Game-based verification and synthesis

Vester, Steen; Hansen, Michael Reichhardt

Publication date:
2016

Document Version
Publisher's PDF, also known as Version of record

Ph.D. Thesis
Doctor of Philosophy

DTU Compute
Department of Applied Mathematics and Computer Science

Game-based verification and synthesis

Steen Vester

Kongens Lyngby, Denmark 2016
PHD-2016-414
ISSN: 0909-3192

DTU Compute
Department of Applied Mathematics and Computer Science
Technical University of Denmark

Richard Petersens Plads
Building 324
2800 Kongens Lyngby, Denmark
Phone +45 4525 3031
compute@compute.dtu.dk
www.compute.dtu.dk

Summary

Infinite-duration games provide a convenient way to model distributed, reactive and open systems in which several entities and an uncontrollable environment interact. Here, each entitity as well as the uncontrollable environment are modelled as players.

A strategy for an entity player in the model corresponds directly to a program for the corresponding entity of the system. A strategy for a player which ensures that the player wins no matter how the other players behave then corresponds to a program ensuring that the specification of the entity is satisfied no matter how the other entities and the environment behaves. Synthesis of strategies in games can thus be used for automatic generation of correct-by-construction programs from specifications.

We consider verification and synthesis problems for several well-known game-based models. This includes both model-checking problems and satisfiability problems for logics capable of expressing strategic abilities of players in games with both qualitative and quantitative objectives.

A number of computational complexity results for model-checking and satisfiability problems in this domain are obtained. We also show how the technique of symmetry reduction can be extended to solve finitely-branching turn-based games more efficiently. Further, the novel concept of winning cores in parity games is introduced. We use this to develop a new polynomial-time under-approximation algorithm for solving parity games. Experimental results show that this algorithm performs better than the state-of-the-art algorithms in most benchmark games.

Two new game-based modelling formalisms for distributed systems are presented. The first makes it possible to reason about systems where several identical entities interact. The second provides a game-based modelling formalism for distributed systems with continuous time and probability distributions over the duration of delays. For these new models we provide decidability and undecidability results for problems concerning computation of symmetric Nash equilibria and for deciding existence of strategies that ensure reaching a target with a high probability.

Resumé

Spil af uendelig varighed udgør bekvemme modeller for distribuerede, reaktive og åbne systemer hvori flere enheder og et ukontrollerbart miljø interagerer. Hver enkelt enhed samt det ukontrollerbare miljø modelleres som spillere.

En strategi for en enhedsspiller i modellen svarer direkte til et program for den tilsvarende enhed i systemet. En strategi for en spiller, som sikrer at spilleren vinder ligegyldigt hvordan de andre spillere opfører sig, svarer så til et program der sikrer at specifikationen for enheden overholdes ligegyldigt hvordan de andre enheder og miljøet opfører sig. Syntese af strategier i spil kan således anvendes til automatisk generering af programmer fra specifikationer, der er korrekte per konstruktion.

Vi betragter verifikations- og synteseproblemer for flere velkendte spilbaserede modeller. Det inkluderer både model-checking problemer og satisfiability problemer for logikker som kan udtrykke strategiske evner for spillere i spil med både kvalitative og kvantitative mål.

En række beregningsmæssige kompleksitetsresultater for model-checking og satisfiability problemer indenfor dette felt bliver opnået. Vi viser også hvordan symmetrireduktion kan genereliseres til turbaserede spil med endelig forgrening. Desuden introduceres det nye koncept vindende kerner i parity games. Dette bruges til at udvikle en ny underapproksimerende algoritme som kører i polynomiel tid. Eksperimentelle resultater viser at denne algoritme præsterer bedre end de eksisterende algoritmer for de fleste benchmark spil.

To nye spilbaserede modelleringsformalismer for distribuerede systemer bliver præsenteret. Den første gør det muligt at ræsonnere om systemer hvor flere identiske enheder interagerer. Den anden er en formalisme for distribuerede systemer med kontinuert tid og sandsynlighedsfordelinger over varigheden af forsinkelser. For disse nye modeller præsenterers afgørbarheds- og uafgørbarhedsresultater for problemer vedrørende beregning af symmetriske Nash ligevægte og eksistens af strategier som sikrer at nå en bestemt tilstand med høj sandsynlighed.

Preface

This Ph.d. thesis was prepared at the department of Applied Mathematics and Computer Science at the Technical University of Denmark in fulfillment of the requirements for acquiring a Ph.d. degree. This work has been supervised by Michael Reichhardt Hansen, Technical University of Denmark and Valentin Goranko, Stockholm University.

The thesis focuses on foundational models and techniques for game-based verification and synthesis. It presents results from the following scientific publications:

- [BMV14] Patricia Bouyer, Nicolas Markey, and Steen Vester. "Nash Equilibria in Symmetric Games with Partial Observation". In: *Proceedings 2nd International Workshop on Strategic Reasoning (SR)*. Electronic Proceedings in Theoretical Computer Science. Grenoble, France, 2014, pages 49–55

- [BMV16] Patricia Bouyer, Nicolas Markey, and Steen Vester. "Nash Equilibria in Symmetric Graph Games with Partial Observation". To appear in a special issue of *Information & Computation*. 2016

- [GV14] Valentin Goranko and Steen Vester. "Optimal Decision Procedures for Satisfiability in Fragments of Alternating-time Temporal Logics". In: *Advances in Modal Logic 10, invited and contributed papers from the tenth conference on Advances in Modal Logic (AiML)*. 2014, pages 234–253

- [HKV16] Holger Hermanns, Jan Krcál, and Steen Vester. "Distributed Synthesis in Continuous Time". In: *Foundations of Software Science and Computation Structures (FOSSACS) - 19th International Conference*. 2016, pages 353–369

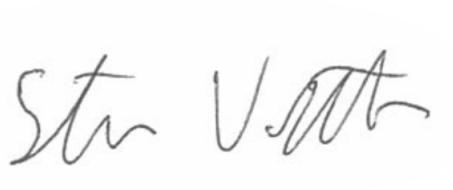
- [MV14] Nicolas Markey and Steen Vester. "Symmetry Reduction in Infinite Games with Finite Branching". In: *Proceedings of the 12th International Symposium on Automated Technology for Verification and Analysis (ATVA)*. volume 8837. Lecture Notes in Computer Science. Springer, November 2014, pages 281–296

- [Ves15] Steen Vester. "On the Complexity of Model-Checking Branching and Alternating-Time Temporal Logics in One-Counter Systems". In: *Proceedings of the 13th International Symposium on Automated Technology for Verification and Analysis (ATVA)*. 2015, pages 361–377

- [Ves16] Steen Vester. "Winning Cores in Parity Games". To appear in *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. 2016

All results presented in this thesis are from these papers. The presentation of the results have been adapted to this thesis, but no new results have been added. Before each chapter we write which papers the chapter is based on and summarize the changes that have been made when adapting them for the thesis.

Kongens Lyngby, Denmark, April 30, 2016



Steen Vester

Acknowledgements

I want to thank my two supervisors Michael Reichhardt Hansen and Valentin Goranko for supporting, teaching, helping, pushing, guiding, criticizing, believing and inspiring me. I have learned a lot from you both. Not just in a strictly academic way, but in all sorts of small ways. Also, both of you have taught me very different things about life and academia. The surprisingly small overlap between the things you have taught me has taught me another thing: The supervisor role matters a lot and is a very personal thing. I can also conclude that there is not one single good way to supervise. I have enjoyed all the discussions, the visiting and the tennis playing. You have both had a large impact on me.

I also want to thank my co-authors Patricia Bouyer, Holger Hermanns, Jan Krćal and Nicolas Markey for fun and interesting collaborations. I had several visits both to LSV, ENS Cachan with Patricia and Nicolas and to Saarland University with Holger and Jan. Jan also joined me for a week at DTU (during which I was half sick most of the time). I have learned a lot from all of you and appreciated working with you a lot!

Next, I want to thank all my colleagues and friends in AlgoLoG at DTU for letting me have a great time for the past three years. I have looked forward to going to work every day and you are largely responsible for that. I am not going to mention all of your names fearing that I might forget one of them. I am going to miss you all and in particular, I am going to miss the coffee machine and the discussions.

My friends and family also deserve special thanks for always being there for me and being a great support during all parts of my life, including the PhD studies. In particular, many thanks to Pernille for proof reading the thesis.

Finally, I have saved the best for last. I want to thank my girlfriend Pil. You have been the person closest to me during the entire PhD and been there every step of the way. In this period you have followed me (and I have followed you) through all the ups and downs of everyday life. It also occurs to me, looking at the past three years, that everyday life is not very everyday like for us. There always seems to be new experiences, new challenges, new hopes and new dreams around the corner. I enjoy very much sharing the adventure of life with you and I look forward to what life brings next. In the end, I simply want to thank you for being you.

Contents

CHAPTER 1

Introduction

Formal modelling and verification of computing systems dates at least back to works of Alan Turing [Tur36] and Alonso Church [Chu40] that introduced respectively Turing machines and Lambda calculus as means to model computing systems. Impressively, these modelling formalisms are still valid and widely used in theoretical computer science today.

It was shown by Turing that the halting problem, i.e. the problem of verifying whether a given Turing machine halts on a given input, is undecidable. This is one of the earliest indications that formal verification of computing systems poses some very difficult challenges. Indeed, just deciding whether or not a given machine can reach a certain state is impossible!

To find verification problems that are decidable and tractable there are in general two ways to go. The first is to consider simpler, and less general, models of computation. The second is to ask simpler questions about these models. Just asking whether a machine will eventually stop is a quite simple question and therefore it seems that we need to restrict our model of computation in order to be able to verify anything interesting.

This has led to a plethora of different models of computing with varying generality. Some of the more prominent models of computing are finite-state automata and pushdown systems (see e.g. [HMU03]), counter-systems [Min61] and Petri nets [Mur89; Pet62]. In many of these models we obtain decidability of the reachability problem unlike for the Turing machine. Next, it is natural to ask: which questions can we decide on these less general models of computation? And which time and space resources will we need to decide them? These questions are some of the cornerstones of the area of formal verification of computing systems. These are also the types of questions we ask and answer in this thesis.

1.1 Games as models

Traditionally, the purpose of computers was to calculate things in an input-output fashion. They simply calculated the value of a function for a given input. Today, the purpose of computers has shifted from being mathematical calculators to being tools used for a wide variety of different things in a wide variety of different settings.

In particular, it has become increasingly important for computers to be able to *interact*. Not only with human users but also with other machines over networks as well as with their physical environment. This means that computers need to be

reactive to a higher degree than what used to be the case. That is, they need to be able to handle ongoing and non-deterministic behaviour from the surrounding world.

In order to model systems with several interacting components it is quite natural to use the framework of *games* which has been used for modelling interaction between entities in economics for many years (see e.g. [OR94]). In our setting, the players of the game correspond to the components of the system (possibly including an environment player) and further, strategies of these players correspond to programs for the components.

This means that when we ask whether there is a good strategy for a player in the game, we are asking whether there exists a good program for the corresponding component. In addition, if we can find such a strategy in the game model we would like to be able to translate the strategy to a program for the component.

As the systems we wish to model are reactive systems with theoretically infinite behavior (for instance servers, mobile phones, hardware circuits) it is natural that the games we use as models are infinite-duration games.

The idea to consider synthesis in infinite-duration settings such as this dates back to Alonso Church [Tho08] where circuit synthesis was considered in a setting similar to the games we consider here. For a survey on infinite-duration games, see [GTW02].

1.2 Model-checking

Model-checking is a very succesful model-based approach to automated verification where a system to be verified is being modelled [CGP01; BK08; Eme08]. Traditional modelling formalisms are non-deterministic transition systems [CE81; EH86; Pnu77; Hol04]), Markov chains [KNP02] and real-time systems [ACD93; LPY97].

The specification of the system is then expressed using a logical formula, often in a temporal logic such as the linear-time temporal logic LTL [Pnu77] or the computation tree logics CTL [CE81] and CTL* [EH86]. The act of model-checking is to decide, given a model M of a system and a specification φ in a suitable logic, whether the model M satisfies the specification φ. This is usually written $M \models \varphi$.

If the model satisfies the specification this provides a higher degree of certainty that the real system is correct with respect to the specification. Here, it is of course important that the model is as accurate a representation of the system as possible for the analysis to be meaningful.

If the model does not satisfy the specification it is also desirable if the model-checker can give a counter-example. That is, an explanation why the model does not satisfy the specification. In this case the counter-example might reveal problems with the real system that is being modelled.

We consider model-checking where the models are infinite-duration games and specifications are given in the *alternating-time temporal logic* ATL* [AHK02]. This makes it possible to specify properties such as "Player 1 can ensure that a target state is eventually reached" expressed by the ATL* formula

$$\langle\!\langle \{1\} \rangle\!\rangle \mathbf{F}\, \text{target}$$

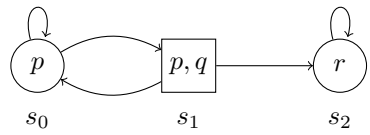


Figure 1.1: A turn-based game. Player 0 controls circle states and player 1 controls square states.

and "Player 2 and 3 can together ensure that a bad state is avoided" expressed by the ATL* formula

$$\langle\!\langle \{2,3\} \rangle\!\rangle \mathbf{G}\, \neg \mathrm{bad}$$

Here, $\mathbf{F}\varphi$ and $\mathbf{G}\varphi$ have the meaning "eventually φ is true" and "globally φ is true" for a given ATL* formula φ. The formula $\langle\!\langle A \rangle\!\rangle \varphi$ expresses that players in the set A can make sure that φ is true.

A *turn-based game* $\mathcal{G}$ between two players 0 and 1 is illustrated in Figure 1.1. Circle states are controlled by player 0 and square states by player 1. It is played by placing a token in an initial state. The player controlling the state must move the token along one of the transitions to a new state. Then the player controlling that state moves the token. This behaviour continues indefinitely. States are labelled with propositions p, q and r that are true in those states.

Now, asking whether $\mathcal{G}, s_0 \models \langle\!\langle \{0\} \rangle\!\rangle \mathbf{F} q$ is the same as asking whether player 0 has a strategy to ensure that q is eventually true when the game $\mathcal{G}$ begins in state s_0. Thus, we ask whether the formula $\langle\!\langle \{0\} \rangle\!\rangle \mathbf{F} q$ is true in the model $\mathcal{G}, s_0$ consisting of a turn-based game $\mathcal{G}$ and an initial state s_0. As player 0 can enforce this by moving the token from s_0 to s_1 this is in fact true.

We can show that neither player 0 nor player 1 can ensure on their own that the play eventually reaches s_2. Player 0 cannot ensure this as player 1 might move the token from s_1 to s_0 every time s_1 is reached. Player 1 cannot ensure this either as player 0 might keep moving the token along the self-loop back to s_0. However, together they can ensure that the token reaches s_2. This can be expressed as follows

- $\mathcal{G}, s_0 \not\models \langle\!\langle \{0\} \rangle\!\rangle \mathbf{F} r$
- $\mathcal{G}, s_0 \not\models \langle\!\langle \{1\} \rangle\!\rangle \mathbf{F} r$
- $\mathcal{G}, s_0 \models \langle\!\langle \{0,1\} \rangle\!\rangle \mathbf{F} r$

Indeed, by solving the model-checking problem we can automatically decide the truth of such statements.

One traditional problem that can be represented as a model-checking problem for alternating-time temporal logic is the *realizability problem* for LTL [PR89a; PR89b] which is the problem of deciding whether there exists a program satisfying a given LTL specification no matter how the environment behaves. This is closely related to the

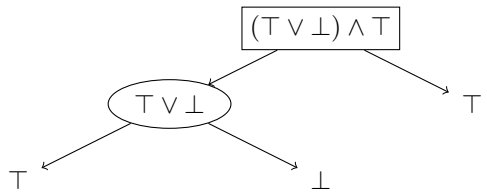


Figure 1.2: Example of a model-checking game.

synthesis problem which consists of generating a program meeting such a specification (see Section 1.4).

In addition to applications in synthesis and realizability the model-checking approach for alternating-time temporal logic is simply a natural approach for the verifation of multi-agent systems that has grown out of the traditions of linear-time and branching-time logics. Indeed, ATL* generalizes both LTL, CTL and CTL*. The important difference is that there is an explicit distinction between the non-deterministic choices made by different agents. In particular, this makes us able to reason about what can be achieved by a subset of the agents no matter how the remaining agents react.

Model-checking games

In addition to using games as models we also use the approach of *model-checking games* (also called evaluation games) to solving the model-checking problem (see e.g. [Sti95]). The idea is to reduce the question of whether a model M satisfies a formula φ, i.e. whether $M \models \varphi$, to deciding the winner in a game $\mathcal{G}$.

The intuition behind this is that $\mathcal{G}$ has two players 0 and 1 that are trying to prove and disprove $M \models \varphi$ respectively. If player 0 has a winning strategy then $M \models \varphi$ and if player 1 has winning strategy then $M \not\models \varphi$.

Example 1.1 *As an example, consider model-checking of a positive Boolean formula. A positive Boolean formula φ consists of variables $x_1, x_2, ..., x_m$, conjunctions and disjunctions. A model M of such a formula is an assignment of truth values ($\top$ for true and $\bot$ for false) to the variables. Given such a model it is possible to decide whether φ is true in the model M using a model-checking game.*

For the particular formula $\varphi = (x_1 \vee x_2) \wedge x_3$ and the model M assigning $\top$ to x_1 and x_3 and $\bot$ to x_2 the corresponding model-checking game can be seen in Figure 1.2.

The game starts by placing a token in the root node of this game tree labelled with the instantiated formula $\varphi[x_1 \mapsto \top, x_2 \mapsto \bot, x_3 \mapsto \top]$. Then, if the main connective of the formula is a conjunction player 1 must move the token along one of the edges. If the main connective of the formula is a disjunction then player 0 must move the

token along one of the edges. The choices of the players correspond to picking one of the subformulas connected by the main connective.

In the figure player 0 *controls the rounded node and player* 1 *controls the rectangle node. The players keep moving the token until a node is reached which is labelled* $\top$ *or* $\bot$*. If* $\top$ *is reached player 0 wins and if* $\bot$ *is reached player 1 wins.*

The intuition behind the construction is that if a conjunctive formula is false then one of the conjuncts must be false. Then player 1 *can show it false by picking such a conjunct. On the other hand, if a disjunctive formula is true then one of the disjuncts must be true. Then player* 0 *can show it true by picking such a disjunct.*

In the example the formula evaluates to true as both conjuncts are true. To show this player 0 *must choose* $\top$ *rather than* $\bot$ *to show that the disjunct of the left-most conjunct of* φ *is true. This winning strategy implies that* φ *is true in the model* M*.*

In Chapter 4 we consider new algorithms for solving a class of games called *parity games* [EJ91]. An important application of parity games is that they are the model-checking games that arise from model-checking of the model μ-calculus in finite-state transition systems [Sti95]. Thus, if we can solve parity games efficiently we can solve the model-checking problem of the modal μ-calculus efficiently.

In Chapter 5 we will apply the technique of model-checking games to develop algorithms for model-checking ATL and ATL* with quantitative extensions where the models are one-counter games. These are some of the first decidability results for ATL* in infinite-state systems.

In Chapter 6 we study a *symmetry reduction* technique that can be used to reduce state spaces in various problems related to games in the presence of symmetry. This includes both solving parity games and model-checking of alternating-time temporal logics. It has yet to be seen how large efficiency gains this technique will give in practice. However, in model-checking of transition systems [ID96; Cla+96; Cla+98], real-time systems [Hen+03] and probabilistic systems [KNP06] this technique has had a large impact of the applicability of model-checking.

1.3 Satisfiability

Whereas the model-checking problem considers the truth of a formula φ in a particular model M the *satisfiability problem* asks, given a specification φ, whether there exist a model satisfying this specification. This is closely related to the validity problem which asks whether the specification φ is true for every model M. Indeed, a formula φ is unsatisfiable if and only if $\neg\varphi$ is valid.

In the classical Boolean satisfiability problem the models are assignments of truth values to the variables. Thus, the Boolean satisfiability problem asks whether there exists a truth assignment of the variables that makes the formula true.

In Chapter 7 we consider satisfiability problems for fragments of the alternating-time temporal logic ATL*. Here, recall that the models of formulas are games. This means that the satisfiability problem for an ATL* formula φ asks whether there exists a game in which φ is true.

Example 1.2 *An example of a satisfiable formula is* $\langle\langle\{0,1\}\rangle\rangle \mathbf{F} r$. *Indeed, a model satisfying this formula was given in Figure 1.1. There it was shown that* $\mathcal{G}, s_0 \models \langle\langle\{0,1\}\rangle\rangle \mathbf{F} r$ *implying that the formula is satisfiable.*

An example of an unsatisfiable formula is $\langle\langle\{0\}\rangle\rangle \mathbf{G} p \wedge \langle\langle\{1\}\rangle\rangle \mathbf{G} \neg p$. *It expresses that player* 0 *has a strategy to force* p *to be true forever and that player* 1 *has a strategy to force* p *to be false forever. This is not true in any game. Thus, the formula is unsatisfiable.*

As ATL* formulas can express properties about the abilities of players the satisfiability problem is in fact quite closely related to the area of *mechanism design* (see e.g. [Nis+07]). Indeed, mechanism design is a discipline in economics where the objective is to design games with desirable properties. By solving the satisfiability problem for ATL* we can automatically decide whether a game meeting a certain specification even exists.

As Boolean satisfiability is already NP-complete [Coo71] and this problem is a special case of all the satisfiability problems we consider, we can in general expect a quite high complexity for these problems. Indeed, satisfiability for ATL* is 2ExpTime-complete [Sch08].

In Chapter 7 we study the computational complexity of various *flat fragments* of ATL*. That is, fragments where nesting of the strategic operator $\langle\langle\cdot\rangle\rangle$ is not allowed. This restriction is considered to obtain a better balance between computational complexity and expressiveness. Indeed, many interesting properties can still be expressed in the flat fragments of ATL*. Another reason is that it can been argued that the semantics of nesting quantifiers in ATL* is not natural [ÅGJ07; Bri+09].

1.4 Synthesis and Realizability

When doing model-checking and satisfiability testing of ATL* formulas we decide the truth of specifications about capabilities of players in games. A similar problem was also tackled in [PR89a; PR89b], namely the *realizability problem* which asks to decide if there exists a program for a system satisfying a given LTL specification no matter how the environment behaves. The very related *synthesis problem* asks to generate such a program if it exists.

A two-player game can be constructed modelling the situation with a controller and an uncooperative environment. A controller player represents the controller and an environment player represents the non-deterministic environment. The goal of the controller player is that the specification is satisfied no matter how the environment player behaves. As such, a strategy for the controller player that is winning no matter how the environment player behaves corresponds to a control program satisfying a specification no matter which input it receives from the environment.

When we do model-checking we decide existence of certain strategies, analogously to the realizability problem. However, to be practical it is also important that we are able to construct winning strategies and not just decide existence of them. This is because the strategies correspond to actual programs that we want to generate.

The techniques we apply in this thesis are constructive in the sense that the algorithms which decide existence of certain strategies work by finding particular strategies as witnesses of the existence. Thus, whereas the purpose of traditional model-checking is to verify correctness of an already implemented program, model-checking of games gives us not just the opportunity to verify properties of multi-agent systems, but also to *synthesize* programs from the specification. The synthesized program will then, by construction, be correct. Correctness is of course provided that our specification does in fact specify how we want the program to behave. For a discussion on the difficulty of specification for synthesis, see [Blo+14].

Thus, our model-checking approach allows us to model situations where a system is not yet built contrary to in traditional model-checking. The purpose is instead to synthesize a correct system from a specification. See [PPS06; Blo+12] for applications of this approach to practical case studies.

1.5 Qualitative and quantitative objectives

Recently, the game-based approach to program synthesis has had more focus on specifications that are not just *qualitative* , but also *quantitative* . Instead of just being able to require qualitative correctness properties such as

- is it possible to ensure reaching a target state?
- is it possible to avoid deadlocks?
- is it possible to keep getting access to the database?

we also want to require that the correct system behaves well according to some performance metrics. We can then ask quantitative questions like

- Can a target state be reached fast?
- How much energy is needed to accomplish a goal?
- Can a device avoid ever running out of battery?

The study of such quantitative objectives is currently an active research area, see e.g. [Blo+09; Hen12; BG13]. In Chapter 5 we will extend ATL* to be able to express combined qualitative and quantitative objectives in games with an *unbounded counter* as was done in [BG13]. Such a counter could for instance model things like battery level, time or even money. We provide some of the first decidability results for ATL* (as well as for the quantitative extension QATL*) in infinite-state systems.

1.6 Models for distributed systems

For the model-checking and satisfiability problems we consider in Part I all the models are *full observation games*. However, in many applications of reactive systems, and particularly *distributed systems* , it is an inherent property that the different entities do not fully observe the state of the other entities. Modelling formalisms for such systems is the topic of Part II. These are *partial-observation games*.

Example 1.3 *As an example of a distributed system with partial observation consider a system with a number of clients communicating with a server that can grant access to a printer. Here, a client does not know the state of the other clients and, in particular, does know whether another client is currently using the printer. To model such situations partial observation of the current state is vital.*

Many settings of infinite-duration games with partial observation have a very high complexity compared to the full observation case. For instance, it was shown in [PR90] that deciding existence of winning strategies in turn-based games with partial observation and LTL objectives is undecidable. In [BK10] it was even shown that the same question is undecidable already for safety specifications. See also [FS05] for a discussion of this phenomenon.

To combat this issue there has been a number of attempts at defining more restricted ways to define partial observation and also restrictions on the structures themselves. See e.g. [Gen+13; FO14; Mus15].

In Chapter 9 and Chapter 10 we study similar models and also obtain a number of similar undecidability results. In Chapter 10 we also prove undecidability of almost-sure reachability in the formalism of *decentralized Markov decision processes* already for 2 players. In both chapters we also provide decidability results. In Chapter 9 these results are obtained by restricting to finite-memory strategies for the players whereas a decidability result is obtained in Chapter 10 by making certain restrictions on the structure of the game.

1.6.1 Non-zero sum objectives

In Part I all the situations we analyze boil down to so-called *zero-sum games* . That is, games where the objective of one player is the opposite of the other. Consider again the game between a controller player and an environment player. Here, the objectives of the two players are exactly the opposite.

In distributed settings, where there are often more than two players, it can be too restrictive to only consider zero-sum objectives. Indeed, entities such as the clients in Example 1.3 are typically interested in what is best for themselves but do not gain by other clients having bad performance.

In such a setting it is also too restrictive for each player to have worst-case assumptions about the other players in the game. It is not rational for a client to assume that all other clients will try to work against it, it is rational to assume that all other clients will try to do what is best for themselves.

These considerations lead us to the study of *non-zero sum games* as known from classical game theory (see e.g. [OR94]). Here, *Nash equilibria* Nash equilibrium is one of the most applied solution concepts. A Nash equilibrium specifies a strategy for all players in a game such that no single player can deviate from this specification and improve his payoff given that all other players stick to the specification.

In Chapter 9 we study a novel modelling formalism for distributed systems called *symmetric game networks* . In this setting symmetry means that several of the players in the game are identical with identical goals. Example 1.3 is a good example of such a setting. We study the problem of deciding existence of *symmetric Nash equilibria* in such a setting. That is, a Nash equilibria where all players use the same strategy.

