wolf. Computational probability for systems biology. In *FMSB*, pages 33–47, 2008.

[Tar51] A. Tarski. *A Decision Method for Elementary Algebra and Geometry*. University of California Press, Berkeley and Los Angeles, 1951.

[Tho90] W. Thomas. Automata on infinite objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, chapter 4, pages 135–191. Elsevier Science Publishers (North-Holland), Amsterdam, 1990.

[Tho95] 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.

[Tho97] Wolfgang Thomas. *Languages, automata, and logic*. Springer-Verlag New York, Inc., New York, NY, USA, 1997.

[TK09] D. Thorsley and E. Klavins. A theory of approximation for stochastic biochemical processes, 2009. to appear in B. Ingalls and P. Iglesias, eds., Control-Theoretic Approaches to Systems Biology. MIT Press, 2009.

[vBCZ+03] 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.

[vBSW08] Franck van Breugel, Babita Sharma, and James Worrell. Approximating a behavioural pseudometric without discount for probabilistic systems. *Logical Methods in Computer Science*, 4(2), 2008.

[vBW01a] F. van Breugel and J. Worrell. An algorithm for quantitative verification of probabilistic transition systems. In *CONCUR*, pages 336–350. Springer-Verlag, 2001.

[vBW01b] F. van Breugel and J. Worrell. Towards quantitative verification of probabilistic transition systems. In *ICALP*, pages 421–432. Springer-Verlag, 2001.

[vNM44] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. New York: John Wiley and Sons, 1944.

[Wey50] H. Weyl. Elementary proof of a minmax theorem due to von Neumann. In *Contributions to the Theory of Games, I*, volume 24 of *Annals of Mathematical Studies*, pages 19–25. Princeton University Press, 1950.

[Ye06] Y. Ye. Improved complexity results on solving real-number linear feasibility problems. *Math. Program.*, 106(2):339–363, 2006.

[ZDB00] Jianying Zhou, Robert H. Deng, and Feng Bao. Some remarks on a fair exchange protocol. In *Public Key Cryptography*, pages 46–57, 2000.

[ZG97] Jianying Zhou and Dieter Gollmann. An efficient non-repudiation protocol.

In *In PCSFW: Proceedings of the 10th Computer Security Foundations Workshop. IEEE Computer*, pages 126–132. IEEE Computer Society Press, 1997.

[ZG98] Jianying Zhou and Dieter Gollmann. Towards verification of non-repudiation protocols. In *In Proceedings of 1998 International Refinement Workshop and Formal Methods Pacific*, pages 370–380. Springer-Verlag, 1998.

[ZH07] L. Zhang and H. Hermanns. Deciding simulations on probabilistic automata. In *ATVA*, pages 207–222. Springer-Verlag, 2007.