1.6.2 Timed and stochastic behavior

In Chapter 10 we present another new modelling formalism for distributed systems called *distributed interactive Markov chains* that includes details about the timed behaviour of the systems. Here, time is assumed to be continuous contrary to the other chapters where time is assumed to be discrete.

The models also include stochastic transitions with probability distributions over the duration of these transitions. This is done in a way similar to the paradigm of *continuous-time Markov chains*.

Finally, the models includes explicit communication primitives making it more natural to model communication between the different players in a model.

As these models are already quite complicated we consider simpler objectives than in previous chapters. Indeed, we focus on situations where the players collaborate towards maximizing the probability of reaching a specified set of target states.

1.7 Goals

The purpose of this thesis is to advance the current knowledge of foundational problems within game-based synthesis and verification as well as improve existing techniques within this area. Our goals can be divided into three main categories:

1. Improving the efficiency of existing techniques for automated verification and synthesis. As computational resources such as time and space are the main bottlenecks for the success of game-based techniques such improvements have the potential to extend the application scope of this approach significantly.

2. Deciding the computational complexity of a number of foundational model-checking, satisfiability and synthesis problems. Such results advance the current insight into the possibilities and future promises of game-based techniques. In particular it is important for deciding the direction of further research and development in the area.

3. Developing new formalisms for modelling of distributed and reactive systems. As game-based synthesis of distributed and reactive systems is still a young

field there are still rich possibilities for introducing new modelling formalisms. In particular, we consider the problems of modelling such systems that contain many identical components. We also consider the problem of modelling game-based distributed settings where time is continuous and duration of delays are modelled by continuous probability distributions.

1.8 Thesis outline

The thesis begins with an introduction and a general chapter on preliminaries. The rest of the thesis is divided into two main parts. Part I considers settings with full observation games and Part II considers settings with partial observation games. For each part there is an introduction chapter. Then follows chapters presenting the results obtained during the Ph.d. studies. Specifically:

- Chapter 1 is an introductory chapter
- Chapter 2 is a general chapter containing preliminaries
- Part I contains the following chapters:
  - Chapter 3 introduces Part I on full observation games.
  - Chapter 4 studies the algorithmic problem of solving parity games. In particular, the novel concept of winning cores in parity games are analyzed and used to develop an under-approximation algorithm for solving parity games. Practical experiments are very promising compared to existing algorithms. It is based on [Ves16].
  - Chapter 5 presents computational complexity results on model-checking quantitative extensions ATL* and ATL in one-counter games. Algorithms based on model-checking games are presented and matching lower bounds are shown. It is based on [Ves15].
  - Chapter 6 develops the symmetry reduction technique for turn-based games with finite branching. The technique is shown to be applicable both for parity game solving and model-checking of ATL*. It is based on [MV14].
  - Chapter 7 presents computational complexity results on satisfiability for various flat fragments of ATL*. That is, fragments where the strategic operators $\langle\!\langle \cdot \rangle\!\rangle$ cannot be nested. It is based on [GV14].
- Part II contains the following chapters:
  - Chapter 8 introduces Part II on partial observation games
  - Chapter 9 develops the new modelling formalism of *symmetric game networks*. This setting is used to model distributed settings where several players are identical. Several undecidability and complexity results are shown for questions concerning the existence of symmetric Nash equilibria in such models. It is based on [BMV14] and [BMV16].

  - Chapter 10 introduces another new modelling formalism, *distributed interactive Markov chains*. The purpose of this formalism is to model distributed settings with continuous time and probability distributions over durations of transitions. A number of undecidability results are shown. We present a decidability result for the value problem for 2-player non-urgent models. It is based on [HKV16].

- Chapter 11 contains a conclusion of the thesis.

CHAPTER 2

Preliminaries

In this chapter we introduce a number of definitions and concepts which will be of use to us for the rest of the thesis. This includes basic models, games, automata as well as different logics. Further, we introduce the complexity classes that can be encountered throughout the thesis.

2.1 Modelling formalisms

In this thesis the focus is on games as *modelling formalisms* for reactive and multi-agent systems. Except for the models in Chapter 10 we deal only with discrete-time models without probabilistic choices. That is, discrete-state systems where all non-determinism stems from choices made by the players in the games. All games we consider are infinite duration games as the target applications are systems that should be able to run forever.

We consider both turn-based games where players alternate turns as well as concurrent games where players simultaneously and independently choose actions. Further, in Chapter 9 and 10 we consider games where players have only partial information about the state of the game.

2.1.1 Turn-based games

Turn-based games comprise a simple, yet powerful way to model interaction. We focus on computational aspects of various subclasses of turn-based games in Chapter 4, 5 and 6; all games considered are games of infinite duration.

A turn-based game is played by a finite number of players. It is played in a finite transition system where the states are partitioned according to the players. We say that a player *controls* the states in his partition. The game is played by placing a token in an initial state s_0 of the transition system. The player controlling the current state must choose a successor state to move the token to while respecting the transition relation. Then the player controlling the successor state chooses a new successor state and so on indefinitely or until there is no outgoing transition from the current state. Thus, the play is either a finite or infinite sequence of states.

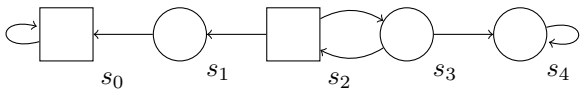


Figure 2.1: A turn-based game.

Definition 2.1 *A* turn-based game *is a tuple* $\mathcal{G} = (S, \Pi, (S_j)_{j\in\Pi}, R)$ *where*

- S *is a non-empty set of* states
- Π *is a non-empty, finite set of* players
- $S_j \subseteq S$ *is the set of states that player* $j \in \Pi$ *controls*
- $S_i \cap S_j = \emptyset$ *when* $i \neq j$ *and* $S = \bigcup_{j\in\Pi} S_j$
- $R \subseteq S \times S$ *is the* transition relation

Example 2.2 *A simple example of a turn-based game is shown in Figure 2.1. In this game there are two players* 0 *and* 1*. Circle states are in* S_0 *and square states in* S_1*. In other words, player* 0 *controls the circle states and player* 1 *controls the square states. There is an arrow from state* s *to state* t *if* $(s,t) \in R$*. In this game every state has an outgoing transition and therefore every play of this game is an infinite sequence of states.*

Let us fix a turn-based game $\mathcal{G} = (S, \Pi, (S_j)_{j\in\Pi}, R)$ in the following. A turn-based game is played by placing a token on an initial state $s_0 \in S$. Then it proceeds for a number of rounds as follows. In each round the player who controls the current state s (the state where the token is placed) must choose to move the token to a state t such that $(s,t) \in R$ if one such exists. In this case we call (s,t) a *legal move*. The game is played in this fashion for an infinite number of rounds or until there is no legal move in the current state.

More formally, for a set X we denote by X^*, X^+ and X^ω the set of finite sequences, non-empty finite sequences and infinite sequences of elements from X respectively. For a sequence $\rho = s_0 s_1 ...$ and $i \in \mathbb{N}_0$ we define $\rho_i = s_i$, $\rho_{\leq i} = s_0 ... s_i$ and $\rho_{\geq i} = s_i s_{i+1} ...$. When ρ is finite, i.e. $\rho = s_0 ... s_\ell$ we write $\text{last}(\rho) = s_\ell$ for the last state in ρ and $|\rho| = \ell$ for the number of transitions in ρ.

A *play* is either an infinite sequence $s_0 s_1 ... \in S^\omega$ such that $(s_i, s_{i+1}) \in R$ for all $i \geq 0$ or a non-empty finite sequence $s_0 ... s_\ell \in S^+$ such that $(s_i, s_{i+1}) \in R$ for all $0 \leq i < \ell$ and with the requirement that there does not exist t so $(s_\ell, t) \in R$. The set of plays is denoted $\text{Play}(\mathcal{G})$. For $s_0 \in S$ the set of plays with initial state s_0 is denoted $\text{Play}(\mathcal{G}, s_0)$. An *objective* for a player is a subset of $\text{Play}(\mathcal{G})$.

A *history* is a non-empty, proper prefix of a play. The set of all histories (respectively histories with initial state s_0) is denoted $\text{Hist}(\mathcal{G})$ (respectively $\text{Hist}(\mathcal{G}, s_0)$).

Note that a *transition system* is simply a turn-based game with a single player. In this case we simply omit the set of players and the partition of the states and write (S, R) for the transition system with the set S of states and transition relation R.

Strategies

In a turn-based game $\mathcal{G}$ a *strategy* for player $j \in \Pi$ is a partial function

$$\sigma_j : \mathrm{Hist}(\mathcal{G}) \to S$$

defined for histories $s_0 ... s_\ell$ such that $s_\ell \in S_j$ with a requirement that $(s_\ell, \sigma_j(s_0 ... s_\ell)) \in R$. We say that an infinite play ρ is *compatible* with a strategy σ_j if $\sigma_j(\rho_{\leq i}) = \rho_{i+1}$ for every $i \geq 0$ such that $\rho_i \in S_j$. For a finite play ρ we require the same but only for $0 \leq i < |\rho|$. The set of plays with initial state s_0 compatible with σ_j is denoted $\mathrm{Play}(\mathcal{G}, s_0, \sigma_j)$. Compatibility of histories with strategies is defined analogously. The set of histories with initial state s_0 compatible with σ_j is denoted $\mathrm{Hist}(\mathcal{G}, s_0, \sigma_j)$.

Depending on the type of game, a player does not always need to remember the entire history of the play in order to play well. In some games a bounded amount of memory suffices to win and in others a play might need no memory at all except knowledge of the current state of the game. One advantage of showing that a small amount of memory is enough in order to win in a certain class of games is that it can make the search for a winning strategy easier. This trick is used later in the thesis.

We define a *bounded-memory strategy* as a certain type of *deterministic finite-state transducer* (DFST) which is a finite-state automaton extended with output symbols on the transitions. The idea is that the DFST reads the states of the game during the play and in each game step it updates its memory and outputs a symbol according to what is observed.

Definition 2.3 *A deterministic finite-state transducer is a tuple* $\mathcal{T} = (M, m_0, \Sigma, \Gamma, \delta, \mathcal{O})$ *where*

- M *is a finite set of* states
- $m_0 \in M$ *is the* initial state
- Σ *is a set of* input symbols
- Γ *is a set of* output symbols
- $\delta : M \times \Sigma \to M$ *is the* transition function
- $\mathcal{O} : M \times \Sigma \to \Gamma$ *is the* output function.

Note in particular that we allow a DFST to have infinite sets of input and output symbols. This choice makes the DFST a meaningful representation of a bounded-memory strategy for games with an infinite set of states.

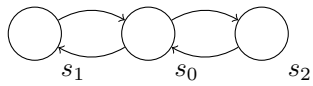


Figure 2.2: A 1-player turn-based game.

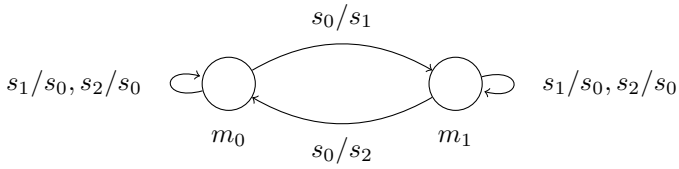


Figure 2.3: DFST representing a 2-memory strategy for the single player in the game in Figure 2.2.

We say that a strategy σ for player j in $\mathcal{G}$ is a bounded-memory strategy if there exists a DFST $\mathcal{T} = (M, m_0, S, S, \delta, \mathcal{O})$ with S being both the set of input symbols and output symbols of $\mathcal{T}$ and such that for every history $h = s_0...s_\ell$ in $\mathcal{G}$ such that $s_\ell \in S_j$ we have

$$\sigma(h) = \mathcal{O}(\delta(...\delta(\delta(m_0, s_0), s_1), ..., s_{\ell-1}), s_\ell)$$

In addition, we say that σ is represented by the DFST $\mathcal{T}$. Further, if $\mathcal{T}$ has k states we say that σ is a k-memory strategy or that σ has size k.

A *memoryless strategy* is a strategy such that $\sigma(h) = \sigma(h')$ for all histories h and h' with the same final state. Memoryless strategies are precisely the 1-memory strategies and can be seen as functions from S_j to S respecting the transition relation. For memoryless strategies we abuse notation and write $\sigma(s) = s'$ if $\sigma(h) = s'$ for every history h with final state s.

Example 2.4 *Consider the one-player game in Figure 2.2. Suppose that the play begins in s_0 and that the player wins if he keeps visiting both s_1 and s_2 infinitely often. Clearly he can make sure to win as he controls all the states. However, he cannot do so using a memoryless strategy. Indeed, suppose there was a memoryless winning strategy σ. Then either $\sigma(h) = s_1$ for all histories h ending in s_0 or $\sigma(h) = s_2$ for all histories h ending in s_0. In both cases the goal of the player will not be accomplished as he makes the same decision every time s_0 is reached using a memoryless strategy. However, he can win using the 2-memory strategy represented by the DFST depicted in Figure 2.3. Here, a transition from state m to m' labelled by s/t means that $\delta(m, s) = m'$ and $\mathcal{O}(m, s) = t$ where δ and $\mathcal{O}$ are the transition function and output function of the DFST respectively. This strategy makes sure that the player chooses s_1 every other time s_0 is reached and s_2 every other time s_0 is reached.*

A *coalition* in a game $\mathcal{G}$ is a subset $A \subseteq \Pi$ of players in the game. A strategy σ_A for a coalition A is a tuple $(\sigma_a)_{a \in A}$ of strategies, one for each player $a \in A$. A strategy for the coalition Π consisting of all players is called a *strategy profile*.

2.1.2 Concurrent games

In Chapter 7 and 9 we will use a more general class of models than turn-based games, namely *concurrent games* [AHK02]. As in a turn-based game, a concurrent is played by placing a token in an initial state and then an infinite number of rounds are played where the token moves from state to state in each round. However, unlike in turn-based games, the movement of the token can be affected by all the players in the game as follows: In each round, every player has a non-empty set of *actions* available. Concurrently and independently each player chooses one of the actions available to him. Based on these choices of actions and the current state a transition function assigns the successor state deterministically. Concurrent games can be seen as a subclass of stochastic games [Sha53] with a deterministic transition function.

Definition 2.5 *A concurrent game is a tuple* $\mathcal{C} = (S, \Pi, \Sigma, \Gamma, \delta)$ *where*

- S *is a set of* states
- Π *is a finite, non-empty set of* players
- Σ *is a set of* actions
- $\Gamma : S \times \Pi \to 2^{\Sigma} \setminus \{\emptyset\}$ *is a function specifying for each state* s *and player* j *the set of actions available to player* j *in state* s
- $\delta : S \times \Sigma^{\Pi} \to S$ *is the* transition function *which assigns a successor state when the current state* s *is given and an available action for each player in* s.

Example 2.6 *We model the game of Rock-paper-scissors as a two-player concurrent game in Figure 2.4. An arrow from state* s *to state* t *labelled by* (a_1, a_2) *means* $\delta(s, (a_1, a_2)) = t$ *where* δ *is the transition function. Note that in every state, each player must have at least one action available. When a player has won the game we let each player choose the void action* $*$ *indefinitely.*

We fix a concurrent game $\mathcal{C} = (S, \Pi, \Sigma, \Gamma, \delta)$ in the following. A *play* is an infinite sequence $s_0 s_1 ... \in S^{\omega}$ such that for all $i \geq 0$ there exists actions $a_j \in \Gamma(s, t)$ for every $j \in \Pi$ such that $\delta(s_i, (a_j)_{j \in \Pi}) = s_{i+1}$. A *history* is a non-empty finite prefix of a play. The set of histories and plays in $\mathcal{C}$ is denoted as for turn-based games.

Strategies are defined analogously to turn-based games, but are functions from histories to actions. Formally, a *strategy* for player j is a function

$$\sigma : \mathrm{Hist}(\mathcal{C}) \to \Sigma$$

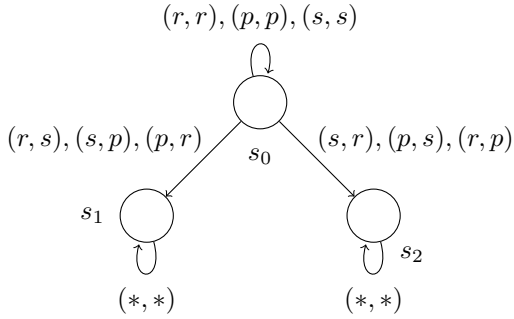


Figure 2.4: Rock-paper-scissors as a concurrent game.

such that $\sigma(h) \in \Gamma(\text{last}(h), j)$ for every history h and player j. That is, a strategy can only choose an action that is available to the player at the current state of the play.

Note that turn-based games can be seen as the special case of concurrent games where for each state $s \in S$ there exists at most one player j such that $|\Gamma(s,j)| > 1$. In other words, in every state of the game there is at most one player with a choice affecting the successor state.

Memoryless strategies and bounded-memory strategies are defined analogously to in turn-based games. This is also the case for strategies for coalitions of players.

2.1.3 Partial observation games

In Part II we study games where the players only have *partial observation* of the current state of the game. This makes us able to model more interesting scenarios but also gives rise to more complex problems.

To model partial observation we use the notion of *observation sets* for each player similarly to the concurrent games in [AHK02], game structures in [Cha+06] and partially observable Markov decision processes. An observation set for a player is a set of states that the player cannot distinguish between. As such, for every player we partition the set of states into observation sets. This gives rise to an equivalence relation $\sim_j$ for player j defined for all states s, s' such that $s \sim_j s'$ means that player j cannot distinguish between s and s'. For this to be meaningful we require that the set of actions available to player j in s and s' are the same when they are indistinguishable. Formally, we define concurrent games with partial observation as follows:

Definition 2.7 *A concurrent game with partial observation is a tuple* $\mathcal{C} = (S, \Pi, \Sigma, \Gamma, \delta, (\sim_j)_{j \in \Pi})$ *where*

- $(S, \Pi, \Sigma, \Gamma, \delta)$ *is a* concurrent game

- $\sim_j \subseteq S \times S$ *is an* equivalence relation *for every player* $j \in \Pi$
- *For all* $s, s' \in S$ *and* $j \in \Pi$ *we require that* $s \sim_j s'$ *implies* $\Gamma(s, j) = \Gamma(s', j)$.

The equivalence relation $\sim_j$ induces a set of equivalence classes on states for each player j. We call such an equivalence class an observation set. We denote by $[s]_j$ the observation set that state s belongs to for player j. This set contains the set of states which are indistinguishable to s from the point of view of player j.

Plays and histories are defined as for concurrent games with full observation. However, in order to define strategies we need the notion of *observation history*. For a given history $h = s_0...s_\ell$ in $\mathcal{G}$ we define the corresponding observation history $[h]_j$ for player j by

$$[h]_j = [s_0]_j...[s_\ell]_j$$

We denote the set of observation histories for player j by $\text{ObsHist}_j(\mathcal{G})$. Instead of being functions from histories to actions, a strategy for player j in a concurrent game with partial information $\mathcal{G}$ is a mapping $\sigma : \text{ObsHist}_j \to \Sigma$ such that $\sigma(o_0...o_\ell) \in \Gamma(s, j)$ for every state $s \in o_\ell$. That is, a player can only base his decision on the sequence of observations seen so far during the play and not on the sequence of states that actually occured so far during the play. Note that full observation concurrent games is just the special case where $\sim_j$ is the identity relation for every player j.

Another way to view the strategies in partial information games is as mappings from histories to actions just as in the full observation case. However, to be compatible with the partial observation a strategy must satisfy certain constraints. Let σ be such a strategy for player j. We say that σ is $\sim_j$*-realizable* (or simply realizable if $\sim_j$ is given by the context) if for all histories h, h' with $[h]_j = [h']_j$ we have $\sigma(h) = \sigma(h')$. The set of realizable strategies are exactly the same as the set of strategies as defined by mappings from observation histories to actions. Depending on the context we will use both definitions of strategies in partial observation games.

Example 2.8 *An example of a concurrent game with partial information can be seen in Figure 2.5. The games has two players, player 0 and player 1. Initially, player 0 chooses either* $+$ *or* $-$ *in* s_0*. Afterwards, player 1 also gets to choose between* $+$ *or* $-$*. However, player 1 cannot observe what player 0 chose because* $s_1 \sim_1 s_2$*. This means that player 1 cannot make sure that the play ends up in* s_3 *as he does not know which action player 0 chose initially.*

In Chapter 9 a special class of concurrent games with partial information, game networks, is introduced and investigated. In Chapter 10 we will also consider models with probabilistic transitions functions and partial observation such as partially observable Markov decision processes (POMDPs) and decentralized partially observable Markov decision processes (DEC-POMDPs). In these cases partial observation is included in the models analogously to in concurrent games using observation sets.

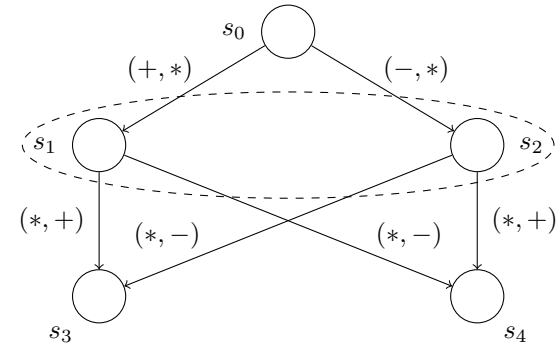


Figure 2.5: A concurrent game with partial information. Here, s_1 and s_2 are indistinguishable to player 1 as illustrated by the dashed ellipse. We have omitted self-loops from s_3 and s_4 in the figure for simplicity.

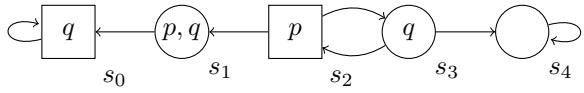


Figure 2.6: A labelled turn-based game.

Labelling

In order to use games as models for specification logics we assign meaning to different states using a finite set AP of *proposition symbols*. A *labelling function* L assigns a set of propositions true in a given state.

The turn-based game $(S, \Pi, (S_j)_{j\in\Pi}, R)$ illustrated in Figure 2.6 have assigned proposition symbols from the set $\mathrm{AP} = \{p, q\}$ to the different states. For instance, $L(s_2) = \{p\}$ and $L(s_4) = \emptyset$.

We call $\mathcal{G} = (S, \Pi, (S_j)_{j\in\Pi}, R, \mathrm{AP}, L)$ where $L : S \to 2^{\mathrm{AP}}$ a labelled turn-based game. In a similar way we can extend transition systems and concurrent games to labelled transition systems and labelled concurrent games.

2.2 Logics

As specifications for systems we use various different *logics*. In particular, we use *temporal logics* which are capable of expressing qualitative properties about infinite executions of systems. The widely used *linear-time temporal logic* (LTL) [Pnu77] is capable of expressing properties of infinite executions whereas the *computation tree logics* CTL [CE81] and CTL* [EH86] are used to express properties of infinite trees of execution.

A natural way of generalizing these logics to systems where several players are responsible for non-deterministic choices is given by the *alternating-time temporal logics* ATL and ATL* [AHK02]. With these logics it is possible to express capabilities of coalitions of players of the form $\langle\langle A\rangle\rangle\Phi$ which, informally, means that coalition A has a strategy to force that the property Φ is true no matter what the other players in the game do. Such formulas can be nested and combined with the usual temporal operators known from LTL. CTL and CTL* are the one-player fragments of ATL and ATL* respectively.

Chapters 5, 6, 7 and 9 all consider automated verification of properties specified in these temporal logics.

2.2.1 Alternating-time temporal logic

We first introduce the alternating-time temporal logic ATL* [AHK02] and then define branching-time logics and linear-time logics as fragments of ATL*. Formulas of ATL* are interpreted in labelled concurrent games and is used to express capabilities of coalitions of players in such a game. The most interesting quantifier in ATL* is the *strategic quantifier* $\langle\langle A\rangle\rangle$ for a coalition A of players in a game. The formula $\langle\langle A\rangle\rangle\Phi$ expresses that the players in coalition A can force the ATL* formula Φ to be true. It can be seen that ATL* is a generalization of CTL* (and thus also CTL and LTL) by noting that the *universal path quantifier* $\mathbf{A}$ and the *existential path quantifier* $\mathbf{E}$ are equivalent to $\langle\langle\emptyset\rangle\rangle$ and $\langle\langle\Pi\rangle\rangle$ respectively, where Π is the set of players in the game.

Syntax

There are two types of formulas in ATL* like for CTL*: *state formulas* that are evaluated at states and *path formulas* that are evaluated on plays. The state formulas of ATL* are defined by the grammar

$$\varphi ::= p \mid \neg\varphi_1 \mid \varphi_1 \wedge \varphi_2 \mid \langle\langle A\rangle\rangle\Phi_1$$

where $p \in \mathrm{AP}$ is an atomic proposition, $A \subseteq \Pi$ is a coalition, φ_1 and φ_2 are ATL* state formulas and Φ_1 is an ATL* path formula. The path formulas of ATL* are defined by the grammar

$$\Phi ::= \varphi_1 \mid \neg\Phi_1 \mid \Phi_1 \wedge \Phi_2 \mid \mathbf{X}\Phi_1 \mid \Phi_1\mathbf{U}\Phi_2$$

where φ_1 is an ATL* state formula and Φ_1 and Φ_2 are ATL* path formulas.

The other Boolean connectives are defined as usual. The temporal operator $\mathbf{F}$ specifying eventual truth is defined by $\mathbf{F}\Phi = \top\mathbf{U}\Phi$ and the temporal operator $\mathbf{G}$ specifying global truth is defined by $\mathbf{G}\Phi = \neg\mathbf{F}\neg\Phi$. The release operator $\mathbf{R}$ is defined by $\Phi_1\mathbf{R}\Phi_2 = \neg(\neg\Phi_1\mathbf{U}\neg\Phi_2)$. To keep the notation lighter we will sometimes list the members of A in $\langle\langle A\rangle\rangle$ without using $\{\}$.

The fragment ATL$^+$ of ATL* is obtained when the temporal operators may only be applied to state formulas, i.e. when path formulas are re-defined as

$$\Phi ::= \varphi_1 \mid \neg\Phi_1 \mid \Phi_1 \wedge \Phi_2 \mid \mathbf{X}\varphi_1 \mid \varphi_1\mathbf{U}\varphi_2 \mid \varphi_1\mathbf{R}\varphi_2$$

where φ_1 and φ_2 are ATL$^+$ state formulas and Φ_1 and Φ_2 are ATL$^+$ path formulas.

The further restricted fragment ATL only contains state formulas of the following form

$$\varphi ::= p \mid \neg\varphi_1 \mid \varphi_1 \wedge \varphi_2 \mid \langle\langle A\rangle\rangle \mathbf{X}\varphi_1 \mid \langle\langle A\rangle\rangle(\varphi_1 \mathbf{U} \varphi_2) \mid \langle\langle A\rangle\rangle(\varphi_1 \mathbf{R} \varphi_2)$$

where $A \subseteq \Pi$ is a coalition, $p \in \mathrm{AP}$ is a proposition and φ_1 and φ_2 are ATL formulas.

Remark 2.9 *Note that for* ATL$^+$ *and* ATL *we include the release operator* $\boldsymbol{R}$ *in the syntax. The reason is that while* $\boldsymbol{R}$ *can be defined using the other operators in* ATL* *as shown above it was shown in [LMO08] that it cannot be defined in* ATL *when only the temporal operators* $\boldsymbol{U}$ *and* $\boldsymbol{X}$ *are available.*

Given a set AP of atomic propositions, the set of *literals* over APis $\mathrm{AP} \cup \{\neg p \mid p \in \mathrm{AP}\}$.

Semantics

The *semantics* of ATL* formulas is given with respect to a labelled concurrent game $\mathcal{G} = (S, \Pi, \Sigma, \Gamma, \delta, \mathrm{AP}, L)$ and defined by two semantics relations

- $\mathcal{G}, s \models \varphi$ meaning that the state formula φ holds at state s in $\mathcal{G}$
- $\mathcal{G}, \rho \models \Phi$ meaning that the path formula Φ holds along the play ρ in $\mathcal{G}$

The two semantic relations are defined by mutual recursion. For state formulas it is given as follows where $s \in S$, $p \in \mathrm{AP}$, φ_1 and φ_2 are ATL* state formulas, Φ is a an ATL* path formula and $A \subseteq \Pi$ is a coalition

$$\begin{array}{ll}
\mathcal{G}, s \models p & \text{if } p \in L(s) \\
\mathcal{G}, s \models \neg\varphi_1 & \text{if } \mathcal{G}, s \not\models \varphi_1 \\
\mathcal{G}, s \models \varphi_1 \wedge \varphi_2 & \text{if } \mathcal{G}, s \models \varphi_1 \text{ and } \mathcal{G}, s \models \varphi_2 \\
\mathcal{G}, s \models \langle\langle A\rangle\rangle\Phi & \text{if there exists a strategy } \sigma_A \in \mathrm{Strat}_{\mathcal{G}}(A), \\
 & \text{such that } \mathcal{G}, \rho \models \Phi \text{ for all } \rho \in \mathrm{Play}_{\mathcal{G}}(s, \sigma_A)
\end{array}$$

The semantics of path formulas is given as follows where φ is a state formula, Φ_1 and Φ_2 are path formulas and $A \subseteq \Pi$ is a coalition

$$\begin{array}{ll}
\mathcal{G}, \rho \models \varphi & \text{if } \mathcal{G}, \rho_0 \models \varphi \\
\mathcal{G}, \rho \models \neg\Phi_1 & \text{if } \mathcal{G}, \rho \not\models \Phi_1 \\
\mathcal{G}, \rho \models \Phi_1 \wedge \Phi_2 & \text{if } \mathcal{G}, \rho \models \Phi_1 \text{ and } \mathcal{G}, \rho \models \Phi_2 \\
\mathcal{G}, \rho \models \mathbf{X}\Phi_1 & \text{if } \mathcal{G}, \rho_{\geq 1} \models \Phi_1 \\
\mathcal{G}, \rho \models \Phi_1 \mathbf{U} \Phi_2 & \text{if } \exists k. \mathcal{G}, \rho_{\geq k} \models \Phi_2 \text{ and } \forall j < k. \mathcal{G}, \rho_{\geq j} \models \Phi_1 \\
\mathcal{G}, \rho \models \Phi_1 \mathbf{R} \Phi_2 & \text{if } \forall k. \mathcal{G}, \rho_{\geq k} \models \Phi_2 \text{ or} \\
 & \exists k. \mathcal{G}, \rho_{\geq k} \models \Phi_1 \text{ and } \forall j \leq k. \mathcal{G}, \rho_{\geq j} \models \Phi_1
\end{array}$$

The branching-time logics CTL*, CTL$^+$ and CTL are the one-player fragments of ATL*, ATL$^+$ and ATL respectively. For an overview of the relationships between the different logics see Figure 2.7.

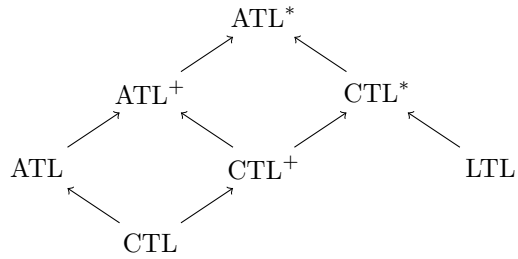


Figure 2.7: Inclusions between different logics. An arrow from $\mathcal{L}_1$ to $\mathcal{L}_2$ means that every $\mathcal{L}_1$ formula is an $\mathcal{L}_2$ formula.

2.3 Automata

We will apply an automata-theoretic approach to verification [VW86]. Here, the main idea is to construct an *automaton* $\mathcal{A}_\varphi$ from a temporal logic formula φ in such a way that the language $\mathcal{L}(\mathcal{A}_\varphi)$ accepted by the automaton is exactly the set of models that make the formula true. As such, reasoning about the automaton corresponds in several ways to reasoning about the formula. For instance, the satisfiability problem for the formula φ asks whether there exists a model for φ. Thus, satisfiability of φ can be reduced to checking emptiness of $\mathcal{L}(\mathcal{A}_\varphi)$.

In this section we introduce several different classes of automata and properties of these known from the litterature. They will be applied as part of our toolbox in later chapters. As the systems we wish to model are reactive systems with infinite duration it is natural that we only consider automata accepting infinite words.

2.3.1 Non-deterministic Büchi automata

The simplest class of automata we introduce is the class of *non-deterministic Büchi automata*. They accept words based on the Büchi condition; a word is accepted if and only if there exists a run of the automaton on the word that passes through an accepting state infinitely many times.

Büchi automata are, for example, used for automated verification via model-checking of LTL properties of finite-state systems [CGP01; BK08].

Definition 2.10 *A non-deterministic Büchi automaton on infinite words (NBA) is a tuple* $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ *where*

- Q *is a finite set of* states
- Σ *is a finite* alphabet
- $q_0 \in Q$ *is the* initial state
- $\delta : Q \times \Sigma \to 2^Q$ *is the* transition function

- $F \subseteq Q$ *is a set of* accepting states.

A run of an NBA $\mathcal{A}$ on an infinite word $w = w_0 w_1 ... \in \Sigma^\omega$ is an infinite sequence $\rho = q_0 q_1 ...$ of states such that for all $i \geq 0$ we have $q_{i+1} \in \delta(q_i, w_i)$. We say that ρ is *accepting* if there exists infinitely many $i \geq 0$ such that $q_i \in F$. We say that $\mathcal{A}$ accepts the word w if there exists an accepting run of $\mathcal{A}$ on the word w. The *language* of $\mathcal{A}$ is the set $\mathcal{L}(\mathcal{A}) \subseteq \Sigma^\omega$ of infinite words which are accepted by $\mathcal{A}$.

It can be shown that the set of languages accepted by NBAs are exactly the ω-regular languages (see e.g. [GTW02]). That is, for a given ω-regular language L there exists an NBA $\mathcal{A}$ such that $\mathcal{L}(\mathcal{A}) = L$ and for every NBA $\mathcal{A}$ the language $\mathcal{L}(\mathcal{A})$ accepted by $\mathcal{A}$ is ω-regular.

In particular, it has been shown that for a given LTL formula φ one can show that there exists an NBA $\mathcal{A}$ which accepts exactly the infinite words that are models for φ. Further, such an automaton can be constructed with $2^{O(|\varphi|)}$ states [WVS83]. By the previous paragraph this means that the set of models for an LTL formula is ω-regular.

2.3.2 Deterministic parity automata

When doing model-checking of games we encounter situations where non-deterministic automata are not suitable due to the alternation between players. Therefore we will need deterministic automata with enough expressiveness for LTL formulas. As deterministic Büchi automata are not capable of this (see e.g. [BK08]) we will apply *deterministic parity automata* which are equally expressive as NBAs (see e.g. [GTW02]).

Definition 2.11 *A deterministic parity automaton on infinite words (DPA) is a tuple* $\mathcal{A} = (Q, \Sigma, q_0, \delta, c)$ *where*

- Q *is a finite set of* states
- Σ *is a finite* alphabet
- $q_0 \in Q$ *is the* initial state
- $\delta : Q \times \Sigma \to Q$ *is the* transition function
- $c : Q \to \mathbb{N}$ *is a* coloring function.

A run of a DPA $\mathcal{A}$ on an infinite word $w = w_0 w_1 ... \in \Sigma^\omega$ is an infinite sequence $\rho = q_0 q_1 ...$ of states such that for all $i \geq 0$ we have $q_{i+1} = \delta(q_i, w_i)$. We say that ρ is *accepting* if the largest color that occurs infinitely often in ρ is even. That is, the largest value e such that $c(q_i) = e$ for infinitely many i must be even.

We say that $\mathcal{A}$ accepts the word w if there exists an accepting run of $\mathcal{A}$ on the word w. The *language* of $\mathcal{A}$ is the set $\mathcal{L}(\mathcal{A}) \subseteq \Sigma^\omega$ of infinite words which are accepted by $\mathcal{A}$.

It is possible to construct a DPA $\mathcal{B}$ from an NBA $\mathcal{A}$ such that the two automata accept the same language. This can be done such that the number of states of $\mathcal{B}$ is in $2^{O(n)}$ and the number of colors are in $O(n)$ where n is the number of states in $\mathcal{A}$ [Pit07]. Thus, from an LTL formula φ one can construct a DPA accepting exactly the models of φ which has $2^{2^{O(|\varphi|)}}$ states and $2^{O(|\varphi|)}$ colors.

2.4 Turing machines

We introduce notation for *deterministic Turing machines* here as we will need reductions from the *halting problem* of such machines in several chapters of this thesis. For a more thorough introduction to Turing machines, see e.g. [HMU03; AB09].

Definition 2.12 *A deterministic Turing machine is a tuple* $\mathcal{T} = (Q, q_0, \Sigma, \delta, q_F)$ *where*

- Q *is a finite set of control states*
- $q_0 \in Q$ *is the initial state*
- Σ *is a finite tape alphabet containing the blank symbol* $\flat \in \Sigma$.
- $\delta : Q \times \Sigma \to Q \times \Sigma \times \{-1, +1\}$ *is the transition function*
- $q_F \in Q$ *is the accepting state*

Note that we use -1 and $+1$ to denote that the tape head moves left and right respectively.

2.5 Two-counter machine

We introduce notation for *deterministic two-counter machines* here as we will need reductions from the halting problem of such machines. This problem is undecidable [Min61]. For a more thorough introduction to counter machines, see e.g. [HMU03].

Definition 2.13 *A two-counter machine is a tuple* $M = (Q, q_0, \delta, q_F)$ *where*

- Q *is a finite set of control states*
- $q_0 \in Q$ *is the initial state*
- $\delta \colon Q \to (Q \times \{c, d\} \times \{+1\}) \cup (Q^2 \times \{c, d\} \times \{-1\})$ *is the transition function*
- $q_F \in Q$ *is the accepting state*

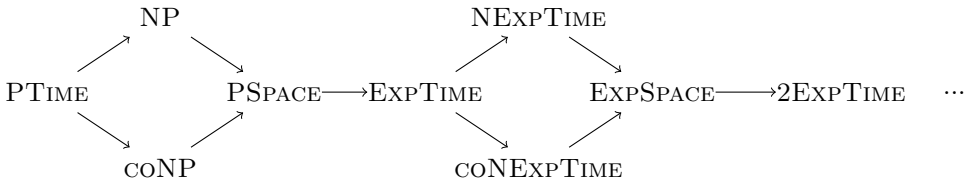


Figure 2.8: Inclusions between complexity classes. An arrow from class $\mathcal{C}_1$ to class $\mathcal{C}_2$ implies that $\mathcal{C}_1 \subseteq \mathcal{C}_2$.

The two counters are called c and d respectively. The transition function δ specifies for each state an increment transition (denoted +1) or a decrement transition (denoted -1). For increment transitions, the counter of the transition is increased and the successor state is chosen according to δ. In the case of decrement transitions there are two possible successor states. The first is chosen if the current counter value is positive and the counter is decreased by one. The other is chosen if the counter value is 0 and the counter value is left unchanged.

2.6 Complexity classes

We will encounter a variety of computational problems with different computational complexities. In fact, a large part of the thesis deals with the classification of the complexity of problems. This both involves providing algorithms for solving problems as well as proving lower bounds. To do this kind of classification we draw upon a number of well-known complexity classes as well as a few more exotic classes. Finally, some problems will also be shown to be undecidable.

We will encounter standard deterministic-time and non-deterministic time complexity classes such as PTime and NP which consist of problems solvable by deterministic polynomial-time Turing machines and non-deterministic polynomial-time Turing machines respectively. In addition, we will see dEXPTIME for $d \geq 1$ and dEXPSPACE for $d \geq 1$ which contain problems solvable using $O(\underbrace{2^{2^{\cdots^{2^{n^k}}}}}_{d})$ time and space respectively where n is the size of the input and k is a constant. In particular, for $d = 1$ we simply denote these classes ExpTime and ExpSpace respectively. We also use co$\mathcal{C}$ to denote the complement of the complexity class $\mathcal{C}$. For instance, coNP is the complement of NP.

Inclusions among these classes are illustrated in Figure 2.8. For a more thorough introduction to these classes, see e.g. [Pap94; AB09]. Below we give a short overview of some of the more non-standard classes used.

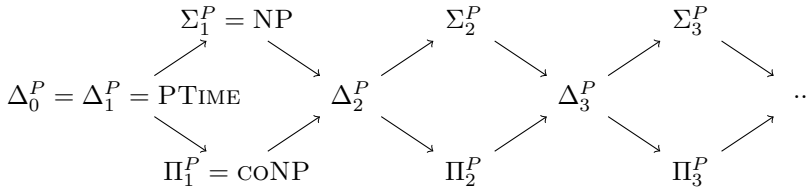


Figure 2.9: Inclusions between complexity classes. An arrow from class $\mathcal{C}_1$ to class $\mathcal{C}_2$ implies that $\mathcal{C}_1 \subseteq \mathcal{C}_2$.

2.6.1 The polynomial hierarchy

The *polynomial hierarchy* consists of an infinite number of complexity classes that are more general than NP and coNP but less general than PSpace . They are based on the notion of *oracle machines* and all have complete problems.

For two decision problems A and B we denote by A^B the class of problems that can be solved by a Turing machine for class A with access to an oracle for a complete problem of class B. For a more detail account, see [Pap94; AB09].

The classes of the polynomial hierarchy are now defined inductively by

$$\Sigma_0^P = \Pi_0^P = \Delta_0^P = \text{PTime}$$

and for every $i \geq 0$

$$\Delta_{i+1}^P = \text{PTime}^{\Sigma_i^P},$$
$$\Sigma_{i+1}^P = \text{NP}^{\Sigma_i^P} \text{ and}$$
$$\Pi_{i+1}^P = \text{coNP}^{\Sigma_i^P}$$

In particular, we have that $\Sigma_1^P = \text{NP}$ and $\Pi_1^P = \text{coNP}$. Relationships between the classes are shown in Figure 2.9.

The canonical PSpace-complete problem is QSat which asks whether a formula of the form

$$Q_1 x_1 Q_2 x_2 ... Q_m x_m . \varphi(x_1, ..., x_m)$$

is satisfiable where $Q_i \in \{\exists, \forall\}$ for all $1 \leq i \leq m$ are quantifiers and $\varphi(x_1, ..., x_m)$ is a Boolean formula over the Boolean variables $x_1, ..., x_m$.

We now define the problem Σ_i^P-Sat for $i \geq 1$ like QSat, but where $Q_1 = \exists$ and there are $i - 1$ alternations between the quantifiers Q_i. Π_i^P-Sat is defined likewise, but with $Q_1 = \forall$. An example of an instance of Σ_3^P-Sat is

$$\exists x_1 \exists x_2 \exists x_3 \forall x_4 \exists x_5 . (x_1 \wedge x_2 \wedge \neg x_5) \vee (x_3 \wedge x_4)$$

Note in particular that Σ_1^P-Sat is the Boolean satisfiability problem which is NP-complete and that Π_1^P-Sat is the Boolean validity problem which is coNP-complete.

More generally, C_i^P-SAT is a complete problem for the complexity class C_i^P where $C \in \{\Sigma, \Pi\}$ and $i \geq 1$.

In Chapter 7 we will encounter problems which are complete for low levels in the polynomial hierarchy.

2.6.2 UP and coUP

The class UP is a subset of NP which contains problems that are recognizable by an unambiguous non-deterministic polynomial-time Turing machine. A non-deterministic polynomial time Turing machine is called unambiguous if for every input it has at most one accepting computation, see e.g. [Pap94; Jur98]. As for other classes, coUP is the complement of the class UP. It is known that PTIME $\subseteq$ UP $\subseteq$ NP, but not whether any of the inclusions are strict. An important problem which is in UP$\cap$coUP (and thus in NP $\cap$ coNP) but not known to be in PTIME is that of solving parity games [Jur98]. We will consider such games in Chapter 4.

Part I

Games with full observation

CHAPTER 3

Introduction and background

Games with *full observation* provide a natural way to model *open systems* [AHK02]. An open system is a system which interacts with its environment. This includes systems such as programs which must react to user input, servers that receive inputs from clients and even hardware circuits which receive inputs and produce corresponding outputs. To model such systems faithfully we need to be able to model the possible behaviors of its environment.

In order to provide guarantees for an open system it is meaningful to require that the system satisfies its specification no matter how the environment behaves. When we model open systems with games this directly corresponds to finding a strategy for the player representing the system which is guaranteed to win no matter how the player representing the environment behaves. Here the winning condition for the system player corresponds directly to the specification of the system.

For the questions considered we are very pessimistic about the behavior of the environment which is assumed to be antagonistic. Intuitively, we imagine that the environment tries to sabotage our system and therefore want to design our system such that the antagonistic environment cannot possible succeed. In fact, this is implicitly assumed for most of the problems we consider in Part I.

In this part it is also assumed that both the system and the environment have full observation of the current state of the world. This assumption restricts us with respect to the systems that we can reasonably model. However, the lack of generality gives us a much lower complexity for the problems considered. As we shall see in Part II it will be more difficult to find decidable problems and problems with a tractable complexity in the partial information case.

An example of an open system where it is reasonable to assume that both the system and the environment have full observation is a hardware circuit. In a synchronous fashion such a circuit receives a fixed number of input signals and must produce a fixed number of output signals in each step. Such steps are then repeated indefinitely. It seems reasonable to assume that the circuit observes all the inputs it receives and that it knows exactly which outputs it has produced. On the other hand, as we are being pessimistic with respect to the environment it is also reasonable to assume that the environment has observed both the sequence of inputs and outputs that has occured.

However, there are also cases which are not meaningful to model in this way. As an example, consider a system for controlling the temperature within a house. This system has a number of sensors in different rooms and has the ability to alter the temperature in different rooms by increasing or decreasing the amount of power spent in different heating devices. This system will receive some information about the current state of the distribution of heat throughout the house, but cannot observe it fully due to imprecision of sensors. As such, it can only act based on partial information. Models to capture partial information systems like this is the topic of Part II.

The purpose of this part of the thesis is to

- investigate the computational complexity and
- design techniques for improving the efficiency of decision procedures

for problems concerning winning strategies in various classes of full observation games.

One of the most notable classes of games we investigate are parity games which have many applications in verification, for instance they are closely related to model-checking of the modal μ-calculus. We also consider one-counter games which are some of the simplest infinite-state games which are useful for modelling systems with an unbounded resource. Whereas our results on parity games and one-counter games are quite specific to these classes of games, some of the other results we develop consider more general classes of infinite-state games.

Specifically, in Chapter 4 we develop the concept of winning cores in parity games and apply this concept to design a new polynomial-time approximation algorithm for solving parity games efficiently. In Chapter 5 we investigate the complexity of model-checking alternating-time temporal logics and branching-time temporal logics in one-counter games as well as deciding the winner in such games with LTL objectives. In Chapter 6 we extend the symmetry reduction technique known from transition systems, probabilistic systems and real-time systems to be able to handle finitely branching turn-based games. In Chapter 7 we consider the complexity of the satisfiability problem for a number of flat fragments of alternating-time temporal logics and branching-time temporal logics.

The chapters in this part are ordered by the generality of the games considered, where Chapter 4 considers finite-state turn-based games, Chapter 5 we consider one-counter games which is a special class of infinite-state turn-based games. Chapter 6 deals with (possibly infinite-state) turn-based games with finite branching and in Chapter 7 we handle concurrent games.

CHAPTER 4

Winning cores in parity games

This chapter is an adapted version of the paper

- [Ves16] Steen Vester. "Winning Cores in Parity Games". To appear in *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. 2016

It has been updated to be compatible with the other chapters of the thesis. Further, some of the experimental results are presented in more detail in this chapter.

4.1 Introduction

Solving *parity games* [EJ91] is an important problem of both theoretical and practical interest. It is known to be in NP∩coNP [EJS01] and UP∩coUP [Jur98] but in spite of the development of many different algorithms (see e.g. [Zie98; Jur00; VJ00; JPZ06; Sch07]), frameworks for benchmarking such algorithms [FL09; Kei14] and families of parity games designed to expose the worst-case behaviour of existing algorithms [Jur00; Fri09; Fri11] it has remained an open problem whether a polynomial-time algorithm exists.

Various problems for which polynomial-time algorithms are not known can been reduced in polynomial time to the problem of solving parity games. Among these are model-checking of the propositional μ-calculus [Koz83; EL86; Sti95], the emptiness problem for parity automata on infinite binary trees [Mos84; EJS01] and solving Boolean equation systems [Mad97]. For relations to other problems in logic and automata theory, see e.g. [GTW02].

Some of the most notable algorithms from the litterature of solving parity games include Zielonka's algorithm [Zie98] using $O(n^d)$ time, the small progress measures algorithm [Jur00] using $O(d \cdot m \cdot (n/d)^{d/2})$ time, the strategy improvement algorithm [VJ00] using $O(n \cdot m \cdot 2^m)$ time, the big step algorithm [Sch07] using $O(m \cdot n^{d/3})$ time and the dominion decomposition algorithm [JPZ06] using $O(n^{\sqrt{n}})$ time. Here, n is the number of states in the game, m is the number of transitions and d is the maximal color of the game.

Contribution

The main contributions of this chapter are to introduce the novel concept of *winning cores* in parity games and develop a fast deterministic polynomial-time under-approximation algorithm for solving parity games based on properties of winning cores. Two different, but equivalent, definitions of winning cores are given both of which are used to show a number of interesting properties. One is based on the new notion of *consecutive dominating sequences*.

We investigate winning cores and show that the winning core of a player is always a subset of the winning region of the player and more importantly that the winning core of a player is empty if and only if the winning region of the player is empty. A result of [DKT12] then implies that emptiness of the winning core of a player can be decided in polynomial time if and only if parity games can be solved in polynomial time. We further show that the winning cores for the two players contain all fatal attractors [HKP13; HKP14] and show properties of winning cores which are similar in nature to the properties of winning regions that form the basis of the recursive algorithms in [Zie98; JPZ06; Sch07] for solving parity games.

We also show that winning cores are not necessarily dominions [JPZ06] which is interesting on its own. To the knowledge of the author no meaningful subsets of the winning regions have been characterized in the litterature which were not dominions. Several of the existing algorithms for solving parity games are based on finding dominions, e.g. [Zie98; JPZ06; Sch07]. However, it was recently shown in [Gaj+15] that there is no algorithm which decides if there exists a dominion with at most k states in time $n^{o(\sqrt{k})}$ unless the exponential-time hypothesis fails. Thus, going beyond dominions could very well be important in the search for a polynomial-time algorithm for solving parity games. Winning cores provide a viable direction for this search.

Next, we show the existence of memoryless optimal strategies for games with a certain type of prefix-dependent objectives using a result of [GZ05]. Based on this we provide a decreasing sequence of sets of states which converges to the winning core in at most n steps. It is also shown that winning cores can be computed in polynomial time if and only if parity games can be solved in polynomial time and that winning core computation is in $\mathrm{UP} \cap \mathrm{coUP}$ by a reduction to solving parity games.

The correctness of our under-approximation algorithm relies on fast convergence of the sequence mentioned above. It uses $O(d \cdot n^2 \cdot (n+m))$ time and $O(d+n+m)$ space. It is an under-approximation algorithm in the sense that it returns subsets of the winning regions for the two players.

The algorithm has been implemented in OCaml on top of the PgSolver framework [FL09] and experiments have been carried out both to test the quality of the approximations as well as the practical running times. The experimental results are very positive as it solved all games from the benchmark set of PgSolver completely and solved a very high ratio of randomly generated games completely. Further, on most of the benchmark games it outperformed the existing state-of-the-art algorithms significantly and solved games with more than 10^7 states. The algorithm also per-

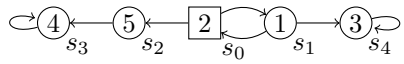


Figure 4.1: Example of a parity game.

formed very well compared to the best existing partial solver for parity games [HKP13; HKP14] both with respect to quality of approximation and running time.

Outline

Section 4.2 contains preliminary definitions and Section 4.3 introduces consecutive dominating sequences. In Section 4.4 winning cores are introduced and a number of properties about them are presented. In Chaper 4.5 the computational complexity of computing winning cores is analyzed. In Section 4.6 the approximation algorithm is presented and Section 4.7 contains experimental results. Finally, Section 4.8 contains a summary.

4.2 Preliminaries

A *parity game* [EJ91] is a two-player finite-state turn-based game where each state is labelled with a natural number. This number is called the *color* of the state. Player 0 wants the greatest color occuring infinitely often during the play to be even whereas player 1 wants it to be odd. The transition relation R is assumed to be *left-total*, i.e. for every state s in the game there exists a state t such that $(s,t) \in R$. This implies that every play is infinite.

Formally, we define parity games as follows.

Definition 4.1 *A parity game is a tuple $\mathcal{G} = (S, S_0, S_1, R, c)$ such that*

- $(S, \{0,1\}, (S_0, S_1), R)$ *is a finite-state* turn-based game *with R being left-total*
- $c : S \mapsto \{1, ..., d\}$ *is a* coloring function *specifying a color for each state*

Example 4.2 *A simple example of a parity game can be seen in Figure 4.1. Circle states are in S_0 and square states in S_1. The values drawn inside states are colors. There is an arrow from state s to state t if $(s,t) \in R$.*

For the rest of this section as well as Sections 4.3 and 4.4 we fix a parity game $\mathcal{G} = (S, S_0, S_1, R, c)$ with colors in $\{1, ..., d\}$ and n states.

Concepts such as plays, histories and strategies in $\mathcal{G}$ are defined as for the underlying turn-based game $(S, \{0,1\}, (S_0, S_1), R)$.

For a path $\rho = s_0 s_1 ...$ in $\mathcal{G}$ we define the associated sequence $c(\rho) = c(s_0)c(s_1)...$ of colors and for a set P of paths define $c(P) = \{c(\rho) \mid \rho \in P\}$. For a sequence

$\pi = e_0 e_1 ...$ let

$$\inf(\pi) = \{e \mid \text{there exists infinitely many } i \text{ s.t. } e = e_i\}$$

We define the *parity condition* Ξ_j for player $j \in \{0, 1\}$ by

$$\Xi_j = \{\pi \in \mathbb{N}^\omega \mid \exists k. \forall i \geq 0. \pi_i \leq k \wedge \max(\inf(\pi)) \equiv j \pmod 2\}$$

where $\mathbb{N}$ is the set of non-negative integers. This is the set of infinite sequences of colors which are winning for player j in a parity game. That is, the infinite sequences of colors where the largest color that occurs infinitely often is equivalent to j modulo 2. Further, let

$$\Lambda_j = \{\pi \in \Xi_j \mid \max_{i>0} \pi_i \equiv j \pmod 2\}$$

be the subset of Ξ_j where it is also required that the largest non-initial color occuring in a sequence is equivalent to j modulo 2. Note that the initial element of the sequence is not counted.

We say that a strategy σ_j for player j is a *winning strategy* for player j from state s_0 if $c(\text{Play}(\mathcal{G}, s_0, \sigma_j)) \subseteq \Xi_j$. When such a strategy exists we call s_0 a *winning state* for player j. We write $W_j(\mathcal{G})$ for the set of winning states of player j in $\mathcal{G}$. This is also called the *winning region* for player j. Since parity games are memoryless determined [EM79; EJ91] we have $W_0(\mathcal{G}) \cup W_1(\mathcal{G}) = S$ and $W_0(\mathcal{G}) \cap W_1(\mathcal{G}) = \emptyset$. Further, there is a memoryless strategy for player j that is winning from every $s \in W_j(\mathcal{G})$.

4.2.1 Restricted parity games

We define the *restricted parity game* $\mathcal{G} \restriction S' = (S', S'_0, S'_1, R', c')$ for a subset $S' \subseteq S$ by

- $S'_j = S' \cap S_j$ for $j \in \{0, 1\}$
- $R' = R \cap (S' \times S')$
- $c'(s) = c(s)$ for every $s \in S'$

Intuitively, the restricted parity game $\mathcal{G} \restriction S'$ is the same as $\mathcal{G}$ where all states not in S' are removed and all transitions (s, s') with either s or s' not in S' are removed. Note that the restricted parity game is only a well-defined parity game when R' is left-total.

4.2.2 Attractor sets

The notion of an *attractor set* is well-known [Zie98] and is the set of states from which a player j can ensure reaching a set of target states.

Definition 4.3 *The attractor set* $\mathrm{Attr}_j(\mathcal{G}, T)$ *for a target set* $T \subseteq S$ *and a player* j *is the limit of the sequence* $\mathrm{Attr}_j^i(\mathcal{G}, T)$ *where*

$$\begin{aligned}
\mathrm{Attr}_j^0(\mathcal{G}, T) &= T \\
\mathrm{Attr}_j^{i+1}(\mathcal{G}, T) &= \mathrm{Attr}_j^i(\mathcal{G}, T) \\
&\cup \{s \in S_j \mid \exists t.(s,t) \in R \wedge t \in \mathrm{Attr}_j^i(\mathcal{G}, T)\} \\
&\cup \{s \in S_{1-j} \mid \forall t.(s,t) \in R \Rightarrow t \in \mathrm{Attr}_j^i(\mathcal{G}, T)\}
\end{aligned}$$

The attractor set and a memoryless strategy to ensure reaching the target from this set can be computed in time $O(n+m)$ in a game with n states and m transitions [AHK98]. The *positive attractor* $\mathrm{Attr}_j^+(\mathcal{G}, T)$ is the set of states from which player j can ensure reaching T in at least 1 step. Formally,

$$\begin{aligned}
\mathrm{Attr}_j^+(\mathcal{G}, T) &= \mathrm{Attr}_j(\mathcal{G}, \\
&\{s \in S_j \mid \exists t \in T.(s,t) \in R\} \\
&\cup \{s \in S_{1-j} \mid \forall t \in S.(s,t) \in R \Rightarrow t \in T\})
\end{aligned}$$

4.2.3 j-closed sets and dominions

A subset $S' \subseteq S$ of states in a parity game is called *j-closed* if

1. For every $s \in S_{1-j} \cap S'$ there exists no $t \in S \setminus S'$ such that $(s,t) \in R$
2. For every $s \in S_j \cap S'$ there exists $t \in S'$ such that $(s,t) \in R$

Thus, a set of states is j-closed if and only if player j can force the play to stay in this set of states.

A *j-dominion* [JPZ06] for player j is a set $T \subseteq S$ such that from every state $s \in T$ player j has a strategy σ such that $c(\mathrm{Play}(\mathcal{G}, s, \sigma)) \subseteq \Xi_j$ and $\mathrm{Play}(\mathcal{G}, s, \sigma) \subseteq T^\omega$. That is, from every state in a j-dominion, player j can ensure to win while keeping the play inside the j-dominion. Thus, a j-dominion is j-closed.

Proposition 4.4 ([JPZ06]) *$W_j(\mathcal{G})$ is a j-dominion.*

Proposition 4.5 ([JPZ06]) *Let $V \subseteq W_j(\mathcal{G})$, $V' = \mathrm{Attr}_j(\mathcal{G}, V)$ and $\mathcal{G}' = \mathcal{G} \restriction (S \setminus V')$. Then $W_j(\mathcal{G}) = V' \cup W_j(\mathcal{G}')$ and $W_{1-j}(\mathcal{G}) = W_{1-j}(\mathcal{G}')$.*

In Figure 4.2 we illustrate Proposition 4.5. As player j is winning in V he is also winning from his attractor set V' of V as he can force the play to V and ensure winning from there. Then, the remaining winning states for the two players can be found by looking at the remaining game $\mathcal{G}' = \mathcal{G} \restriction (S \setminus V')$.

Many of the existing algorithms for solving parity games work by finding a dominion D for some player j and then apply Proposition 4.5 to remove the states in $\mathrm{Attr}_j(\mathcal{G}, D)$ and recursively solve the smaller resulting game. This includes Zielonka's algorithm [Zie98], the dominion decomposition algorithm [JPZ06] and the big step algorithm [Sch07]. The algorithm we present in this chapter also applies this proposition, but the winning cores which we search for are not necessarily dominions.

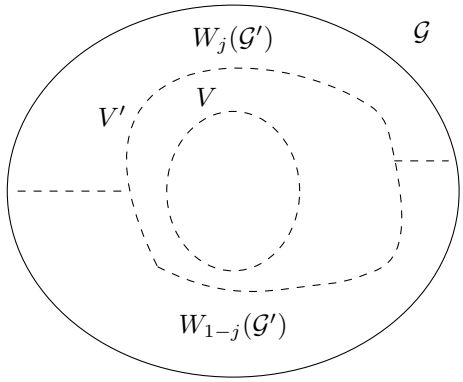


Figure 4.2: Illustration of Proposition 4.5.

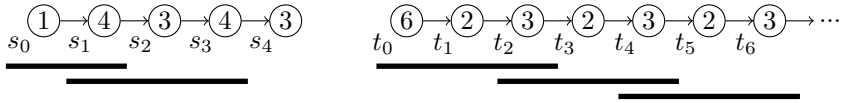


Figure 4.3: Consecutive j-dominating sequences illustrated by bold lines. Note the overlap of one state between sequences.

4.3 Dominating Sequences

We say that a path $\rho = s_0 s_1 ...$ with at least one transition is *0-dominating* if the color $e = \max\{c(s_i) \mid i > 0\}$ is even and *1-dominating* if it is odd. Note that we do not include the color of the first state of the sequence.

We say that a path ρ begins with k *consecutive j-dominating sequences* if there exist indices $i_0 < i_1 < ... < i_k$ with $i_0 = 0$ such that $\rho_{i_\ell}\rho_{i_\ell+1}...\rho_{i_{\ell+1}}$ is j-dominating for all $0 \leq \ell < k$. Similarly, a play ρ begins with an infinite number of consecutive j-dominating sequences if there exists an infinite sequence $i_0 < i_1 < ...$ of indices with $i_0 = 0$ such that $\rho_{i_\ell}\rho_{i_\ell+1}...\rho_{i_{\ell+1}}$ is j-dominating for all $\ell \geq 0$.

As examples, the sequence on the left in Figure 4.3 begins with two consecutive 0-dominating sequences $s_0 s_1$ and $s_1 s_2 s_3$ whereas the sequence to the right begins with only one 0-dominating sequence $t_0 t_1$, but not two consecutive 0-dominating sequences. Also, the sequence to the left does not begin with a 1-dominating sequence whereas the sequence to the right begins with an infinite number of consecutive 1-dominating sequences: $t_0 t_1 t_2$, $t_2 t_3 t_4$, $t_4 t_5 t_6$ etc.

We start with the following well-known lemma, stating that the winner of a play ρ in a parity game is independent of a given finite prefix of the play.

Lemma 4.6 *Let ρ be a play and ρ' be a suffix of ρ. Then $c(\rho) \in \Xi_j$ if and only if $c(\rho') \in \Xi_j$.*

The following proposition shows that a play is winning for player j if and only if it has a suffix that begins with an infinite number of consecutive j-dominating sequences.

Proposition 4.7 *Let ρ be a play. Then $c(\rho) \in \Xi_j$ if and only if there is a suffix of ρ that begins with an infinite number of consecutive j-dominating sequences.*

Proof. ($\Rightarrow$) Let ρ be a play such that $c(\rho) \in \Xi_j$. Let $e \equiv j \pmod 2$ be the greatest color occuring infinitely often in ρ. Since there are only a finite number of different colors, there can only be a finite number of indices i such that $c(\rho_i) > e$. Let ℓ be the largest such index. Further, let $\ell < i_0 < i_1 < ...$ be an infinite sequence of indices such that $c(\rho_{i_k}) = e$ for all $k \geq 0$. Such a sequence exists since e occurs infinitely often in ρ. Now, $\rho_{\geq i_0}$ begins with an infinite number of consecutive j-dominating sequences, namely the sequences $\pi_k = \rho_{i_k}\rho_{i_k+1}...\rho_{i_{k+1}}$ for $k \geq 0$.

($\Leftarrow$) Let ρ be a play with a suffix $\rho_{\geq i_0}$ that begins with an infinite number of consecutive j-dominating sequences. Let $i_0 < i_1 < ...$ be an infinite sequence of indices such that $\pi_\ell = \rho_{i_\ell}\rho_{i_\ell+1}...\rho_{i_{\ell+1}}$ is j-dominating for every $\ell \geq 0$.

Now, suppose for contradiction that $c(\rho) \in \Xi_{1-j}$. Then the greatest color e that occurs infinitely often in ρ satisfies $e \equiv 1 - j \pmod 2$. Now, e is the color of a non-initial state in π_ℓ for an infinite number of indices ℓ. Since every such π_ℓ is j-dominating there is an infinite number of states in ρ with a color $e' > e$ such that $e' \equiv j \pmod 2$. Since there are only finitely many different colors, there exists a particular color $e'' > e$ which is the color of infinitely many states in ρ. This gives a contradiction since e was chosen as the greatest color that occurs infinitely often in ρ. This implies that $c(\rho) \in \Xi_j$. □

Next, we show a slightly surprising fact. A play begins with an infinite number of consecutive j-dominating sequences if and only if it is both winning for player j and j-dominating. This means that we have two quite different characterizations of the same concept. Both will be used to obtain results later.

Proposition 4.8 *Let ρ be a play. Then ρ begins with an infinite number of consecutive j-dominating sequences if and only if ρ is j-dominating and $c(\rho) \in \Xi_j$.*

Proof. ($\Leftarrow$) Suppose that ρ is j-dominating and $c(\rho) \in \Xi_j$. By Proposition 4.7 there exists a suffix $\rho_{\geq \ell}$ of ρ that begins with an infinite number of consecutive j-dominating sequences $\rho^0, \rho^1, ...$ Let $i_0 < i_1 < ...$ be the indices such that $\rho^\ell = \rho_{i_\ell}\rho_{i_\ell+1}...\rho_{i_{\ell+1}}$. Since ρ is j-dominating the greatest color e of a non-initial state in ρ satisfies $e \equiv j \pmod 2$. Let $k \geq 0$ be the smallest index such that $\max\{c(\rho_i) \mid 0 < i \leq i_{k+1}\} = e$. Now we have that ρ begins with an infinite number of consecutive j-dominating sequences, namely $\rho_{\leq i_{k+1}}, \rho^{k+1}, \rho^{k+2}....$

($\Rightarrow$) By Proposition 4.7 we also have that if ρ begins with an infinite number of consecutive j-dominating sequences $\rho^0, \rho^1, ...$ then $c(\rho) \in \Xi$. Now, suppose for contradiction that ρ is not j-dominating. Then the largest color e of a non-initial

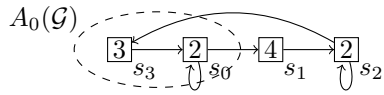


Figure 4.4: A parity game where $W_0(\mathcal{G}) = \{s_0, s_1, s_2, s_3\}$ and $A_0(\mathcal{G}) = \{s_0, s_3\}$.

state in ρ satisfies $e \equiv 1 - j \pmod 2$. Let $i > 0$ be an index such that $c(\rho_i) = e$. Let k be an index such that ρ^k contains ρ_i as a non-initial state. As ρ^k is j-dominating there is a non-initial state in ρ^k with a color $e' > e$ which gives a contradiction. □

As the two characterizations are equivalent, we will just write $c(\rho) \in \Lambda_j$ for the remainder of the chapter when we know that either property is true of ρ.

4.4 Winning cores

We define *the winning core* $A_j(\mathcal{G})$ for player j in the parity game $\mathcal{G}$ as the set of states s from which player j has a strategy σ such that $c(\text{Play}(\mathcal{G}, s, \sigma)) \subseteq \Lambda_j$. According to Proposition 4.8 we have two different characterizations of this set of plays. Both will be used in the following depending on the application.

As an example, consider the game illustrated in Figure 4.4. Here player 0 is winning from all states. The winning core is given by $A_0(\mathcal{G}) = \{s_0, s_3\}$. Indeed, from states s_1 and s_2 player 1 can force the play to be 1-dominating.

4.4.1 The winning core and the winning region

First note that since $\Lambda_j \subseteq \Xi_j$ we have that every state in the winning core for player j is a winning state for player j.

Proposition 4.9 *Let $\mathcal{G}$ be a parity game. Then $A_j(\mathcal{G}) \subseteq W_j(\mathcal{G})$.*

Next, we will show a more surprising fact: If the winning core for player j is empty, then the winning region of player j is empty as well. This is a very important property of winning cores.

Proposition 4.10 *Let $\mathcal{G}$ be a parity game. If $A_j(\mathcal{G}) = \emptyset$ then $W_j(\mathcal{G}) = \emptyset$.*

Proof. Let $A_j(\mathcal{G}) = \emptyset$. Suppose for contradiction that $W_j(\mathcal{G}) \neq \emptyset$. Then there exists $s \in W_j(\mathcal{G})$ and a memoryless winning strategy σ for player j from s.

Since $s \notin A_j(\mathcal{G})$ there exists $\rho \in \text{Play}(\mathcal{G}, s, \sigma)$ that does not begin with an infinite number of consecutive j-dominating sequences. However, since $c(\rho) \in \Xi_j$ there is a suffix of ρ that begins with an infinite number of consecutive j-dominating sequences. Let $\ell_0 > 0$ be the smallest index such that $\rho_{\geq \ell_0}$ begins with an infinite number of consecutive j-dominating sequences. Then $\rho_{\leq \ell_0}$ is $(1-j)$-dominating, because otherwise ρ would begin with an infinite number of consecutive j-dominating sequences.

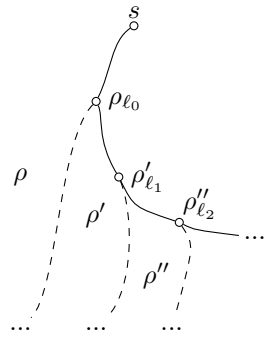


Figure 4.5: Construction of a play $\pi \in \text{Play}(\mathcal{G}, s, \sigma)$ that begins with an infinite number of consecutive $(1-j)$-dominating sequences. π is the solid path in the figure.

Since $A_j(\mathcal{G}) = \emptyset$ there exists $\rho' \in \text{Play}(\mathcal{G}, \rho_{\ell_0}, \sigma)$ that does not begin with an infinite number of consecutive j-dominating sequences. Since σ is memoryless we have $\rho_{<\ell_0} \cdot \rho' \in \text{Play}(\mathcal{G}, s, \sigma)$ which means that $c(\rho') \in \Xi_j$ according to Lemma 4.6. Here, $\cdot$ is the concatenation operator. This implies that there is a suffix of ρ' that begins with an infinite number of consecutive j-dominating sequences. Let $\ell_1 > 0$ be the smallest index such that $\rho'_{\geq \ell_1}$ begins with an infinite number of consecutive j-dominating sequences. Then $\rho'_{\leq \ell_1}$ is $(1-j)$-dominating, because otherwise ρ' would begin with an infinite number of consecutive j-dominating sequences.

Since $A_j(\mathcal{G}) = \emptyset$ there exists $\rho'' \in \text{Play}(\mathcal{G}, \rho'_{\ell_1}, \sigma)$ that does not begin with an infinite number of consecutive j-dominating sequences. Since σ is memoryless we have that $\rho_{<\ell_0} \cdot \rho'_{<\ell_1} \cdot \rho'' \in \text{Play}(\mathcal{G}, s, \sigma)$ which means that $c(\rho'') \in \Xi_j$ according to Lemma 4.6. We can continue this construction in the same way to obtain the play $\pi = \rho_{<\ell_0} \cdot \rho'_{<\ell_1} \cdot \rho''_{<\ell_2} \cdot ...$ which belongs to $\text{Play}(\mathcal{G}, s, \sigma)$. The construction is illustrated in Figure 4.5.

Observe that π begins with an infinite number of consecutive $(1-j)$-dominating sequences, namely $\rho_{\leq \ell_0}, \rho'_{\leq \ell_1}, \rho''_{\leq \ell_2}, ...$ which are all $(1-j)$-dominating. By Proposition 4.7 we have $c(\pi) \in \Xi_{1-j}$. This is a contradiction since $\pi \in \text{Play}(\mathcal{G}, s, \sigma)$ and $c(\pi) \in \Xi_j$. Thus, $W_j(\mathcal{G}) = \emptyset$. □

Proposition 4.9 and 4.10 give us the following result.

Theorem 4.11 *Let $\mathcal{G}$ be a parity game. The winning core $A_j(\mathcal{G})$ for player j in $\mathcal{G}$ is empty if and only if the winning region $W_j(\mathcal{G})$ for player j in $\mathcal{G}$ is empty.*

Remark 4.12 *As shown in [DKT12] parity games can be solved in polynomial time if and only if it can be decided in polynomial time whether the winning region $W_j(\mathcal{G}) = \emptyset$ for player j. Thus, Theorem 4.11 implies that parity games can be solved in polynomial*

time if and only if emptiness of the winning core for player j can be decided in polynomial time.

In [HKP13] the concept of a *fatal attractor* is defined and used for partially solving parity games. A fatal attractor is a set X of states colored $e \equiv j \pmod 2$ with the property that player j can ensure that when the play begins in a state in X then it will eventually reach X again without having passed through any states with color greater then e along the way. Player j can thus force the play to begin with an infinite number of consecutive j-dominating sequences from states in X by repeatedly forcing the play back to X in this fashion.

Proposition 4.13 *Let X be a fatal attractor for player j in $\mathcal{G}$. Then $X \subseteq A_j(\mathcal{G})$.*

Note that the winning core $A_j(\mathcal{G})$ for player j need not be a j-dominion. In addition, neither does $\mathrm{Attr}_j(\mathcal{G}, A_j(\mathcal{G}))$. Indeed, consider again the parity game in Figure 4.4. In this game, the winning core for player 0 is $A_0(\mathcal{G}) = \{s_0, s_3\}$ and also $\mathrm{Attr}_0(\mathcal{G}, A_0(\mathcal{G})) = A_0(\mathcal{G})$. Clearly, this is not a 0-dominion as player 1 can force the play to go outside this set. Note also that this game has no fatal attractors. Thus, the winning core can contain more states than just the fatal attractors.

The property that the winning core for player j is not necessarily a j-dominion is interesting as, to the knowledge of the author, no meaningful subsets of the winning region for player j that are not necessarily j-dominions have been characterized in the litterature. Thus, many algorithms focus on looking for dominions which can be removed from the game, e.g. the algorithms from [Zie98; JPZ06; Sch07]. However, it was recently shown in [Gaj+15] that there is no algorithm which decides if there exists a dominion with at most k states in time $n^{o(\sqrt{k})}$ unless the exponential-time hypothesis fails. This, along with Theorem 4.11 make winning cores very interesting objects for further study as they propose a fresh direction of research in solving parity games.

In the remainder of this subsection we show some more interesting properties about winning cores which are similar in nature to the results on winning regions in parity games from [Zie98] that form the basis of Zielonka's algorithm for solving parity games as well as optimized versions in [JPZ06; Sch07]. To do this we first need a lemma.

Lemma 4.14 *Let $T \subseteq S$ be j-closed and σ' be a strategy for player j in $\mathcal{G} \restriction T$. Then there exists a strategy σ for player j in $\mathcal{G}$ such that*

$$\mathrm{Play}(\mathcal{G}, s, \sigma) = \mathrm{Play}(\mathcal{G} \restriction T, s, \sigma')$$

for all $s \in T$. Moreover, if σ' is memoryless then σ can be chosen to be memoryless as well.

Proof. Define σ by $\sigma(h) = \sigma'(h)$ for every history h that only contains states in T and arbitrarily for all other histories.

First we have that every play $\rho = s_0 s_1 ... \in \text{Play}(\mathcal{G} \restriction T, s, \sigma')$ where $s \in T$ belongs to $\text{Play}(\mathcal{G}, s, \sigma)$ as well since $\sigma(s_0...s_\ell) = \sigma'(s_0...s_\ell) = s_{\ell+1}$ for every prefix $s_0...s_\ell$ of ρ such that $s_\ell \in S_j$ and $(s_\ell, s_{\ell+1}) \in R$ whenever $s_\ell \in S_{1-j}$.

On the other hand, for every play $\rho = s_0 s_1 ... \in \text{Play}(\mathcal{G}, s, \sigma)$ where $s \in T$ we have that $s_i \in T$ for every $i \geq 0$. This can be shown by induction as follows. For the base case we have that $s_0 \in T$. For the induction step we have that whenever $s_i \in S_{1-j} \cap T$ then there exists no $t \in S \setminus T$ with $(s_i, t) \in R$ since T is j-closed. Further, whenever $s_i \in S_j \cap T$ and $s_0, ..., s_i \in T$ then $\sigma(s_0...s_i) = \sigma'(s_0.., s_i) \in T$. This also implies that $\rho \in \text{Play}(\mathcal{G} \restriction T, s, \sigma')$. Thus, $\text{Play}(\mathcal{G}, s, \sigma) = \text{Play}(\mathcal{G} \restriction T, s, \sigma')$ for $s \in T$.

Note that if σ' is memoryless then σ can be chosen to be memoryless as well. □

Corollary 4.15 *If $T \subseteq S$ is j-closed in $\mathcal{G}$ then $W_j(\mathcal{G} \restriction T) \subseteq W_j(\mathcal{G})$.*

Now, let $\mathcal{G} = (S, S_0, S_1, R, c)$ be a parity game with largest color d. Let k be the player such that $d \equiv k \pmod 2$. Let $S^{=d}$ be the set of states with color d and $U = \text{Attr}_k^+(\mathcal{G}, S^{=d})$. Let $\mathcal{G}' = \mathcal{G} \restriction (S \setminus U)$. We use S' to denote the set of states of $\mathcal{G}'$.

Proposition 4.16 $A_{1-k}(\mathcal{G}) = A_{1-k}(\mathcal{G}')$

Proof. Suppose first that $s \in A_{1-k}(\mathcal{G}')$. Then there exists a strategy σ' for player $1-k$ in $\mathcal{G}'$ such that every play $\rho \in \text{Play}(\mathcal{G}', s, \sigma')$ begins with an infinite number of consecutive $(1-k)$-dominating sequences. Let σ be a strategy in $\mathcal{G}$ for player $1-k$ defined by $\sigma(h) = \sigma'(h)$ for histories h that only contain states from S'. Let σ be defined arbitrarily for all other histories. We now have that $\text{Play}(\mathcal{G}, s, \sigma) = \text{Play}(\mathcal{G}', s, \sigma')$ as player k does not control any state in S' with a transition to U and σ only prescribes taking transitions that make the play stay in S' if no state outside S' is reached. This implies that $s \in A_{1-k}(\mathcal{G})$.

Suppose on the other hand that $s \in A_{1-k}(\mathcal{G})$. Then there exists a strategy σ for player $1-k$ such that every play $\rho \in \text{Play}(\mathcal{G}, s, \sigma')$ begins with an infinite number of consecutive $(1-k)$-dominating sequences. As $d \equiv k \pmod 2$ it follows from Proposition 4.8 that no state in a play $\rho \in \text{Play}(\mathcal{G}, s, \sigma')$ is contained in U. Thus, we can define a strategy σ' in $\mathcal{G}'$ by $\sigma'(h) = \sigma(h)$ for every history with initial state s and obtain $\text{Play}(\mathcal{G}, s, \sigma) = \text{Play}(\mathcal{G}', s, \sigma')$. This implies that $s \in A_k(\mathcal{G}')$. □

We define $\mathcal{G}'' = \mathcal{G} \restriction (S \setminus \text{Attr}_{1-k}(\mathcal{G}, A_{1-k}(\mathcal{G}))$ as the parity game obtained from $\mathcal{G}$ by removing the set of states from which player $1-k$ can force the play to go to his winning core in $\mathcal{G}$. We use S'' to denote the set of states of $\mathcal{G}''$.

Proposition 4.17 $A_k(\mathcal{G}) = A_k(\mathcal{G}'')$

Proof. Suppose first that $s \in A_k(\mathcal{G}'')$. Then there exists a strategy σ'' for player k in $\mathcal{G}''$ such that every play $\rho \in \text{Play}(\mathcal{G}'', s, \sigma'')$ begins with an infinite number of consecutive k-dominating sequences. Let σ be a strategy in $\mathcal{G}$ for player k defined

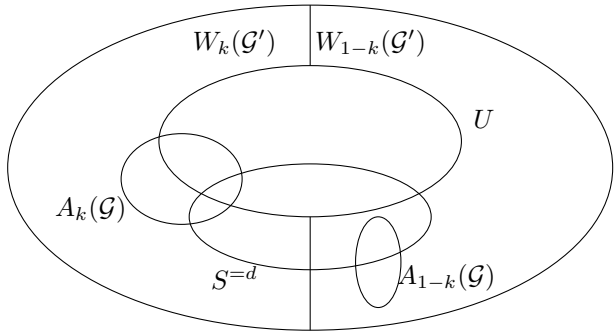


Figure 4.6: Illustration of winning cores and winning regions in $\mathcal{G}$.

by $\sigma(h) = \sigma''(h)$ for histories h that only contain states from S''. Let σ be defined arbitrarily for all other histories. We now have that $\text{Play}(\mathcal{G}, s, \sigma) = \text{Play}(\mathcal{G}'', s, \sigma'')$ as player $1-k$ does not control any state in S'' with a transition to $S \setminus S''$ and σ only prescribes taking transitions that make the play stay in S'' if no state outside S'' is reached. This implies that $s \in A_k(\mathcal{G})$.

On the other hand suppose that $s \in A_k(\mathcal{G})$. Then there exists a strategy σ for player k in $\mathcal{G}$ such that every play $\rho \in \text{Play}(\mathcal{G}, s, \sigma)$ begins with an infinite number of consecutive k-dominating sequences. Note that no such play has a state contained in $\text{Attr}_{1-k}(\mathcal{G}, A_{1-k}(\mathcal{G}')) = \text{Attr}_{1-k}(\mathcal{G}, A_{1-k}(\mathcal{G}))$ because all such states are winning for player $1-k$. Therefore, it is possible to define a strategy σ'' in $\mathcal{G}''$ for player k by $\sigma''(h) = \sigma(h)$ for every history h in $\mathcal{G}''$ with initial state s. Further, we get $\text{Play}(\mathcal{G}, s, \sigma) = \text{Play}(\mathcal{G}'', s, \sigma'')$ which implies $s \in A_k(\mathcal{G}'')$. □

The situation is illustrated in Figure 4.6 where the winning regions and winning cores for the two players in $\mathcal{G}'$ are shown as well (recall that $\mathcal{G}'$ is obtained from $\mathcal{G}$ by removing states in U).

Note first that $A_{1-k}(\mathcal{G})$ is contained in $W_{1-k}(\mathcal{G}')$ but that $A_k(\mathcal{G})$ can contain states in U. And as Proposition 4.16 tells us, $A_{1-k}(\mathcal{G}) = A_{1-k}(\mathcal{G}')$. The game $\mathcal{G}''$ is then obtained by removing the attractor for player $1-k$ of this winning core. This is not illustrated in the figure.

4.4.2 Memoryless strategies

In this subsection we will show that from winning core states, player j has a *memoryless strategy* σ which ensures that the play begins with an infinite number of consecutive j-dominating sequences. In fact, we will show something even stronger, namely the following.

Theorem 4.18 *Let $\mathcal{G}$ be a parity game and j be a player. There is a memoryless strategy σ for player j such that*

- $c(\text{Play}(\mathcal{G}, s, \sigma)) \subseteq \Lambda_j$ *for every* $s \in A_j(\mathcal{G})$
- $c(\text{Play}(\mathcal{G}, s, \sigma)) \subseteq \Xi_j$ *for every* $s \in W_j(\mathcal{G})$
- $c(\text{Play}(\mathcal{G}, s, \sigma)) \cap \Lambda_{1-j} = \emptyset$ *for every* $s \notin A_{1-j}(\mathcal{G})$

That is, player j has a memoryless strategy that ensures that the play begins with an infinite number of consecutive j-dominating sequences when the play starts in a winning core state. Moreover, it ensures that the play is winning when the play begins in a winning state for player j. Finally, it ensures that the play does not begin with an infinite number of $(1-j)$-dominating sequences when the play does not begin in a winning core state of player $1-j$.

In order to prove Theorem 4.18 we will use a result from [GZ05]. But first we need a few definitions. Let a *preference relation* be a binary relation on infinite sequences of colors (from a finite set of colors) that is reflexive, transitive and total. Let $\sqsubseteq_0$ be a preference relation for player 0. Intuitively, for two infinite sequences α and α' of colors we write $\alpha \sqsubseteq_0 \alpha'$ when α' is at least as good for player 0 as α. As we deal only with antagonistic games here, we assume that there is a corresponding preference relation $\sqsubseteq_1$ for player 1 such that for all infinite sequences α, α' of colors we have $\alpha \sqsubseteq_0 \alpha'$ if and only if $\alpha' \sqsubseteq_1 \alpha$. We write $\alpha \sqsubset_j \alpha'$ for player j when $\alpha \sqsubseteq_j \alpha'$ and $\alpha' \not\sqsubseteq_j \alpha$.

An *optimal strategy* σ_j^* for player j in a game $\mathcal{G}$ with preference relation $\sqsubseteq_j$ is a strategy such that for every state s, all strategies σ_j and σ_{1-j} of player j and $1-j$, respectively, the unique plays $\rho^* \in \text{Play}(\mathcal{G}, s, \sigma_j^*) \cap \text{Play}(\mathcal{G}, s, \sigma_{1-j})$ and $\rho \in \text{Play}(\mathcal{G}, s, \sigma_j) \cap \text{Play}(\mathcal{G}, s, \sigma_{1-j})$ satisfy $\rho \sqsubseteq_j \rho^*$.

We now define a total order $\trianglelefteq_j$ for player j with corresponding strict order $\triangleleft_j$ on $\{\Lambda_j, \Lambda_{1-j}, \Xi_j \setminus \Lambda_j, \Xi_{1-j} \setminus \Lambda_{1-j}\}$ by

$$\Lambda_{1-j} \triangleleft_j \Xi_{1-j} \setminus \Lambda_{1-j} \triangleleft_j \Xi_j \setminus \Lambda_j \triangleleft_j \Lambda_j$$

Note that an infinite sequence α of colors in a parity game belongs to exactly one of the four sets above. We write $\kappa(\alpha)$ for the set that α belongs to. For instance, $\kappa(\alpha) = \Xi_0 \setminus \Lambda_0$ for the infinite sequence $\alpha = 232222...$

Now, more specifically, let $\leq_j$ be a preference relation for player j on infinite sequences α, α' of colors induced by the order $\trianglelefteq_j$ as follows

$$\alpha \leq_j \alpha' \text{ if and only if } \kappa(\alpha) \trianglelefteq_j \kappa(\alpha')$$

As a special case of Proposition 7 in [GZ05] we have the following.

Proposition 4.19 *Let player 0 have preference relation* $\leq_0$ *and player 1 have preference relation* $\leq_1$*. If*

1. *every parity game* $\mathcal{G} = (S, S_0, S_1, R, c)$ *with* $S = S_0$ *has a memoryless optimal strategy for player 0 and*

2. *every parity game $\mathcal{G} = (S, S_0, S_1, R, c)$ with $S = S_1$ has a memoryless optimal strategy for player 1*

then in every parity game $\mathcal{G}$ both player 0 and player 1 have memoryless optimal strategies.

That is, there exist memoryless optimal strategies in every game if and only if there exist memoryless optimal strategies in every game where one player controls all the states.

As the preference relations $\leq_j$ are defined symmetrically, Proposition 4.19 tells us that if we can show that player 0 has a memoryless optimal strategy with preference relation $\leq_0$ in all parity games where player 0 controls every state then Theorem 4.18 follows. This is in fact the case.

Proposition 4.20 *Let $\mathcal{G} = (S, S_0, S_1, R, c)$ be a parity game with $S = S_0$. Then player 0 has a memoryless optimal strategy with preference relation $\leq_0$.*

Proof. First, we need the *reward order* $\prec_j$ for player j on colors which was introduced in [VJ00]. It is defined by

$$v \prec_j u \Leftrightarrow (v < u \wedge u \equiv j \pmod 2) \vee (u < v \wedge v \equiv 1 - j \pmod 2)$$

We let it be defined for 0 in this way as well. Intuitively, the preference order tells us which color player j would rather like to see during a play. For instance, if d is even then

$$d-1 \prec_0 d-3 \prec_0 ... \prec_0 1 \prec_0 0 \prec_0 2 \prec_0 ... \prec_0 d-2 \prec_0 d$$

Let $m(\rho)$ denote the largest non-initial color of a play ρ. For each state $s_0 \in W_0(\mathcal{G})$ there exists a play $\rho = s_0 s_1 ...$ such that $c(\rho) \in \Xi_0$. In particular, let $\mathrm{val}(s_0) = m(\rho)$ for a play ρ from s_0 such that $c(\rho) \in \Xi_j$ and such that $m(\rho)$ is maximal with respect to the reward order $\preceq_0$.

Intuitively, $\mathrm{val}(s_0)$ for a state $s_0 \in W_0(\mathcal{G})$ is the $\preceq_0$-best value that player 0 can ensure as a maximal color of a non-initial state of a play beginning in s_0 while still ensuring that he wins the play. Note that if $\mathrm{val}(s_0)$ is known for a state, then we can simply tell whether $s_0 \in \Lambda_0$ or $s_0 \in \Xi_0 \setminus \Lambda_0$ by checking whether $\mathrm{val}(s_0) \succeq_0 0$.

For states s_0 such that $\mathrm{val}(s_0) \succeq_0 0$ we define $\mathrm{dist}(s_0)$ to be the length of the shortest history $\rho = s_0 ... s_\ell$ from s_0 to a state s_ℓ with color $\mathrm{val}(s_0)$ such that for all $0 < i < \ell$ we have $c(s_i) < \mathrm{val}(s_0)$ and such that $\mathrm{val}(s_\ell) \succeq_0 \mathrm{val}(s_0) - 1$. Note that such a history must exist since $\mathrm{val}(s_0) \succeq_0 0$.

For states s_0 such that $\mathrm{val}(s_0) \prec_0 0$ we define $\mathrm{dist}(s_0)$ to be the length of the shortest history $\rho = s_0 ... s_\ell$ from s_0 to a state s_ℓ with color $\mathrm{val}(s_0)$ such that for all $0 < i < \ell$ we have $c(s_i) < \mathrm{val}(s_0)$ and such that $\mathrm{val}(s_\ell) \succ_0 \mathrm{val}(s_0)$. Note that such a history must exist since $\mathrm{val}(s_0) \prec_0 0$ and $s_0 \in W_0(\mathcal{G})$.

For states s with $\mathrm{val}(s) \succeq_0 0$ and $\mathrm{dist}(s) > 1$ there must exist a successor t of s such that $\mathrm{val}(t) = \mathrm{val}(s), c(t) < \mathrm{val}(s)$ and $\mathrm{dist}(t) = \mathrm{dist}(s) - 1$. Define $\mathrm{next}(s) = t$ for an arbitrary such successor t.

For states s with $\text{val}(s) \succeq_0 0$ and $\text{dist}(s) = 1$ there must exist a successor t of s such that $c(t) = \text{val}(s)$ and $\text{val}(t) \succeq_0 \text{val}(s) - 1$. Define $\text{next}(s) = t$ for an arbitrary such successor t.

For states s with $\text{val}(s) \prec_0 0$ and $\text{dist}(s) > 1$ there must exist a successor t of s such that $\text{val}(t) = \text{val}(s), c(t) < \text{val}(s)$ and $\text{dist}(t) = \text{dist}(s) - 1$. Define $\text{next}(s) = t$ for an arbitrary such successor t.

For states s with $\text{val}(s) \prec_0 0$ and $\text{dist}(s) = 1$ there must exist a successor t of s such that $c(t) = \text{val}(s)$ and $\text{val}(t) \succ_0 \text{val}(s)$. Define $\text{next}(s) = t$ for an arbitrary such successor t.

Now, define a memoryless strategy σ for player 0 by $\sigma(s) = \text{next}(s)$ for every $s \in W_0(\mathcal{G})$. For states in $W_1(\mathcal{G})$ there are no transitions to states in $W_0(\mathcal{G})$ as $S = S_0$. As player 0 cannot win from a state in $W_1(\mathcal{G})$ all he can hope to achieve is a play with corresponding color sequence that is not in Λ_1 which can be obtained if and only if the largest color in the play is even. This is also known as the weak parity condition. As weak parity games are memoryless determined [Cha08] player 0 can use a memoryless optimal strategy from the weak parity game obtained by restricting $\mathcal{G}$ to states in $W_1(\mathcal{G})$. Let σ play in this way from states in $W_1(\mathcal{G})$.

We will now show that σ is a memoryless optimal strategy for player 0 with preference relation $\leq_0$. As argued above this is already the case from states in $W_1(\mathcal{G})$. Thus, we now focus on states in $W_0(\mathcal{G})$.

Let ρ be the single play in $\text{Play}(\mathcal{G}, s_0, \sigma)$ for a state $s_0 \in W_0(\mathcal{G})$. We will now show that $c(\rho) \in \Xi_0$ and that $m(\rho) = \text{val}(s_0)$. These two properties imply that σ ensures that $c(\text{Play}(\mathcal{G}, s, \sigma)) \subseteq \Xi_0$ for every $s \in W_0(\mathcal{G})$ and that $c(\text{Play}(\mathcal{G}, s, \sigma)) \subseteq \Lambda_0$ for every $s \in A_0(\mathcal{G})$ as states s in $A_0(\mathcal{G})$ are exactly those states with $\text{val}(s) \succeq_0 0$.

Let $\rho = s_0 s_1 ...$ and let $i_0 < i_1 < ...$ be all indices such that $\text{dist}(s_{i_k}) = 1$ for every $k \geq 0$. Notice that there are infinitely many such indices since $\text{dist}(s_i) - 1 = \text{dist}(s_{i+1})$ whenever $\text{dist}(s_i) > 1$ and $\text{dist}(s_i) \geq 1$ for all $i \geq 0$.

As argued before, whenever $\text{dist}(s_\ell) > 1$ we have $\text{val}(s_\ell) = \text{val}(s_{\ell+1})$. When $\text{dist}(s_\ell) = 1$ and $\text{val}(s_\ell) \succeq_0 0$ then $\text{val}(s_{\ell+1}) \succeq_0 \text{val}(s_\ell) - 1$ which implies that $\text{val}(s_\ell) \geq \text{val}(s_{\ell+1})$. Finally, when $\text{dist}(s_\ell) = 1$ and $\text{val}(s_\ell) \prec_0 0$ then $\text{val}(s_\ell + 1) \succ_0 \text{val}(s_\ell)$ which implies that $\text{val}(s_\ell) \geq \text{val}(s_{\ell+1})$. Thus, we have

$$\text{val}(s_0) \geq \text{val}(s_1) \geq \text{val}(s_2) \geq ...$$

For all ℓ such that $\text{dist}(s_\ell) > 1$ we have $\text{val}(s_{\ell+1}) = \text{val}(s_\ell)$. However, when $\text{dist}(s_\ell) = 1$ and $\text{val}(s_\ell) \prec_0 0$ then $\text{val}(s_\ell) > \text{val}(s_{\ell+1})$. As there are infinitely many ℓ such that $\text{dist}(s_\ell) = 1$ there can only be finitely many ℓ such that $\text{val}(s_\ell) \prec_0 0$. Thus, there exists q such that

$$0 \preceq_0 \text{val}(s_q) = \text{val}(s_{q+1}) = \text{val}(s_{q+2}) = ...$$

Note that for all $\ell > 0$ we have $\text{val}(\ell) \geq c(\ell)$. However, we have $\text{val}(s_{i_k}) = c(s_{i_k+1})$ for all $k \geq 0$. Let p be the smallest index such that $i_p > q$. Then this implies that $c(s_{i_p+1}) = \text{val}(s_{i_p})$ is the largest color that occurs infinitely often in ρ. Thus,

$c(\rho) \in \Xi_0$. In addition, note that $c(s_{i_0+1}) = \text{val}(s_{i_0}) = \text{val}(s_0)$ is the largest non-initial color that occurs in ρ. Thus, $m(\rho) = \text{val}(s_0)$. This concludes the proof that σ is a memoryless optimal strategy. □

4.4.3 A sequence that converges quickly to the winning core

Let $A_j^i(\mathcal{G})$ be the set of states from which player j can ensure that the play begins with at least i consecutive j-dominating sequences. First, note that this defines an infinite decreasing sequence

$$A_j^0(\mathcal{G}) \supseteq A_j^1(\mathcal{G}) \supseteq ...$$

of sets of states and that $A_j^i(\mathcal{G}) \supseteq A_j(\mathcal{G})$ for all $i \geq 0$.

Theorem 4.21 $A_j^n(\mathcal{G}) = A_j(\mathcal{G})$

Proof. First note that $A_j^n(\mathcal{G}) \supseteq A_j(\mathcal{G})$. To show that $A_j^n(\mathcal{G}) \subseteq A_j(\mathcal{G})$ suppose for contradiction that $s \in A_j^n(\mathcal{G})$ and $s \notin A_j(\mathcal{G})$. By Theorem 4.18 player $1-j$ has a memoryless strategy σ in $\mathcal{G}$ such that $c(\text{Play}(\mathcal{G}, s, \sigma)) \cap \Lambda_j = \emptyset$.

Since $s \in A_j^n(\mathcal{G})$ there exists a play $\rho \in \text{Play}(\mathcal{G}, s, \sigma)$ that begins with n consecutive j-dominating sequences. Let $i_0 < ... < i_n$ be indices with $i_0 = 0$ such that $\rho_{i_k}\rho_{i_k+1}...\rho_{i_{i+1}}$ is j-dominating for all $0 \leq k < n$. As there are $n+1$ indices and only n different states in $\mathcal{G}$ there must exist two indices u, v with $u < v$ such that $\rho_{i_u} = \rho_{i_v}$. Now, the play $\pi = \rho_0...\rho_u(\rho_{u+1}...\rho_v)^\omega$ belongs to $\text{Play}(\mathcal{G}, s, \sigma)$ as well since σ is memoryless. This gives a contradiction since $c(\pi) \in \Lambda_j$ and $\text{Play}(\mathcal{G}, s, \sigma)$ contains no such play according to the definition of σ. □

This proposition implies that if there is a way to compute $A_j^i(\mathcal{G})$ from the sequence $A_j^0(\mathcal{G}), ..., A_j^{i-1}(\mathcal{G})$ in polynomial time for a given i then the winning core can be computed in polynomial time as the sequence converges after at most n steps. This would also imply that parity games could be solved in polynomial time.

To illustrate why it is not necessarily easy to compute this in a simple way consider again the parity game in Figure 4.4 and the history $h = s_0s_1s_2s_2s_2s_2$ which begins with the 5 consecutive 0-dominating sequences s_0s_1, s_1s_2, s_2s_2, s_2s_2 and s_2s_2. As player 1 controls every state of the game he might force the play after h to continue with the suffix $s_3s_0^\omega$. Now, the way we chopped h into 5 consecutive 0-dominating sequences cannot be extended such that the entire play $\rho = h \cdot s_3s_0^\omega$ begins with an infinite number of consecutive 0-dominating sequences as the color of s_3 is larger than all colors that appear later in ρ. However, if we pick the first 0-dominating sequence to be $s_0s_1s_2s_2s_2s_2s_3$ then it is easy to see that ρ begins with an infinite number of consecutive 0-dominating sequences. Thus, during the play of a game we might not know how to chop up the play in a way which ensures that the play begins with an infinite number of consecutive 0-dominating sequences when the play begins in a winning core state for player 0. However, we know that it is possible for player

```
ParityGameSolver(G):
A ← WinningCore(G, 0)
if A = ∅ then return (∅, S)
A' = Attr_0(G, A)
(W_0, W_1) ← ParityGameSolver(G ↾ (S \ A'))
return (A' ∪ W_0, W_1)
```

Figure 4.7: Solving parity games using winning core computation.

0 to force that the play begins with an infinite number of consecutive 0-dominating sequences.

4.5 Complexity of winning core computation

In this section we show how computation of winning cores can be used to solve parity games. Next, we provide a polynomial-time reduction of solving parity games to computing winning cores.

4.5.1 Solving parity games by winning core computation

By Proposition 4.9 the winning core for player j is a subset of the winning region. Thus, according to Proposition 4.5 we get the following corollary which forms the basis of a recursive algorithm for solving parity games by computing winning cores.

Corollary 4.22 *Let $\mathcal{G} = (S, S_0, S_1, R, c)$ be a parity game, $A' = \mathrm{Attr}_j(\mathcal{G}, A_j(\mathcal{G}))$ and $\mathcal{G}' = \mathcal{G} \upharpoonright (S \setminus A')$. Then*

- $W_j(\mathcal{G}) = A' \cup W_j(\mathcal{G}')$
- $W_{1-j}(\mathcal{G}) = W_{1-j}(\mathcal{G}')$

Given an algorithm WinningCore$(\mathcal{G}, j)$ that computes the winning core $A_j(\mathcal{G})$ for player j in $\mathcal{G}$ we can compute winning regions in parity games using the algorithm in Figure 4.7.

The algorithm first calculates the winning core for player 0. If it is empty then by Theorem 4.11 player 1 wins in all states. Otherwise, $A' = \mathrm{Attr}_0(\mathcal{G}, A_0(\mathcal{G}))$ is winning for player 0 and further, the remaining winning states can be computed by a recursive call on $\mathcal{G} \upharpoonright (S \setminus A')$ according to Corollary 4.22. Note that this game has a strictly smaller number of states than $\mathcal{G}$ as $A' \neq \emptyset$. Thus, the algorithm performs at most n recursive calls. This implies that if winning cores can be computed in polynomial time then parity games can be solved in polynomial time.

4.5.2 Reducing winning core computation to solving parity games

We have seen how existence of a polynomial-time algorithm for computing winning cores would imply the existence of a polynomial-time algorithm for solving parity games. Here, we show the converse by a reduction from computing winning cores to solving parity games.

We begin by introducing the notion of a *product game* $\mathcal{G}_j^\dagger$ of a parity game $\mathcal{G}$ for player j. Let $\mathcal{G} = (S, R, S_0, S_1, c)$ be a parity game with colors in $\{1, ..., d\}$ and $j \in \{0, 1\}$ be a player. Construct from this a game $\mathcal{G}_j^\dagger = (S', S_0', S_1', R', c')$ such that $S' = S \times \{0, 1, ..., d\}$, $S_j' = S_j \times \{0, 1, ..., d\}$ for $j \in \{0, 1\}$, $R' = \{((s, v), (s', v')) \in S' \times S' \mid (s, s') \in R \wedge v' = \max(v, c(s'))\}$. Finally, $c'(s, v) = c(s)$ if $v \equiv j \pmod 2$ and $c'(s, v) = v$ otherwise.

The idea is that the rules of $\mathcal{G}_j^\dagger$ when the play starts in $(s, 0)$ are the same as in $\mathcal{G}$ when the play starts in s, but with two main differences. The first is that in $\mathcal{G}_j^\dagger$, the greatest color that has occured during the play (excluding the color of the initial state) is recorded in the state. The second is that the colors of states in $\mathcal{G}_j^\dagger$ are as in $\mathcal{G}$ when the greatest color e occuring so far in the play satisfies $e \equiv j \pmod 2$. Otherwise, the state is colored e.

We define a bijection $\gamma_\mathcal{G} : \mathrm{Path}(\mathcal{G}) \times \{0, ..., d\} \to \mathrm{Path}(\mathcal{G}_j^\dagger)$ for $\rho = s_0 s_1 ... \in \mathrm{Path}(\mathcal{G})$ and $v \in \{0, ..., d\}$ by

$$\gamma_\mathcal{G}(\rho, v) = (s_0, w_0)(s_1, w_1)...$$

where $w_i = \max(v, \max_{0<k\leq i} c(s_i))$. In particular, $w_0 = v$.

For a pair $(s, v) \in S \times \{0, ..., d\}$ we define $\mathrm{st}(s, v) = s$ and $\mathrm{val}(s, v) = v$. This is extended to paths $\rho = (s_0, v_0)(s_1, v_1)...$ in $\mathcal{G}_j^\dagger$ such that the state sequence $\mathrm{st}(\rho) = s_0 s_1 ...$ and value sequence $\mathrm{val}(\rho) = v_0 v_1 ....$

The following two lemmas show that $s \in S$ is in the winning core of player j in $\mathcal{G}$ if and only if $(s, 0)$ is a winning state for player j in $\mathcal{G}_j^\dagger$. As $\mathcal{G}_j^\dagger$ has size polynomial in the size of $\mathcal{G}$ this gives a reduction from computing winning cores to computing winning regions.

Lemma 4.23 *Let $\mathcal{G}$ be a parity game and s be a state. Then $s \in A_j(\mathcal{G})$ implies $(s, 0) \in W_j(\mathcal{G}_j^\dagger)$.*

Proof. Let $s \in A_j(\mathcal{G})$. This implies that player j has a strategy σ in $\mathcal{G}$ such that $c(\mathrm{Play}(\mathcal{G}, s, \sigma)) \subseteq \Lambda_j$. Now, construct from this a strategy σ' for player j in $\mathcal{G}_j^\dagger$ defined by

$$\sigma'(h) = (\sigma(\mathrm{st}(h)), \max(v_\ell, c(\sigma(\mathrm{st}(h)))))$$

for every history $h = (s_0, v_0)...(s_\ell, v_\ell)$ in $\mathcal{G}_j^\dagger$.

Consider an arbitrary play

$$\rho' = (s_0, v_0)(s_1, v_1)... \in \mathrm{Play}(\mathcal{G}_j^\dagger, (s, 0), \sigma')$$

from $(s,0)$ compatible with σ'. By the definition of σ' we have that $\rho = s_0 s_1 ... \in \text{Play}(\mathcal{G}, s, \sigma)$ and thus $c(\rho) \in \Lambda_j$ as for every $i \geq 0$ such that $(s_i, v_i) \in S_j$ we have $s_{i+1} = \sigma(s_0 ... s_i)$.

Let e be the largest color that occurs in ρ. As $c(\rho) \in \Lambda_j$ we have $e \equiv j \pmod 2$. Thus, there exists an ℓ such that $v_i = e$ for all $i \geq \ell$. This implies that $c'((s_i, v_i)) = c(s_i)$ for all $i \geq \ell$ by the definition of $\mathcal{G}_j^\dagger$. Thus, the sequence of colors occuring in $\rho'_{\geq \ell}$ is the same as in $\rho_{\geq \ell}$ and therefore $c(\rho') \in \Xi_j$ using Lemma 4.6. As ρ' was chosen arbitrarily from $\text{Play}(\mathcal{G}_j^\dagger, (s,0), \sigma')$ we have $(s,0) \in W_j(\mathcal{G}_j^\dagger)$. □

Lemma 4.24 *Let $\mathcal{G}$ be a parity game and s be a state. Then $(s,0) \in W_j(\mathcal{G}_j^\dagger)$ implies $s \in A_j(\mathcal{G})$.*

Proof. Suppose that $(s,0) \in W_j(\mathcal{G}_j^\dagger)$. Then player j has a strategy σ' in $\mathcal{G}_j^\dagger$ such that $c(\text{Play}(\mathcal{G}_j^\dagger, (s,0), \sigma')) \subseteq \Xi_j$. Define a strategy σ of player j for every history h in $\mathcal{G}$ by

$$\sigma(h) = \text{st}(\sigma'(\gamma_{\mathcal{G}}(h,0)))$$

Consider an arbitrary play $\rho = s_0 s_1 ... \in \text{Play}(\mathcal{G}, s, \sigma)$ from s compatible with σ. By the definition of σ and $\mathcal{G}_j^\dagger$ we have that

$$\rho' = \gamma_{\mathcal{G}}(\rho, 0)$$

belongs to $\text{Play}(\mathcal{G}_j^\dagger, (s,0), \sigma')$ and thus $c(\rho') \in \Xi_j$. This implies that the greatest color e occuring in ρ' satisfies $e \equiv j \pmod 2$ by the definition of $\mathcal{G}_j^\dagger$. Further, the greatest color e' that occurs infinitely often in ρ' also satisfies $e' \equiv j \pmod 2$. We have that e and e' are also the greatest color occuring and greatest color occuring infinitely often respectively in ρ. Using Proposition 4.8 this implies that $c(\rho) \in \Lambda_j$. As ρ is an arbitrary play in $\text{Play}(\mathcal{G}, s, \sigma)$ we have $s \in A_j(\mathcal{G})$. □

This means that solving parity games can be done in polynomial time if and only if winning cores can be computed in polynomial time. We also have that computing winning cores is in $\text{NP} \cap \text{coNP}$ and $\text{UP} \cap \text{coUP}$ like parity games [Jur98].

Theorem 4.25 *Computing winning cores is in $\text{NP} \cap \text{coNP}$ and $\text{UP} \cap \text{coUP}$.*

This fact is important as it makes the search for a polynomial-time algorithm for computing winning cores a viable direction in the search for a polynomial-time algorithm for solving parity games. This had not been the case if computing winning cores was e.g. NP-hard (which is still possible, but only if $\text{NP} = \text{coNP}$).

4.6 A polynomial-time approximation algorithm

A natural approach for computing the winning core is to try to apply Theorem 4.21 using an algorithm resembling the standard algorithm for solving Büchi games using repeated attractor computations [Tho95]. The idea is to first compute the set of states from which player j can ensure that the play begins with one j-dominating sequence, then use this to compute the set of states from which player j can ensure that the play begins with two consecutive j-dominating sequences, then three consecutive j-dominating sequences and so on until convergence. However, this turns out not to be so simple to do efficiently. In this section we propose a polynomial-time algorithm using the intuition above, but which is only guaranteed to compute a subset of the winning core. However, as we will see in Section 4.7, this algorithm turns out to be very fast in practice and solves many games completely. We will show to make it work using $O(d \cdot n^2 \cdot (n+m))$ time and $O(d+n+m)$ space.

4.6.1 The basic algorithm

For a parity game $\mathcal{G}$, a player j and integer $i \geq 0$ we define sets $B_j^i(\mathcal{G})$ as underapproximations of $A_j^i(\mathcal{G})$ by $B_j^0(\mathcal{G}) = S$ and by letting $B_j^{i+1}(\mathcal{G})$ be the set of states from which player j can force the play to begin with a j-dominating sequence ending in $B_j^i(\mathcal{G})$. Formally, for every $i \geq 0$ let $B_j^{i+1}(\mathcal{G}) \subseteq B_j^i(\mathcal{G})$ contain all states s such that that there exists a strategy σ for player j satisfying

$$\forall \rho \in \mathrm{Play}(\mathcal{G}, s, \sigma). \exists k. \rho_{\leq k} \text{ is } j\text{-dominating and } \rho_k \in B_j^i(\mathcal{G})$$

Note that this sequence converges in at most n steps since it is decreasing. Let the limit of this sequence be $B_j(\mathcal{G})$.

Proposition 4.26 $B_j(\mathcal{G}) \subseteq A_j(\mathcal{G})$

Proof. For every $s \in B_j(\mathcal{G})$ there exists a strategy σ_s for player j such that for every $\rho \in \mathrm{Play}(\mathcal{G}, s, \sigma_s)$ there exists k such that $\rho_{\leq k}$ is j-dominating and $\rho_k \in B_j(\mathcal{G})$.

Now, define a strategy σ for player j in $\mathcal{G}$ on histories with initial state $s_0 \in B_j(\mathcal{G})$ (and arbitrarily for all other initial states) as follows. Let σ play like σ_{s_0} until it reaches a state s_1 in a way such that the sequence of states from s_0 to s_1 is j-dominating and $s_1 \in B_j(\mathcal{G})$. From this point on σ plays like σ_{s_1} would do if the play started in s_1 until a state s_2 is reached in a way such that the sequence of states from s_1 to s_2 is j-dominating and $s_2 \in B_j(\mathcal{G})$. Let σ prescribe continuing in this fashion indefinitely.

With this definition we have that $\mathrm{Play}(\mathcal{G}, s_0, \sigma) \subseteq \Lambda_j$ for every $s_0 \in B_j(\mathcal{G})$ as every play must begin with an infinite number of consecutive j-dominating sequences. Thus, $s_0 \in B_j(\mathcal{G})$ implies $s_0 \in A_j(\mathcal{G})$. □

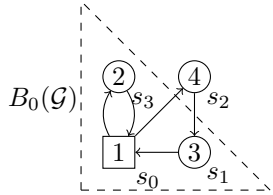


Figure 4.8: A parity game with no fatal attractors where $B_0(\mathcal{G}) = A_0(\mathcal{G}) = \{s_0, s_1, s_3\}$.

Remark 4.27 *Note that we do not always have $B_j(\mathcal{G}) = A_j(\mathcal{G})$. For instance, this is not the case in the game in Figure 4.4 where $A_0(\mathcal{G}) = \{s_0, s_3\}$ but $B_0^1(\mathcal{G}) = \{s_0, s_1, s_3\}$, $B_0^2(\mathcal{G}) = \{s_0, s_3\}$, $B_0^3(\mathcal{G}) = \{s_3\}$ and $B_0^4(\mathcal{G}) = B_0(\mathcal{G}) = \emptyset$. The reason is that from s_0 player 0 cannot force the play to ever get back to the set $\{s_0, s_3\}$ as player 1 controls all states. It can be shown that $A_j^1(\mathcal{G}) = B_j^1(\mathcal{G})$ always, but there are parity games where $A_j^2(\mathcal{G}) \neq B_j^2(\mathcal{G})$.*

However, as we shall see later, this underapproximation of the winning core is very good as a tool to compute underapproximations of winning regions in parity games. And in practice, it is often good enough to compute the entire winning regions. Moreover, we will show that $B_j(\mathcal{G})$ can be computed in polynomial time and linear space. To motivate the practicality we note that the underapproximation $B_j(\mathcal{G})$ contains all fatal attractors for player j. It was shown in [HKP13; HKP14] that just being able to compute fatal attractors is enough to solve a lot of games in practice. Figure 4.8 shows an example where $B_j(\mathcal{G})$ contains even more states than just fatal attractors.

Let $[1, d]_j = \{v \in \{1, ..., d\} \mid v \equiv j \pmod 2\}$. We can now show the following proposition which provides us with a naïve way to compute $B_j^{i+1}(\mathcal{G})$ given that we know $B_j^i(\mathcal{G})$.

Lemma 4.28 *Let $i \geq 0$ be an integer and j be a player. Then*

$$s \in B_j^{i+1}(\mathcal{G}) \text{ if and only if } (s, 0) \in \mathrm{Attr}_j(\mathcal{G}_j^\dagger, B_j^i(\mathcal{G}) \times [1, d]_j)$$

where $\mathcal{G}_j^\dagger$ is the product game of $\mathcal{G}$ for player j.

Proof. ($\Rightarrow$) Suppose that $s \in B_j^{i+1}(\mathcal{G})$. Then there exists a strategy σ for player j in $\mathcal{G}$ such that for every $\rho = s_0 s_1 ... \in \mathrm{Play}(\mathcal{G}, s, \sigma)$ there exists $k > 0$ such that $s_0 ... s_k$ is j-dominating and $s_k \in B_j^i(\mathcal{G})$. Define a strategy σ' in $\mathcal{G}_j^\dagger$ by

$$\sigma'(h) = (\sigma(\mathrm{st}(h)), \max(v_\ell, c(\sigma(\mathrm{st}(h)))))$$

for every history $h = (t_0, v_0)...(t_\ell, v_\ell)$ in $\mathcal{G}_j^\dagger$. Let $\rho' = (s_0', v_0')(s_1', v_1')...$ be an arbitrary play in $\mathrm{Play}(\mathcal{G}_j^\dagger, (s, 0), \sigma')$. Then $s_0' s_1' ... \in \mathrm{Play}(\mathcal{G}, s, \sigma)$. Thus, there exists k such

that $s'_0...s'_k$ is j-dominating and $s'_k \in B^i_j(\mathcal{G})$. This implies that $v'_k \equiv j \pmod 2$ and thus $(s'_k, v'_k) \in B^i_j(\mathcal{G}) \times [1,d]_j$. As ρ' was arbitrarily chosen in $\mathrm{Play}(\mathcal{G}^\dagger_j, (s,0), \sigma')$ this means that $(s,0) \in \mathrm{Attr}_j(\mathcal{G}^\dagger_j, B^i_j(\mathcal{G}) \times [1,d]_j)$.

($\Leftarrow$) For the other direction suppose that $(s,0) \in \mathrm{Attr}_j(\mathcal{G}^\dagger_j, B^i_j(\mathcal{G}) \times [1,d]_j)$. Then there exists a strategy σ' for player j in $\mathcal{G}^\dagger_j$ such that for every $\rho' = (s_0,v_0)(s_1,v_1)... \in \mathrm{Play}(\mathcal{G}^\dagger_j, (s,0), \sigma')$ there exists k such that $\rho'_k \in B^i_j(\mathcal{G}) \times [1,d]_j$. Define a strategy σ for player j in $\mathcal{G}$ by

$$\sigma(h) = \mathrm{st}(\sigma'(\gamma_{\mathcal{G}}(h,0)))$$

for every history h in $\mathcal{G}$. Let $\rho = s_0 s_1 ...$ be an arbitrary play in $\mathrm{Play}(\mathcal{G}, s, \sigma)$. By the definition of σ and $\mathcal{G}^\dagger_j$ we have that

$$\rho' = \gamma_{\mathcal{G}}(\rho, 0)$$

belongs to $\mathrm{Play}(\mathcal{G}^\dagger_j, (s,0), \sigma')$. Thus, there exists k such that $\rho'_k \in B^i_j(\mathcal{G}) \times [1,d]_j$. This implies that $s_0...s_k$ is j-dominating and that $s_k \in B^i_j(\mathcal{G})$. As ρ was chosen arbitrarily from $\mathrm{Play}(\mathcal{G}, s, \sigma)$ this means that $s \in B^{i+1}_j(\mathcal{G})$. □

Note that Lemma 4.28 makes us able to compute $B_j(\mathcal{G})$ in time $O(d \cdot n \cdot (n+m))$ and space $O(d \cdot (n+m))$. This is because the sequence converges in at most n steps and in each step we just have to compute the attractor set in $\mathcal{G}^\dagger_j$ which has $O(d \cdot n)$ states and $O(d \cdot m)$ transitions.

4.6.2 Improving the complexity

We will now show how to improve the space complexity to $O(d + n + m)$ while keeping the same time complexity. Indeed, we will show how to compute $B_j(\mathcal{G})$ without actually having to construct $\mathcal{G}^\dagger_j$ explicitly. This makes a very large difference in practice, especially when the number of colors is large.

Recall that $\prec_j$ is the reward order for player j. We can now show that the attractor set needed to compute $B^{i+1}_j(\mathcal{G})$ in $\mathcal{G}^\dagger_j$ is upward-closed in the following sense.

Lemma 4.29 *Let $s \in S$, $i \geq 0$ and $k \geq 0$ be such that $(s,k) \in \mathrm{Attr}_j(\mathcal{G}^\dagger_j, B^i_j(\mathcal{G}) \times [1,d]_j)$. Then for all $k' \succ_j k$ we have*

$$(s,k') \in \mathrm{Attr}_j(\mathcal{G}^\dagger_j, B^i_j(\mathcal{G}) \times [1,d]_j)$$

Proof. For a strategy σ in $\mathcal{G}^\dagger_j$ and $v \in \{0,...,d\}$ define a strategy σ_v for every history $s_0...s_\ell$ in $\mathcal{G}$ by

$$\sigma_v(s_0...s_\ell) = \mathrm{st}(\sigma((s_0,v)...(s_\ell, \max(v, \max_{0<\ell' \leq \ell} c(s_{\ell'})))))$$

Suppose $(s,k) \in \mathrm{Attr}_j(\mathcal{G}_j^\dagger, B_j^i(\mathcal{G}) \times [1,d]_j)$. Then there exists a particular strategy σ for player j in $\mathcal{G}_j^\dagger$ such that for every $\rho \in \mathrm{Play}(\mathcal{G},(s,k),\sigma)$ there exists q such that $\rho_q \in B_j^i(\mathcal{G}) \times [1,d]_j$. Let $k' \succ_j k$ and let σ' be a strategy for player j in $\mathcal{G}_j^\dagger$ defined by

$$\sigma'((s_0,v_0)...(s_\ell,v_\ell)) = (\sigma_k(s_0...s_\ell), \max(v_\ell, c(\sigma_k(s_0...s_\ell))))$$

for every history $h = (s_0,v_0)...(s_\ell,v_\ell)$ with $s_0 = s$ and $v_0 = k'$.

We now want to show that σ' ensures that a state in $B_j^i(\mathcal{G}) \times [1,d]_j$ is eventually reached when the play begins in (s,k'). From this the lemma follows.

Consider a given play

$$\rho' = (s_0,v_0)(s_1,v_1)... \in \mathrm{Play}(\mathcal{G}_j^\dagger,(s,k'),\sigma')$$

By the definition of σ' we have $s_0 s_1... \in \mathrm{Play}(\mathcal{G},s,\sigma_k)$. Further, the sequence

$$\rho = (s_0,w_0)(s_1,w_1)...$$

where $w_0 = k$ and $w_{\ell+1} = \max(w_\ell, c(s_{\ell+1}))$ for all $\ell \geq 0$ belongs to $\mathrm{Play}(\mathcal{G},(s,k),\sigma)$. Thus, there exists q such that $\rho_q \in B_j^i(\mathcal{G}) \times [1,d]_j$. This means that either

1. $w_0 \equiv j \pmod 2$ and $w_0 = w_q$ or
2. $w_q = \max(c(s_\ell))_{1 \leq \ell \leq q} > w_0$

In the first case we have $v_0 > w_0$ and $v_0 \equiv j \pmod 2$ since $w_0 \equiv j \pmod 2$ and $v_0 \succ_j w_0$ which implies $v_q = v_0$. Thus, in this case $(s_q,v_q) \in B_j^i(\mathcal{G}) \times [1,d]_j$.

In the second case, if $v_0 \leq w_0$ then $(s_q,v_q) = (s_q,w_q) \in B_j^i(\mathcal{G}) \times [1,d]_j$ immediately. On the other hand, suppose $v_0 > w_0$. Since $v_0 \succ_j w_0$ this implies $v_0 \equiv j \pmod 2$ by the definition of $\succ_j$. Since either $v_q = v_0$ or $v_q = \max(c(s_\ell))_{1 \leq \ell \leq q} = w_q$ this implies $(s_q,v_q) \in B_j^i(\mathcal{G}) \times [1,d]_j$. □

We can use Lemma 4.29 to compute $\mathrm{Attr}_j(\mathcal{G}_j^\dagger, B_j^i(\mathcal{G}) \times [1,d]_j)$ as follows. In each step of the attractor computation we store for each state $s \in S$ the $\prec_j$-smallest value k such that (s,k) belongs to the part of the attractor set computed so far. Thus, in each step we store an n-dimensional vector $\mathbf{k} = (k_0,...,k_{n-1})$ of these values, one for each state. In each step of the attractor computation we compute the $\prec_j$-smallest values **k'** in the next step of the attractor computation based on $\mathbf{k}$. Using a technique similar to the way the standard attractor set can be computed in time $O(n+m)$ [AHK98] the computation of $\mathrm{Attr}_j(\mathcal{G}_j^\dagger, B_j^i(\mathcal{G}) \times [1,d]_j)$ can be done in this fashion in time $O(d \cdot (n+m))$. Thus, $B_j(\mathcal{G})$ can be computed using $O(d \cdot n \cdot (n+m))$ time and $O(n+m+d)$ space.

```
PartialSolver(𝒢):
A ← B_0(𝒢)
if A ≠ ∅ then
    A' = Attr_0(𝒢, A)
    (W_0, W_1) ← PartialSolver(𝒢 ↾ (S \ A'))
    return (A' ∪ W_0, W_1)
A ← B_1(𝒢)
if A ≠ ∅ then
    A' = Attr_1(𝒢, A)
    (W_0, W_1) ← PartialSolver(𝒢 ↾ (S \ A'))
    return (W_0, A' ∪ W_1)
return (∅, ∅)
```

Figure 4.9: A partial solver for parity games based on winning cores.

4.6.3 Partially solving parity games

Using a similar idea as in the algorithm from Section 4.5 we present an algorithm for solving parity games partially which relies on the underapproximation $B_j(\mathcal{G})$. It can be seen in Figure 4.9.

This algorithm uses the procedure outlined in the previous subsection for computing the underapproximation $B_j(\mathcal{G})$. It is guaranteed to return underapproximations of the winning regions according to Proposition 4.26. Further, as each call to the algorithm makes at most one recursive call to a game with fewer states there are at most $O(n)$ recursive calls in total. Thus, the algorithm for partially solving parity games runs in time $O(d \cdot n^2 \cdot (n+m))$. It can be implemented to use $O(n+m+d)$ space.

4.6.4 Quality of approximation

For approximation algorithms a widely used notion is that of *approximation ratio* (see e.g. [WS11]) which is used to give guarantees on the value of an approximation.

A meaningful way to define approximation ratio in parity games is to say that an algorithm is an α-approximation algorithm for $0 < \alpha \leq 1$ if the algorithm always decides the winning player of at least $\lceil \alpha \cdot n \rceil$ states where n is the number of states in the game. The problem with this, however, is that if there exists a polynomial-time α-approximation algorithm for some $0 < \alpha \leq 1$ then this algorithm can be used to solve parity games completely in polynomial time. One could run such an algorithm and remove the attractor sets of the winning states it finds. Then, run the approximation algorithm on the remaining game and continue in the same fashion until the entire winning regions are computed.

This tells us that it will probably be hard to show that there exists a polynomial-time α-approximation algorithm as this would show solvability of parity games in

polynomial time. In particular, our partial solver is not an α-approximation algorithm.

A game that the partial solver cannot solve is the one in Figure 4.4. The reason is that from every state player 1 can force the play to leave as well as stay outside of the winning core for player 0. This simple example implies that the algorithm is not guaranteed to solve games completely on standard subclasses of games investigated in the litterature such as games with bounded tree-width [Obd03], bounded DAG-width [Ber+06] and other games with restrictions on the game graph [DKT12]. Though, the algorithm always solves Büchi games completely and it does so in time $O(nm)$.

Despite the lack of theoretical guarantees we will show that the algorithm performs remarkably well in practice, with respect to solving games completely and with respect to running time.

4.7 Experimental results

We present experimental results for the improved version of the winning core approximation algorithm presented in Section 4.6, it is called the `WC` algorithm for the remainder of this section.

The experimental results are both performed to investigate how often the algorithm solves games completely and to investigate the running-time of the algorithm in practice compared to existing parity game solvers. The algorithm has been implemented in OCaml on top of the PGSOLVER framework [FL09].

We both compare with results for state-of-the-art complete solvers implemented in the PGSOLVER framework, namely

- `Zie`: Zielonka's algorithm [Zie98]
- `DD`: Dominion decomposition algorithm [JPZ06]
- `SI`: Strategy improvement algorithm [VJ00]
- `SPM`: Small progress measures algorithm [Jur00]
- `BS`: Big step algorithm [Sch07]

and with the partial solver `psolB` from [HKP13; HKP14] that is based on fatal attractor computation. The experiments with the `WC` algorithm and the other solvers from the PGSOLVER framework have been performed on a machine with an Intel® Core^{TM} i7-4600M CPU with 4 2.90GHz processors and 15.6GiB memory. All optimizations of the PGSOLVER framework were disabled in all experiments. The `WC` algorithm uses the same basic data structures as the other solvers from the PGSOLVER framework. All results of the partial solver `psolB` are taken from [HKP14]. Thus, one should be careful about these results as it was implemented in Scala and experiments were run on a different machine.

Game	psolB		WC	
	k	$\lvert S\rvert$	k	$\lvert S\rvert$
Clique	5.232	5.232	8.979	8.979
Ladder	1.596.624	3.193.248	7.308.357	14.616.714
Jur(10,k)	2.890	121.380	4.784	200.928
Jur(k,10)	4.380	175.220	96.881	3.875.260
Jur(k,k)	200	160.400	635	1.614.170
RecLad	2.064	10.320	14.008	70.040
MCLad	12.288	36.865	5.178.332	15.534.997
Hanoi	10	236.196	13	6.377.292
Elev	6	108.336	8	7.744.224

Table 4.10: The table shows the maximum input parameter k and number $|S|$ of states for which the solvers terminated within 20 minutes. The numbers for `psolB` are from [HKP14].

4.7.1 Benchmark games

Experiments have been performed on benchmark games from the classes `Clique Games`, `Ladder Games`, `Jurdzinski Games`, `Recursive Ladder Games`, `Model Checker Ladder Games`, `Towers of Hanoi`, `Elevator Verification` and `Language Inclusion` of the PGSOLVER framework. `WC` solved all these benchmark games completely. As reported in [HKP14] the best existing partial solver `psolB` solves all games completely except for the `Elevator Verification` games.

In Table 4.10 we compare `WC` to `psolB` with respect to the size of benchmark games solvable within 20 minutes. `WC` vastly outperforms `psolB` in all cases considered solving games with between 1.65 and 421 times as many states in 20 minutes depending on the benchmark.

Comparison of running time[1] of the complete solvers and `WC` can be seen in Figures 4.11 and 4.12. It can be seen that in the experiments `WC` never performs much worse than the best state-of-the-art complete solvers and in some cases it vastly outperforms the complete solvers. Thus, it seems to be very robust compared to the best complete solvers each of which have games on which they perform poorly compared to the rest.

4.7.2 Random games

Although random games are not necessarily good representatives for real-life instances of parity games they can give us some indication of the quality of a partial solver. In order to compare with the results in [HKP14] we have used the same program with the same parameters for generating random games.

[1] For the recursive ladder games some solvers were much better with an odd input parameter and some were much better with an even input parameter. Thus, for each input k in the data set, the running time on both input k and $k+1$ was measured and the worst result is displayed in the plot.

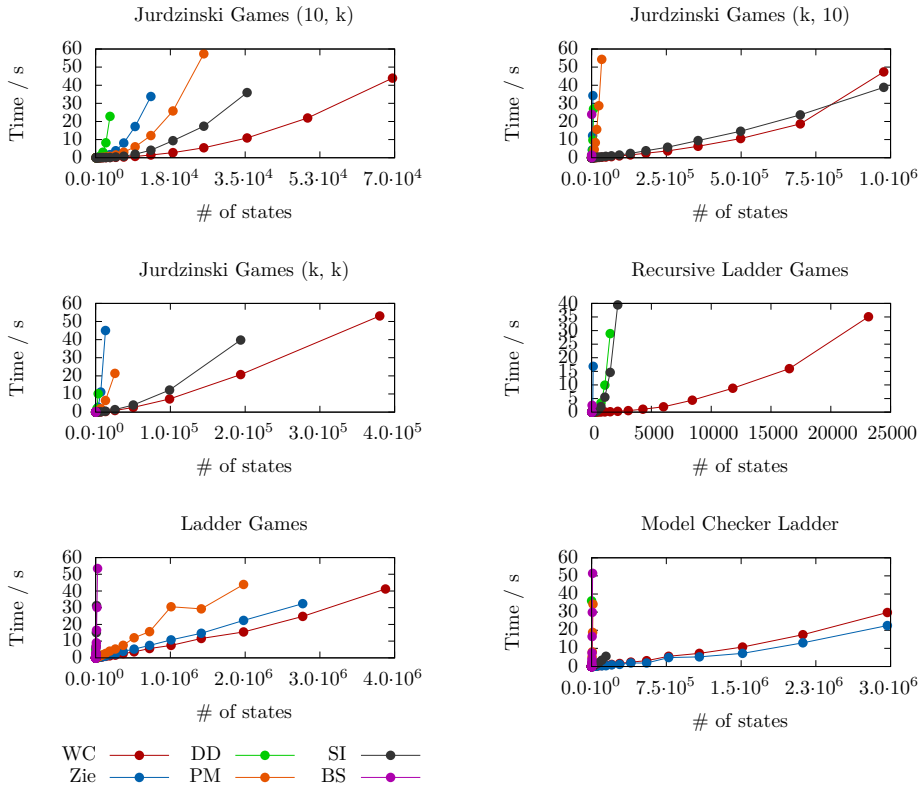


Figure 4.11: Performance for benchmark games.

The games are generated using the `randomgame` function in the PGSOLVER framework which takes as input n, d, ℓ, u where n is the number of states, the color of each state is chosen uniformly at random in $[1, d]$ and each node has a number of successors chosen uniformly at random in $[\ell, u]$ without any self-loops. 16 different configuration settings were chosen as in [HKP14]: $n = 500$ for all parameter settings, $d \in \{5, 50, 250, 500\}$ and $(\ell, u) \in \{(1, 5), (5, 10), (1, 100), (50, 250)\}$. 100.000 games were solved for each configuration and the results are shown in Table 4.13.

It can be seen that the algorithm only failed to solve 295 of the 1.600.000 games completely and thus solved 99.98% of the games completely. For the 295 games that were not solved completely by the winning core algorithm it still found the winning player for 56% of the states on average. Also note that the algorithm only failed to solve games with a very low out-degree as was the case for `psolB`. For the dense games it solved all the games completely.

Compared to `psolB` the `WC` solver does very well. The partial solver `lift(X)`

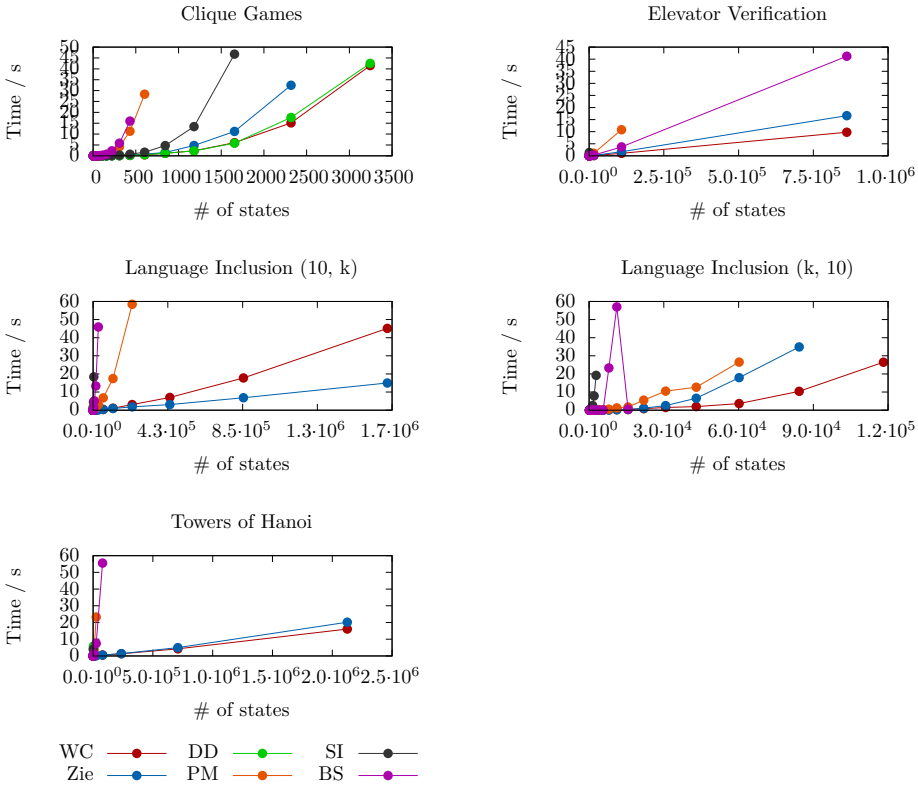


Figure 4.12: Performance for benchmark games.

d	(l, u)	# n.c.s. `psolB`	# n.c.s. `lift(psolB)`	# n.c.s. `WC`
5	$(1, 5)$	1275	233	258
50	$(1, 5)$	1030	43	9
250	$(1, 5)$	1138	36	16
500	$(1, 5)$	1086	35	12
5	$(5, 10)$	0	0	0
50	$(5, 10)$	1	0	0
250	$(5, 10)$	2	0	0
500	$(5, 10)$	2	0	0

Table 4.13: Table shows the number random of games out of 100.000 that were not completely solved (# n.c.s.) by the solvers for each configuration. For all configurations with $(\ell, u) \in \{(1, 100), (50, 250)\}$ all solvers solved all games completely.

is a generic solver from [HKP14] which uses a partial solver X to improve in cases where X does not solve the complete game. It also runs in polynomial time but gives a very large overhead in practice as it potentially calls the solver X a quadratic number of times in the number of transitions of a game. Even compared with this generic optimization of `psolB` the `WC` solver does well with respect to solving games completely.

4.8 Summary

We have introduced winning cores in parity games and motivated their importance by showing a number of interesting properties about them. In particular, they are interesting to investigate due to the fact that they are not necessarily dominions and because emptiness of the winning core of a player is equivalent to emptiness of the winning region of the player. Further, we have provided a new algorithm for solving parity games approximately which increases the size of parity games that can be solved in practice siginificantly compared to existing techniques.

CHAPTER 5

Model-checking in one-counter systems

This chapter is an adapted version of the paper

- [Ves15] Steen Vester. "On the Complexity of Model-Checking Branching and Alternating-Time Temporal Logics in One-Counter Systems". In: *Proceedings of the 13th International Symposium on Automated Technology for Verification and Analysis (ATVA)*. 2015, pages 361–377

It has been updated to be compatible with the other chapters of the thesis. More proof details are presented in this chapter, especially for the lower bound proofs. Further, an example of a model-checking game has been added together with a few new references.

5.1 Introduction

In this chapter focus is on the *model-checking problem* for some of the simplest infinite-state systems one can construct, namely finite-state systems combined with an *unbounded counter* that can hold a non-negative integer value. The complexity of model-checking such systems has been determined for LTL [DG09; Göl+10] and CTL [Göl+10; GL13] but not yet for CTL*, ATL and ATL* which is the main purpose here. Recently it was also shown that reachability in succinct one-counter games is ExpSpace-complete [Hun15].

Another focus of this chapter is to consider generalizations of the logics capable of expressing combined qualitative and quantitative properties of systems. This is done by extending to subsets of the *quantitative alternating-time temporal logics* QATL and QATL* [BG13] making it possible to compare the counter value with constants. This extension lets us express many interesting properties of systems in a simple way. As an example, consider deciding the winner in an energy game [Bou+08] modelling systems in which a controller needs to keep an energy level positive. This can be done by model-checking the QATL formula $\langle\langle \texttt{Ctrl} \rangle\rangle \mathbf{G}(r > 0)$ where r is used to denote the current value of the counter.

Let us give another example of a QATL* specification. Consider the game in Figure 5.1 modelling the interaction between the controller of a vending machine and an environment. The environment controls the rectangular states and the controller

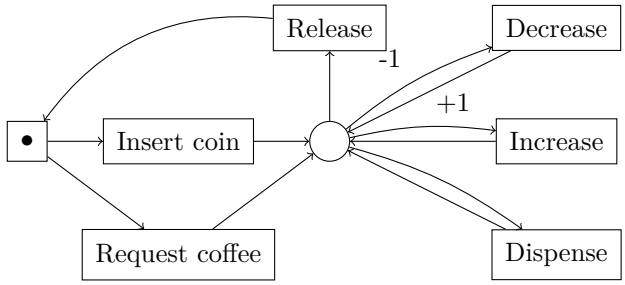


Figure 5.1: Model of interaction between a vending machine controller and an environment.

controls the circular state. Initially, the environment can insert a coin or request coffee. Upon either input the controller can decrease or increase the balance, dispense coffee or release control to the environment again. Some examples of specifications in QATL* using this model are

- $\langle\!\langle \{\texttt{Ctrl}\} \rangle\!\rangle \mathbf{G}(\text{Request} \wedge (r < 3) \rightarrow \mathbf{XX}\,\text{Release})$: The controller can make sure that control is released immediately whenever coffee is requested and the balance is less than 3.
- $\langle\!\langle \{\texttt{Ctrl}\} \rangle\!\rangle \mathbf{G}(\text{Request} \wedge (r \geq 3) \rightarrow \mathbf{F}\,\text{Dispense})$: The controller can make sure that whenever coffee is requested and the balance is at least 3 then eventually a cup of coffee is dispensed.

It is indeed quite natural to model systems with a resource (e.g. battery level, time, money) using a counter where production and consumption correspond to increasing and decreasing the counter respectively. Extending to several counters would be meaningful, but as reachability games are already undecidable for games with two counters [BJK10] the restriction to a single counter is very important.

Contribution

The contribution of this chapter is to present algorithms and complexity results for model-checking of CTL*, ATL and ATL* in one-counter systems. The complexity is investigated both in terms of whether only edge weights in $\{-1, 0, +1\}$ can be used (*non-succinct systems*) or if we allow any integer weights encoded in binary (*succinct systems*). We also distinguish between *data complexity* and *combined complexity*. In data complexity, the formula is assumed to be fixed whereas in combined complexity both the formula and the game are parameters.

We characterize the complexity of all the model-checking problems that arise from these distinctions. For CTL* the results on data complexity follow directly from results in [Ser06; Göl+10; GL13] even though this is not mentioned explicitly. We also show that the logics considered can be extended with quantitative constraints without

Table 5.2: Complexity of model-checking in non-succinct systems.

	Non-succinct	
	Data	Combined
CTL	PSpace [Ser06; GL13]	PSpace [Ser06; GL13]
CTL*	PSpace [Ser06; GL13]	ExpSpace (Theorem 5.8)
μ-calculus	PSpace [Ser06; JS07; Hol95]	PSpace [Ser06; JS07; Hol95]
ATL	PSpace (Theorem 5.5)	PSpace (Theorem 5.5)
ATL*	PSpace (Theorem 5.7)	2ExpSpace (Theorem 5.7)

Table 5.3: Complexity of model-checking in succinct systems.

	Succinct	
	Data	Combined
CTL	ExpSpace [Göl+10]	ExpSpace [Göl+10]
CTL*	ExpSpace [Göl+10]	ExpSpace (Theorem 5.8)
μ-calculus	ExpSpace [Göl+10]	ExpSpace [Göl+10]
ATL	ExpSpace (Theorem 5.5)	ExpSpace (Theorem 5.5)
ATL*	ExpSpace (Theorem 5.7)	2ExpSpace (Theorem 5.7)

Table 5.4: Complexity of deciding the winner in one-counter games with LTL objectives. The results are from Corollary 5.13.

Non-succinct		Succinct	
Data	Combined	Data	Combined
PSpace	2ExpSpace	ExpSpace	2ExpSpace

a jump in complexity and that deciding the winner in a one-counter game with LTL objectives is 2ExpSpace-complete in all cases considered as for ATL* model-checking in one-counter games. This closes a gap between 2ExpTime-completeness in finite-state games with LTL objectives and 3ExpTime-completeness in pushdown games with LTL objectives [LMS04]. The results are presented in Table 5.2, 5.3 and 5.3 together with related results from the literature. All results are completeness results.

The decidability results are, together with results from [MP15; CSW16], some of the first decidability results for ATL and ATL* model-checking in infinite-state games.

Outline

In Section 5.2 we introduce the technical setup for the chapter. In Section 5.3 model-checking algorithms based on model-checking games are presented. In Section 5.4 we provide lower bounds matching the complexity of the algorithms from Section 5.3. Section 5.5 provides a summary.

5.2 Preliminaries

A *one-counter game* (OCG) is a particular kind of finitely representable infinite-state turn-based game. Such a game is represented by a finite game graph where each transition is labelled with an integer from the set $\{-1, 0, 1\}$ as well as a counter that can hold any non-negative value. When a transition labelled v is taken and the current counter value is c, the counter value changes to $c + v$. We require that transitions are only applicable when $c + v \geq 0$.

Definition 5.1 *A labelled one-counter game is a tuple* $\mathcal{G} = (S, \Pi, (S_j)_{j\in\Pi}, R, \mathrm{AP}, L)$ *where* S *is a finite set of states,* Π *is a finite set of players and* $S_j \subseteq S$ *for all* $j \in \Pi$. *We require* $\bigcup_{j\in\Pi} S_j = S$ *and* $S_i \cap S_j = \emptyset$ *for all* $i \neq j$. $R \subseteq S \times \{-1, 0, 1\} \times S$ *is a transition relation,* AP *is a finite set of propositions and* $L : S \to 2^{\mathrm{AP}}$ *is a labelling function.*

As all one-counter games in this chapter are labelled, we just write one-counter game instead of labelled one-counter game. An OCG $\mathcal{G} = (S, \Pi, (S_j)_{j\in\Pi}, R, \mathrm{AP}, L)$ is a finite representation of the labelled infinite-state turn-based game $\mathcal{H} = (S \times \mathbb{N}, \Pi, (S_j \times \mathbb{N})_{j\in\Pi}, R', \mathrm{AP}, L')$ where $(s, v, s') \in R$ if and only if $((s, w), (s', w+v)) \in R'$ for all $w \in \mathbb{N}$ such that $w + v \in \mathbb{N}$. Further, $L'(s, v) = L(s)$ for all $s \in S$ and $v \in \mathbb{N}$.

We call $\mathcal{H}$ the turn-based game induced by $\mathcal{G}$. For OCGs we refer to the elements of $S \times \mathbb{N}$ as configurations and elements of S as states even though $S \times \mathbb{N}$ technically is the set of states of the induced turn-based game. Note that every configuration with a given state s has the same labelling. Strategies, plays and histories are defined in the same way for $\mathcal{G}$ and $\mathcal{H}$.

We extend one-counter games so arbitrary integer weights are allowed and such that transitions are still disabled if they would make the counter go negative. Such games are called *succinct one-counter games* (SOCGs). The special cases of OCGs and SOCGs where $|\Pi| = 1$ are called *one-counter processes* (OCPs) and *succinct one-counter processes* (SOCPs) respectively. In these cases we omit Π and $(S_j)_{j\in\Pi}$ from the definition.

By a *one-counter parity game* (OCPG) we mean a one-counter game with two players and a parity winning condition. It was shown in [Ser06] that the winner can be determined in an OCPG in PSPACE by reducing to the emptiness problem for alternating two-way parity automata [Var98].

Proposition 5.2 *Determining the winner in OCPGs is in* PSPACE.

When interpreting formulas of alternating-time temporal logics in an OCG/SOCG $\mathcal{G}$ with induced turn-based game $\mathcal{H}$ we use the notation $\mathcal{G}, s, i \models \Phi$ to be equivalent to $\mathcal{H}, (s, i) \models \Phi$ for every $s \in S, i \in \mathbb{N}$ and ATL* state formula Φ. Likewise we use $\mathcal{G}, (s_0, v_0)(s_1, v_1)... \models \Phi$ to be equivalent to $\mathcal{H}, (s_0, v_0)(s_1, v_1)... \models \Phi$ for every play $(s_0, v_0)(s_1, v_1)...$ in $\mathcal{G}$ and every ATL* path formula Φ.

As an extension of the logics introduced in Section 2.2 we consider fragments of the quantitative alternating-time temporal logics QATL and QATL* [BG13]. These

logics extend ATL and ATL* with state formulas of the form $r \bowtie c$ where $c \in \mathbb{Z}$, $\bowtie \in \{\leq, <, =, >, \geq, \equiv_k\}$ and $k \in \mathbb{N}$. This type of formula is called a counter constraint and is interpreted such that e.g. $r \leq 5$ is true if the current counter value is at most 5 whereas $r \equiv_4 3$ is true if the current counter value is equivalent to 3 modulo 4.

Formally, the semantics is given by $\mathcal{G}, s, i \models r \bowtie c$ iff $i \bowtie c$. The extension of $\mathcal{L} \in \{\text{LTL}, \text{CTL}, \text{CTL}^*\}$ with counter constraints is called Q$\mathcal{L}$ as for QATL*.

In this chapter we focus on the model-checking problem. That is to decide, given an OCG/SOCG $\mathcal{G}$, a state s in $\mathcal{G}$, a natural number i and a state formula φ whether $\mathcal{G}, s, i \models \varphi$. When model-checking non-succinct models, the initial counter value, edge weights and integers in the formula are assumed to be input in unary. For succinct models they are input in binary.

5.3 Model-checking algorithms

When model-checking branching and alternating-time temporal logics in finite-state systems, the standard approach is to process the state subformulas from the innermost to the outermost, at each step labelling states where the subformula is true. This approach does not work directly in our setting since the state space is infinite. We therefore take a different route and develop a model-checking game in which we can avoid explicitly labelling the configurations in which a subformula is true. This approach gives us optimal complexity in all cases considered and also allows us to extend to quantitative constraints in a natural way. We first present the approach for ATL and afterwards adapt it to ATL* and CTL* by combining it with automata on infinite words. Finally it is shown how to handle counter constraints.

5.3.1 A model-checking game for ATL

We convert the model-checking problem asking whether $\mathcal{G}, s_0, i \models \varphi$ for an ATL formula φ and OCG $\mathcal{G} = (S, \Pi, (S_j)_{j \in \Pi}, R, \text{AP}, L)$ to a model-checking game $\mathcal{H}_{\mathcal{G}, s_0, i}(\varphi)$ between two players 0 and 1 that are trying to respectively verify and falsify the formula. The construction is done so player 0 has a winning strategy in $\mathcal{H}_{\mathcal{G}, s_0, i}(\varphi)$ if and only if $\mathcal{G}, s_0, i \models \varphi$ and is done inductively on the structure of φ. For a given ATL formula, a given OCG $\mathcal{G}$ and a given state s in $\mathcal{G}$ we define a characteristic OCG $\mathcal{H}_{\mathcal{G}, s}(\varphi)$. Note that the initial counter value is not present in the construction yet. When illustrating the games, the circle states are controlled by player 0, square states are controlled by player 1 and diamond states can be both depending on the game. States with color 1 are filled and states with color 2 are not. Player 0 wins the game if the greatest color that appears infinitely often during the play is even, otherwise player 1 wins the game. The edges are labelled with counter updates, but 0-labels are omitted. The construction is done as follows.

- $\mathcal{H}_{\mathcal{G}, s}(p)$: There are two cases which are illustrated in Figure 5.5 to the left.
- $\mathcal{H}_{\mathcal{G}, s}(\varphi_1 \vee \varphi_2)$: The game is shown in Figure 5.5 in the middle.

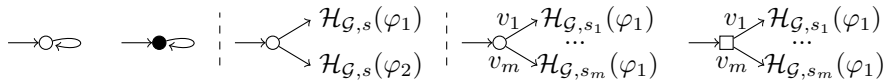


Figure 5.5: $\mathcal{H}_{\mathcal{G},s}(p)$ to the left for $p \in L(s)$ and $p \notin L(s)$. $\mathcal{H}_{\mathcal{G},s}(\varphi_1 \vee \varphi_2)$ is in the middle. $\mathcal{H}_{\mathcal{G},s}(\langle\!\langle A \rangle\!\rangle \mathbf{X}\varphi_1)$ to the right in the cases where $s \in \bigcup_{j\in A} S_j$ and where $s \notin \bigcup_{j\in A} S_j$. Circle states are controlled by player 0 and square states are controlled by player 1.

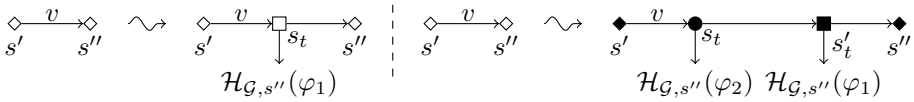


Figure 5.6: $\mathcal{H}_{\mathcal{G},s}(\langle\!\langle A \rangle\!\rangle \mathbf{G}\varphi_1)$ and $\mathcal{H}_{\mathcal{G},s}(\langle\!\langle A \rangle\!\rangle \varphi_1 \mathbf{U} \varphi_2)$ are obtained by updating each transition in $\mathcal{G}$ as shown to the left and right respectively.

- $\mathcal{H}_{\mathcal{G},s}(\neg\varphi_1)$: The game is constructed from $\mathcal{H}_{\mathcal{G},s}(\varphi_1)$ by interchanging circle states and square states and either adding or subtracting 1 to/from all colors.

- $\mathcal{H}_{\mathcal{G},s}(\langle\!\langle A \rangle\!\rangle \mathbf{X}\varphi_1)$: Let $R(s) = \{(s, v, s') \in R\} = \{(s, v_1, s_1), ..., (s, v_m, s_m)\}$. There are two cases to consider. One when $s \in S_j$ for some $j \in A$ and one when $s \notin S_j$ for all $j \in A$. Both are illustrated in Figure 5.5 to the right.

- $\mathcal{H}_{\mathcal{G},s}(\langle\!\langle A \rangle\!\rangle \mathbf{G}\varphi_1)$: We let $\mathcal{H}_{\mathcal{G},s}(\langle\!\langle A \rangle\!\rangle \mathbf{G}\varphi_1)$ have the same structure as $\mathcal{G}$, but with a few differences. Player 0 controls all states that are in $\bigcup_{j\in A} S_j$ and player 1 controls the rest. Further, for each transition $t = (s', v, s'') \in R$ we add a state s_t controlled by player 1 between s' and s''. If the player controlling s' chooses transition t the play is taken to the state s_t from which player 1 can either choose to continue to s'' or to $\mathcal{H}_{\mathcal{G},s''}(\varphi_1)$. Every state in $\mathcal{H}_{\mathcal{G},s}(\langle\!\langle A \rangle\!\rangle \mathbf{G}\varphi_1)$ which is not part of $\mathcal{H}_{\mathcal{G},s''}(\varphi_1)$ has color 2. It is illustrated in Figure 5.6. The intuition is that player 1 can challenge and claim that φ_1 is not true in the current configuration. If he does so, player 0 must be able show that it is in fact true in order to win. In addition, such a module is added before the initial state s so player 1 can challenge the truth of φ_1 already in the initial state.

- $\mathcal{H}_{\mathcal{G},s}(\langle\!\langle A \rangle\!\rangle \varphi_1 \mathbf{U} \varphi_2)$: The game is similar to the case of $\langle\!\langle A \rangle\!\rangle \mathbf{G}$. The differences are that every state is colored 1 and for each transition $t = (s', v, s'') \in R$ we add two states s_t and s'_t controlled by player 0 and player 1 respectively with transitions to $\mathcal{H}_{\mathcal{G},s''}(\varphi_2)$ and $\mathcal{H}_{\mathcal{G},s''}(\varphi_1)$ respectively. It is illustrated in Figure 5.6. The intuition is similar, but here player 0 loses unless he can claim φ_2 is true at some point (and subsequently show this). In addition φ_1 cannot become false before this point, because then player 1 can claim φ_1 is false and win. As for the previous case, such a module is added before the initial state as well.

Finally, we define the game $\mathcal{H}_{\mathcal{G},s,i}(\varphi)$ from $\mathcal{H}_{\mathcal{G},s}(\varphi)$ and $i \in \mathbb{N}$ by adding an initial module such that the counter is increased to i before entering $\mathcal{H}_{\mathcal{G},s}(\varphi)$. We can now show the following by induction on the structure of the ATL formula.

Proposition 5.3 *For every OCG $\mathcal{G}$, state s in $\mathcal{G}$, $i \in \mathbb{N}$ and $\varphi \in$ ATL*

$$\mathcal{G}, s, i \models \varphi \text{ if and only if player 0 has a winning strategy in } \mathcal{H}_{\mathcal{G},s,i}(\varphi)$$

Proof. The proof is done for QATL formulas (and thus works for ATL as well). The proof is done by induction on the structure of φ. First, we consider the base case.

$\varphi = p$: In this case player 0 has a winning strategy in $\mathcal{H}_{\mathcal{G},s,i}(p)$ if and only if $p \in L(s)$ if and only if $\mathcal{G}, s, i \models p$.

Next, we consider the inductive cases.

$\varphi = \varphi_1 \vee \varphi_2$: Clearly, if player 0 has a winning strategy in $\mathcal{H}_{\mathcal{G},s,i}(\varphi_1)$ or in $\mathcal{H}_{\mathcal{G},s,i}(\varphi_2)$ then he has a winning strategy in $\mathcal{H}_{\mathcal{G},s,i}(\varphi_1 \vee \varphi_2)$ since he can choose which of the games to play and reuse the winning strategy. On other hand, if player 0 has a winning strategy in $\mathcal{H}_{\mathcal{G},s,i}(\varphi_1 \vee \varphi_2)$ then he is either winning in $\mathcal{H}_{\mathcal{G},s,i}(\varphi_1)$ or in $\mathcal{H}_{\mathcal{G},s,i}(\varphi_2)$ because he can reuse the strategy and be sure to win in at least one of these games. Then, by using the induction hypothesis we have that player 0 has a winning strategy in $\mathcal{H}_{\mathcal{G},s,i}(\varphi_1 \vee \varphi_2)$ if and only if he has a winning strategy in $\mathcal{H}_{\mathcal{G},s,i}(\varphi_1)$ or in $\mathcal{H}_{\mathcal{G},s,i}(\varphi_2)$ if and only if $\mathcal{G}, s, i \models \varphi_1$ or $\mathcal{G}, s, i \models \varphi_2$ if and only if $\mathcal{G}, s, i \models \varphi_1 \vee \varphi_2$.

$\varphi = \neg\varphi_1$: The construction essentially switches player 0 with player 1 when creating $\mathcal{H}_{\mathcal{G},s,i}(\neg\varphi_1)$ from $\mathcal{H}_{\mathcal{G},s,i}(\varphi_1)$. This means that player 0 has a winning strategy in $\mathcal{H}_{\mathcal{G},s,i}(\neg\varphi_1)$ if and only if player 1 has a winning strategy in $\mathcal{H}_{\mathcal{G},s,i}(\varphi_1)$. We have that one-counter games with parity conditions are determined [Ser06]. It follows that player 0 has a winning strategy in $\mathcal{H}_{\mathcal{G},s,i}(\neg\varphi_1)$ if and only if player 0 does not have a winning strategy in $\mathcal{H}_{\mathcal{G},s,i}(\varphi_1)$. Using the induction hypothesis this means that player 0 has a winning strategy in $\mathcal{H}_{\mathcal{G},s,i}(\neg\varphi_1)$ if and only if $\mathcal{G}, s, i \not\models \varphi_1$ if and only if $\mathcal{G}, s, i \models \neg\varphi_1$.

$\varphi = \langle\langle A \rangle\rangle \mathbf{X}\varphi_1$: There are two cases to consider. First, suppose $s \in S_j$ for some $j \in A$. Then player 0 has a winning strategy in $\mathcal{H}_{\mathcal{G},s,i}(\langle\langle A \rangle\rangle \mathbf{X}\varphi_1)$ if and only if there is a transition $(s, v, s') \in R$ with $v + i \geq 0$ such that player 0 has a winning strategy in $\mathcal{H}_{\mathcal{G},s',i+v}(\varphi_1)$ since parity objectives are prefix independent. Using the induction hypothesis, this is the case if and only if there is a transition $(s, v, s') \in R$ with $v + i \geq 0$ such that $\mathcal{G}, s', i + v \models \varphi_1$ which is the case if and only if $\mathcal{G}, s, i \models \langle\langle A \rangle\rangle \mathbf{X}\varphi_1$. For the case where $s \notin S_j$ for all $j \in A$ the proof is similar, but uses universal quantification over the transitions.

$\varphi = \langle\langle A \rangle\rangle \mathbf{G}\varphi_1$: The intuition of the construction is that player 0 controls the players in A and player 1 controls the players in $\Pi \setminus A$. At each configuration $(s', v) \in$

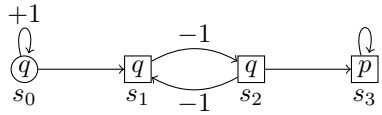


Figure 5.7: One-counter game $\mathcal{G}$.

$S \times \mathbb{N}$ of the game player 1 can challenge the truth value of φ_1 by going to $\mathcal{H}_{\mathcal{G},s',v}(\varphi_1)$ in which player 1 has a winning strategy if and only if φ_1 is indeed false in $\mathcal{G}, s', v$ by the induction hypothesis. If player 1 challenges at the wrong time or never challenges then player 0 can make sure to win.

More precisely, suppose player 0 has a winning strategy σ in $\mathcal{H}_{\mathcal{G},s,i}(\langle\langle A\rangle\rangle \mathbf{G}\varphi_1)$ then every possible play when player 0 plays according to σ either never goes into one of the modules $\mathcal{H}_{\mathcal{G},s'}(\varphi_1)$ or the play goes into one of the modules at some point and never returns. Since σ is a winning strategy for player 0, we have by the induction hypothesis that every pair $(s', v) \in S \times \mathbb{N}$ reachable when player 0 plays according to σ is such that $\mathcal{G}, s', v \models \varphi_1$, because otherwise σ would not be a winning strategy for player 0. If coalition A follows the same strategy σ adapted to $\mathcal{G}$ then the same state, value pairs are reachable. Since for all these reachable pairs (s', v) we have $\mathcal{G}, s', v \models \varphi_1$ this strategy is a witness that $\mathcal{G}, s, i \models \langle\langle A\rangle\rangle \mathbf{G}\varphi_1$.

On the other hand, suppose that coalition A can ensure $\mathbf{G}\varphi_1$ from (s, i) using strategy σ. Then in every reachable configuration (s', v) we have $\mathcal{G}, s', v \models \varphi_1$. From this we can generate a winning strategy for player 0 in $\mathcal{H}_{\mathcal{G},s,i}(\langle\langle A\rangle\rangle \mathbf{G}\varphi_1)$ that plays in the same way until (if ever) player 1 challenges and takes a transition to a module $\mathcal{H}_{\mathcal{G},s',v}(\varphi_1)$ for some (s', v). Since the same configurations can be reached before a challenge as when A plays according to σ, this means that player 0 can make sure to win in $\mathcal{H}_{\mathcal{G},s',v}(\varphi_1)$ by the induction hypothesis. Thus, if player 1 challenges player 0 can make sure to win and if player 1 never challenges player 0 also wins since all states reached have color 2. Thus, player 0 has a winning strategy in $\mathcal{H}_{\mathcal{G},s,i}(\langle\langle A\rangle\rangle \mathbf{G}\varphi_1)$.

$\varphi = \langle\langle A\rangle\rangle \varphi_1 \mathbf{U} \varphi_2$: The proof works as in the case above with some minor differences. In this case, player 0 needs to show that he can reach a configuration where φ_2 is true when controlling the players in A and therefore he loses if player 1 does not challenge incorrectly and he never reaches a module $\mathcal{H}_{\mathcal{G},s',v}(\varphi_2)$ such that $\mathcal{G}, s', v \models \varphi_2$. At the same time, he has to make sure that configurations (s', v) where $\mathcal{G}, s', v \not\models \varphi_1$ are not reached in an intermediate configuration since player 1 still has the ability to challenge, as in the previous case. Note that player 0 gets the chance to commit to showing that φ_2 is true in a given configuration before player 1 gets the chance to challenge the value of φ_1. This is due to the definition of the until operator that does not require φ_1 to be true at the point where φ_2 becomes true. We leave out the remaining details. □

Example 5.4 *Consider the OCG $\mathcal{G}$ in Figure 5.7 between player a and b. Player a controls circle states and player b controls square states. Let $\varphi = \langle\langle\{a\}\rangle\rangle q \boldsymbol{U} p$ be an* ATL *formula which specifies that player a can force the play to eventually reach a p state while only visiting q states along the way.*

We create the model-checking game $\mathcal{H}_{\mathcal{G},s_0,2}(\varphi)$ in Figure 5.8. It can be used to decide whether player a can force the play to eventually reach a p state in $\mathcal{G}$ while only visiting q states along the way given that the play begins in s_0 with an initial counter value of 2*.*

In the model-checking game recall that player 0 *controls circle states and player* 1 *controls square states. Filled states are colored* 1 *and the non-filled are colored* 2*. If player* 0 *can force the play to infinitely often reach a non-filled state in this one-counter parity game then we have $\mathcal{G}, s_0, 2 \models \varphi$.*

In general, such one-counter parity games can be solved in polynomial space by using a reduction to the emptiness problem for two-way alternating parity automata on words. For details, see [Ser06].

In this example, we can argue that player 0 *has a winning strategy as follows. Consider a strategy which ensures that the value of the counter is odd when s_1 is reached and which takes the play to the module $\mathcal{H}_{\mathcal{G},s_3}(p)$ from the challenge module between s_2 and s_3. Indeed, if the counter value is odd when s_1 is reached then player* 1 *cannot avoid taking the play to the challenge module between s_2 and s_3 (or making a losing challenge earlier). If he keeps avoiding this the counter value will eventually be* 0 *when s_2 is reached and he has no other choice than moving the play towards s_3. In this case player* 0 *can take the play to $\mathcal{H}_{\mathcal{G},s_3}(p)$ and win.*

Note that this strategy resembles the strategy needed for player a in $\mathcal{G}$ in order to force that $q \boldsymbol{U} p$ is true.

We can create a model-checking game for ATL in SOCGs as for OCGs and obtain a model-checking game which is a succinct OCPG. This can be transformed into an OCPG that is exponentially larger. It is done by replacing each transition with weight v with a path that has $|v|$ transitions and adding small gadgets so a player loses if he takes a transition with value $-w$ for $w \in \mathbb{N}$ when the current counter value $r < w$. The exponential blowup is due to weights being input in binary. By Proposition 5.2 this gives upper bounds for ATL model-checking. Matching lower bounds follow from PSPACE-hardness [GL13] and EXPSPACE-hardness [Göl+10] of data complexity of CTL in OCPs and SOCPs respectively.

Theorem 5.5 *The data complexity and combined complexity of model-checking* ATL *are* PSPACE*-complete for OCGs and* EXPSPACE*-complete for SOCGs.*

5.3.2 Adapting the construction to ATL*

As for ATL we rely on the approach of a model-checking game when model-checking ATL*. However, due to the extended possibilities of nesting we do not handle temporal operators directly as for ATL. Instead, we resort to translation of LTL formu-

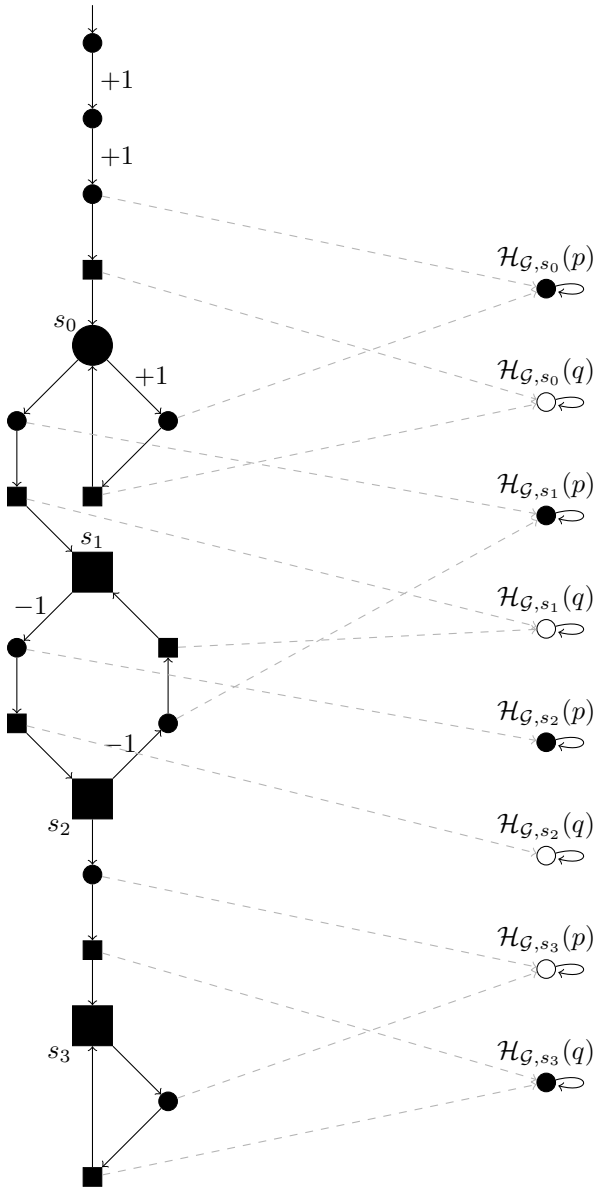


Figure 5.8: The model-checking game $\mathcal{H}_{\mathcal{G},s_0,2}(\langle\!\langle\{a\}\rangle\!\rangle q\mathbf{U}p)$.

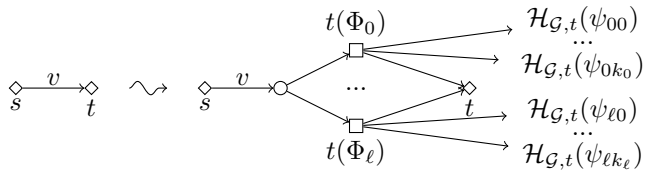


Figure 5.9: $\mathcal{H}_{\mathcal{G},s}(\langle\langle A\rangle\rangle\varphi)$ is obtained by updating each transition as shown in the figure.

las into deterministic parity automata (DPA) which are combined with the model-checking games. This gives us model-checking games which are one-counter parity games as for ATL, but with a doubly-exponential blowup.

Let $\mathcal{G} = (S, \Pi, (S_j)_{j\in\Pi}, R, AP, L)$ be an OCG, $s_0 \in S$, $i \in \mathbb{N}$ and φ be an ATL* state formula. The algorithm to decide whether $\mathcal{G}, s_0, i \models \varphi$ follows along the same lines as our algorithm for ATL. That is, we construct a model-checking game $\mathcal{H}_{\mathcal{G},s_0,i}(\varphi)$ between player 0 and player 1 such that player 0 has a winning strategy in $\mathcal{H}_{\mathcal{G},s_0,i}(\varphi)$ if and only if $\mathcal{G}, s_0, i \models \varphi$. The construction is done inductively on the structure of φ. For each state $s \in S$ and state formula φ we define a characteristic OCG $\mathcal{H}_{\mathcal{G},s}(\varphi)$.

For formulas of the form p, $\neg\varphi_1$ and $\varphi_1 \vee \varphi_2$ the construction is as for ATL assuming in the inductive cases that $\mathcal{H}_{\mathcal{G},s}(\varphi_1)$ and $\mathcal{H}_{\mathcal{G},s}(\varphi_2)$ have already been defined. The interesting case is $\varphi = \langle\langle A\rangle\rangle\varphi_1$. Here, let $\psi_1, ..., \psi_m$ be the outermost proper state subformulas of φ_1. Let $P = \{p_1, ..., p_m\}$ be fresh propositions and let $f(\varphi_1) = \varphi_1[\psi_1 \mapsto p_1, ..., \psi_m \mapsto p_m]$ be the formula obtained from φ_1 by replacing the outermost proper state subformulas with the corresponding fresh propositions. Let $\mathrm{AP}' = \mathrm{AP} \cup P$. Now, $f(\varphi_1)$ is an LTL formula over AP'. We can therefore construct a DPA $\mathcal{A}_{f(\varphi_1)}$ with input alphabet $2^{\mathrm{AP}'}$ such that the language $L(\mathcal{A}_{f(\varphi_1)})$ of the automaton is exactly the set of linear models of $f(\varphi_1)$. The number of states of the DPA can be bounded by $2^{2^{O(|f(\varphi_1)|)}}$ and the number of colors of the DPA can be bounded by $2^{O(|f(\varphi_1)|)}$ [Pit07].

The game $\mathcal{H}_{\mathcal{G},s}(\varphi)$ is now constructed with the same structure as $\mathcal{G}$ where player 0 controls the states for players in A and player 1 controls the states for players in $\Pi \setminus A$. However, we need to deal with truth values of the formulas $\psi_1, ..., \psi_m$ which can in general not be labelled to states in $\mathcal{G}$ since they depend both on the current state and counter value. Therefore we change the structure to obtain $\mathcal{H}_{\mathcal{G},s}(\varphi)$: For each transition $(s, v, t) \in R$ we embed a module as shown in Figure 5.9. Here, $2^{\mathrm{AP}'} = \{\Phi_0, ..., \Phi_\ell\}$ and for each $0 \le j \le \ell$ we let $\{\psi_{j0}, ..., \psi_{jk_j}\} = \{\psi_i \mid p_i \in \Phi_j\} \cup \{\neg\psi_i \mid p_i \notin \Phi_j\}$. Such a module is added before the initial state as well.

The idea is that when a transition is taken from (s, w) to $(t, w + v)$, player 0 must specify which of the propositions $p_1, ..., p_m$ are true in $(t, w + v)$. This is done by picking one of the subsets Φ_j (which is the set of propositions that are true in state $t(\Phi_j)$). Then, to make sure that player 0 does not cheat, player 1 has the opportunity

to challenge any of the truth values of the propositions specified by player 0. If player 1 challenges, the play never returns again. Thus, if player 1 challenges incorrectly, player 0 can make sure to win the game. However, if player 1 challenges correctly then player 1 can be sure to win the game. If player 0 has a winning strategy, then it consists in choosing the correct values of the propositions at each step. If player 0 does choose correctly and player 1 never challenges, the winner of the game should be determined based on whether the LTL property specified by $f(\varphi_1)$ is satisfied during the play. We handle this by labelling $t(\Phi_j)$ with the propositions in Φ_j. Further, since every step of the game is divided into three steps (the original step, the specification by player 0 and the challenge opportunity for player 1) we alter $\mathcal{A}_{f(\varphi_1)}$ such that it only takes a transition every third step. This simply increases its size by a factor 3. We then perform a product of the game with the updated parity automaton to obtain the parity game $\mathcal{H}_{\mathcal{G},s}(\langle\langle A\rangle\rangle\varphi_1)$. It is important to note that the product with the automaton is not performed on the challenge modules (which are already colored), but only with states in the main module. This keeps the size of the game doubly-exponential in the size of the formula. It is possible to prove the following by induction on the structure of the formula. All cases except for $\langle\langle A\rangle\rangle\varphi$ are as for ATL. In the last case the proof follows the intuition outlined above.

Proposition 5.6 *For every OCG $\mathcal{G}$, state s in $\mathcal{G}$, $i \in \mathbb{N}$ and $\varphi \in \mathrm{ATL}^*$*

$$\mathcal{G}, s, i \models \varphi \text{ if and only if 0 has a winning strategy in } \mathcal{H}_{\mathcal{G},s,i}(\varphi)$$

Proof. The proof is done by induction on the structure of φ. The base cases as well as boolean combinations are omitted since they work as for ATL. The interesting case is $\varphi = \langle\langle A\rangle\rangle\varphi_1$.

Suppose first that $\mathcal{G}, s, i \models \langle\langle A\rangle\rangle\varphi_1$. Then coalition A has a winning strategy σ in $\mathcal{G}$. From this, we generate a strategy σ' for player 0 in $\mathcal{H}_{\mathcal{G},s,i}(\langle\langle A\rangle\rangle\varphi_1)$ that consists in never cheating when specifying values of atomic formulas and choosing transitions according to what σ would have done in $\mathcal{G}$. Then, if player 1 challenges at some point, player 0 can be sure to win by the induction hypothesis since he never cheats. If player 1 never challenges (or, until he challenges), player 0 simply mimics the collective winning strategy σ of coalition A in $\mathcal{G}$ from (s, i). This ensures that he wins in the parity game due to the definition of the parity condition from the parity automaton corresponding to $f(\varphi_1)$.

Suppose on the other hand that in $\mathcal{H}_{\mathcal{G},s,i}(\langle\langle A\rangle\rangle\varphi_1)$ player 0 has a winning strategy σ. Then σ never cheats when specifying values of propositions, because then player 1 could win according to the induction hypothesis. Define a strategy σ' for coalition A in $\mathcal{G}$ that plays like σ in the part of $\mathcal{H}_{\mathcal{G},s,i}(\langle\langle A\rangle\rangle\varphi_1)$ where no challenge has occured. σ' is winning for A with condition φ_1 in $\mathcal{G}$ due to the definition of $\mathcal{H}_{\mathcal{G},s,i}(\langle\langle A\rangle\rangle\varphi_1)$ using the automaton $\mathcal{A}_{f(\varphi_1)}$. □

The size of the model-checking game is doubly-exponential in the size of the formula for both OCGs and SOCGs. Indeed, we extend the technique to SOCGs as

in the case of ATL. However, with respect to complexity, the blowup caused by the binary representation of edge weights only matters when the formula is fixed since the game is already doubly-exponential when the input formula is a parameter. Using Proposition 5.2 we get upper bounds on complexity of model-checking ATL*. For data complexity the lower bounds follow from data complexity of CTL. 2ExpSpace-hardness for combined complexity is proved in Section 5.4.

Theorem 5.7 *The data complexity for* ATL* *model-checking is* PSpace*-complete and* ExpSpace*-complete for OCGs and SOCGs respectively. The combined complexity is* 2ExpSpace*-complete for both OCGs and SOCGs.*

5.3.3 Adapting the construction to CTL*

While the model-checking game for ATL* works immediately for CTL*, the doubly-exponential size can improved. The reason is that when the model is not alternating, we can use non-deterministic Büchi automata (NBAs) for path subformulas instead of DPAs. To handle a formula of the form $\varphi = \mathbf{E}\varphi_1$ we do as for $\langle\langle A\rangle\rangle\varphi_1$ in the previous section except that the automaton $\mathcal{A}_{f(\varphi)}$ is now an NBA with $2^{O(|f(\varphi)|)}$ states [WVS83]. Further, we need to handle the fact that the automaton is non-deterministic and therefore can have several legal transitions.

The game is simply adjusted by letting player 0 choose the transitions in the original system as well as of the automaton in each step of the main module in $\mathcal{H}_{\mathcal{G},s}(\varphi)$. This works as he just needs to show that there exists a path in the OCP along with an accepting run of the automaton in order to be sure to win. If one such exists he can show it by playing this path as well as playing the corresponding run of the automaton. The only power that player 1 has in the main module is the possibility to challenge the values for subformulas proposed by player 0. Thus, if player 0 proposes an incorrect valuation or plays a path that is not accepting then player 1 can make sure to win.

Note that this construction makes the model-checking game exponential in the size of the formula. Again, Proposition 5.2 provides us with upper bounds. A matching ExpSpace lower bounds for combined complexity of model-checking CTL* in OCPs is shown in Section 5.4.

Theorem 5.8 *The combined complexity of model-checking* CTL* *for both OCPs and SOCPs is* ExpSpace*-complete.*

The PSpace-completeness and ExpSpace-completeness of data complexity of CTL* model-checking in OCPs and SOCPs follow immediately from results in the literature. Indeed, lower bounds are inherited from CTL model-checking results [Göl+10; GL13] and upper bounds can be derived from μ-calculus results [Ser06] as for every CTL* formula there is an equivalent μ-calculus formula. However, note that in these cases our construction above provides the matching upper bounds as well without resorting to a translation from CTL* formulas to μ-calculus formulas.

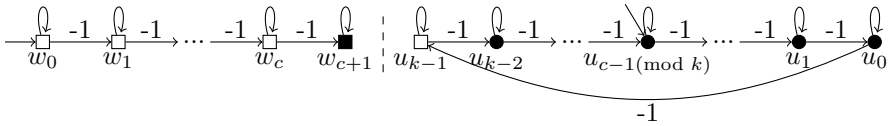


Figure 5.10: $\mathcal{H}_{\mathcal{G},s}(r \leq c)$ to the left and $\mathcal{H}_{\mathcal{G},s}(r \equiv_k c)$ to the right.

5.3.4 Adding counter constraints

The model-checking game can be extended to handle *counter constraints* by creating characteristic games $\mathcal{H}_{\mathcal{G},s}(r \bowtie c)$ for atomic formulas of the form $r \bowtie c$ with $\bowtie \in \{\leq, <, =, >, \geq, \equiv_k\}$ where $k \in \mathbb{N}$ and $c \in \mathbb{Z}$. As examples, see $\mathcal{H}_{\mathcal{G},s}(r \leq c)$ and $\mathcal{H}_{\mathcal{G},s}(r \equiv_k c)$ illustrated in Figure 5.10. Using similar constructions we can handle the other cases as well. Note that adding these constraints to the logics does not increase the complexity of the algorithms in any of the cases considered.

Having added counter constraints it is quite easy to see that model-checking of CTL* in one-counter processes with zero-tests can be done without increasing the complexity. This can be accomplished by updating the CTL* formula to only consider paths that are legal according to the zero-tests. By reusing a technique from [BGM14] we can also handle systems where the counter value is allowed to be negative. Similar constructions can be made for ATL and ATL* model-checking by using alternation between players to check that no player can choose a transition that he is not allowed to choose without losing.

5.4 Lower bounds

In this section we provide lower bounds for combined complexity of CTL* in OCPs and combined complexity of ATL* in OCGs.

5.4.1 Lower bound for CTL*

For the combined complexity of CTL* in OCPs an ExpSpace lower bound does not follow immediately from results for CTL since the combined complexity of CTL is PSpace-complete in OCPs. To show this lower bound we do a reduction from the data complexity of CTL in SOCPs which requires some more work.

Proposition 5.9 *The combined complexity of model-checking* CTL* *in OCPs is* ExpSpace*-hard.*

Proof. We do the proof by a reduction from the model-checking problem for a fixed CTL formula in an SOCP. That is, given a CTL formula φ, an SOCP $\mathcal{G} = (S, R, \mathrm{AP}, L)$, an initial state s_0 and value $v \in \mathbb{N}$ we want to construct a CTL* formula

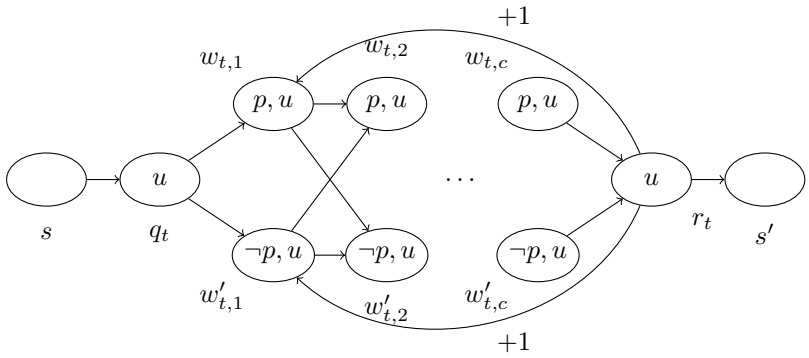


Figure 5.11: Module $\mathcal{G}'_t$ for transition $t = (s, v, s')$ with $v \geq 0$.

φ' and an OCP $\mathcal{G}' = (S', R', \mathrm{AP}', L')$ such that

$$\mathcal{G}, s_0, 0 \models \varphi \text{ if and only if } \mathcal{G}', s_0, 0 \models \varphi'$$

The challenge of the construction is that $\mathcal{G}'$ can only have transitions with weights in $\{-1, 0, 1\}$. In order to accomplish this without blowing up the state space exponentially we add a module for each transition $(s, v, s') \in R$ designed to simulate adding v to the counter value.

We explain the construction for $v \geq 0$ first. Let $c \in \mathbb{N}$ be the smallest number such that $2^c > w$ for every integer w that is the label of a transition in $\mathcal{G}$. Then every edge weight can be represented using c bits. Now, to obtain $\mathcal{G}'$ we do as follows. For every transition $t = (s, v, s') \in R$ with $v \geq 0$ we replace t with a module $\mathcal{G}'_t$ as shown in Figure 5.11.

In this module it is possible to increase the counter by any non-negative value before completing the transition from s to s'. It is even possible to stay in the module between s and s' forever (unlike in $\mathcal{G}$). Note also that v does not appear in the module at all. We will use the CTL* formula to focus on paths that behave as the transition (s, v, s') in $\mathcal{G}$ when passing through this module.

We suppose that all new states in $\mathcal{G}'$ are labelled with the proposition u and all states from $\mathcal{G}$ are not. The idea is that this proposition expresses that an update of the counter value is currently being performed. Further, for each state s in $\mathcal{G}'$ there is a special proposition s which is true exactly in that state.

A similar module can be created for $v < 0$ where the $+1$ transitions are changed to -1 transitions.

Observe that the resulting structure $\mathcal{G}'$ is an OCP since there are only transition weights in $\{-1, 0, 1\}$ and further, the reduction is polynomial in the number of bits used to represent the integer weights in $\mathcal{G}$. We next propose a function f mapping CTL formulas to CTL* formulas such that $\mathcal{G}, s_0, 0 \models \varphi$ if and only if $\mathcal{G}', s_0, 0 \models f(\varphi)$

for every CTL formula φ. First, let

$$\psi_{count} = \bigwedge_{t=(s,v,s')\in R} \mathbf{G}(\psi_1(t) \wedge \psi_2(t) \wedge \psi_3(t) \wedge \psi_4(t))$$

The intuition is that the path formula ψ_{count} is true along a path in $\mathcal{G}'$ if every subpath through a module $\mathcal{G}'_t$ with $t = (s, v, s')$ updates the counter by adding exactly v before reaching s'. Thus, the formula is true along a path ρ' in $\mathcal{G}'$ if and only if ρ' corresponds to a path in $\mathcal{G}$ (where each counter update of value v takes $(|v|+1)\cdot(c+1)+2$ steps in ρ'). The reason that we need to enforce counter updates in this way is to avoid blowing up the size of the OCP $\mathcal{G}'$. This is important since edge weights are exponentially large in the input in $\mathcal{G}$.

The truth value of proposition p is used as the bit-representation of the value that the counter has already been updated with where the least significant bit occurs first. The intuitive meaning of the subformulas are as follows for each transition $t = (s, a, s')$.

- $\psi_1(t)$: When the module $\mathcal{G}'_t$ is entered, the path goes through only $\neg p$ states until r_t since the counter has initially been updated with 0.
- $\psi_2(t)$: The counter value must be updated by one every time r_t is reached except the last time before the module is left.
- $\psi_3(t)$: The path must exit the module before the counter has been updated 2^c times.
- $\psi_4(t)$: If the path exits the module, the counter must have been updated exactly $|v|$ times.

The subformulas are defined in LTL as below, where $\mathbf{X}^j$ is defined inductively by $\mathbf{X}^1 = \mathbf{X}$ and $\mathbf{X}^j = \mathbf{X}^{j-1}\mathbf{X}$ for $j > 1$.

$$\psi_1(t) = q_t \rightarrow \left(\bigwedge_{i=1}^{c} \mathbf{X}^i \neg p\right)$$

$$\psi_2(t) = \left[(q_t \vee r_t) \rightarrow \bigwedge_{i=1}^{c} \left(\bigwedge_{j=1}^{i-1} \mathbf{X}^j p \leftrightarrow \left(\mathbf{X}^{i+c+1} p \leftrightarrow \mathbf{X}^i \neg p\right)\right)\right] \mathbf{U}\left[\mathbf{X}(\neg r_t \mathbf{U}(r_t \wedge \mathbf{X} s'))\right]$$

$$\psi_3(t) = r_t \rightarrow \left(\bigvee_{i=1}^{c} \mathbf{X}^i p\right) \mathbf{U}\mathbf{X} s'$$

Finally, for each transition $t = (s, v, s')$ let $v' = |v|$ and let $b_1, ..., b_c$ be the c-bit representation of v' where b_1 is the least significant bit. Let B_t be the set of indices

j such that $b_j = 1$ and C_t be the set of indices j such that $b_j = 0$. Now, define

$$\psi_4(t) = (q_t \vee r_t) \wedge \mathbf{X}(\neg r_t \mathbf{U}(r_t \wedge \mathbf{X}s')) \rightarrow \bigwedge_{j \in B_t} \mathbf{X}^j p \wedge \bigwedge_{j \in C_t} \mathbf{X}^j \neg p$$

We now define f inductively on the structure of a CTL formula. Thus, for every proposition q from the labelling of $\mathcal{G}$ and all CTL formulas φ_1, φ_2

$$f(q) = q$$

$$f(\varphi_1 \vee \varphi_2) = f(\varphi_1) \vee f(\varphi_2)$$

$$f(\neg\varphi_1) = \neg f(\varphi_1)$$

$$f(\mathbf{EG}\varphi_1) = \mathbf{E}(\psi_{count} \wedge \mathbf{G}(u \vee (\neg u \wedge f(\varphi_1))))$$

$$f(\mathbf{E}\varphi_1 \mathbf{U} \varphi_2) = \mathbf{E}(\psi_{count} \wedge (u \vee (\neg u \wedge f(\varphi_1))) \mathbf{U} (\neg u \wedge f(\varphi_2)))$$

$$f(\mathbf{EX}\varphi_1) = \mathbf{E}(\psi_{count} \wedge (\mathbf{X}u \mathbf{U} \mathbf{X}(\neg u \wedge f(\varphi_1))))$$

Now, by induction on the structure of the CTL formula φ we show that for every state $s_0 \in S$ and every $v \in \mathbb{N}$ we have $\mathcal{G}, s_0, v \models \varphi$ if and only if $\mathcal{G}', s_0, v \models f(\varphi)$.

$\varphi = q$: For the base case it is true immediately since for every state $s \in S$ and $q \in \mathrm{AP}$ we have $q \in L(s)$ iff $q \in L'(s)$.

Assume as induction hypothesis that the claim is true for all proper subformulas of φ. Then we have the following cases.

$\varphi = \varphi_1 \vee \varphi_2$: By the induction hypothesis we have

$\mathcal{G}, s_0, v \models \varphi_1 \vee \varphi_2$

iff $\mathcal{G}, s_0, v \models \varphi_1$ or $\mathcal{G}, s_0, v \models \varphi_2$

iff $\mathcal{G}', s_0, v \models f(\varphi_1)$ or $\mathcal{G}', s_0, v \models f(\varphi_2)$

iff $\mathcal{G}', s_0, v \models f(\varphi_1) \vee f(\varphi_2)$

iff $\mathcal{G}', s_0, v \models f(\varphi_1 \vee \varphi_2)$

$\varphi = \neg\varphi_1$: By the induction hypothesis we have

$\mathcal{G}, s_0, v \models \neg\varphi_1$

iff $\mathcal{G}, s_0, v \not\models \varphi_1$

iff $\mathcal{G}', s_0, v \not\models f(\varphi_1)$

iff $\mathcal{G}', s_0, v \models \neg f(\varphi_1)$

iff $\mathcal{G}', s_0, v \models f(\neg\varphi_1)$

$\varphi = \mathbf{EG}\varphi_1$: Suppose first that $\mathcal{G}, s_0, v \models \mathbf{EG}\varphi_1$. Then there exists a path ρ in $\mathcal{G}$ from s_0 such that φ_1 is true in every configuration of ρ. There exists a corresponding path in $\mathcal{G}'$ which passes through the same configurations as ρ but with update modules in between. Let ρ' be the unique path with this property and such that ρ' also satisfies ψ_{count}. Now, in every configuration of ρ' either u is true or false. When u is false the current configuration of ρ' is also a configuration of ρ since ρ' satisfies ψ_{count}. By the induction hypothesis, each such configuration of ρ' thus satisfies $f(\varphi_1)$. This means that we have $\mathcal{G}', s_0, v \models \mathbf{E}(\psi_{count} \wedge \mathbf{G}(u \vee (\neg u \wedge f(\varphi_1)))) = f(\mathbf{EG}\varphi_1)$. For the other direction the argument is similar, showing that if there is a path ρ' in $\mathcal{G}'$ from s_0, v satisfying $\psi_{count} \wedge \mathbf{G}(u \vee (\neg u \wedge f(\varphi_1)))$ then there exists a corresponding path ρ in $\mathcal{G}$ from s_0, v satisfying $\mathbf{EG}\varphi_1$.

The proofs for $\mathbf{E}(\varphi_1 \mathbf{U} \varphi_2)$ and $\mathbf{EX}\varphi_1$ are done similarly to the case above. Thus, we have that $\mathcal{G}, s_0, v \models \varphi$ iff $\mathcal{G}', s_0, v \models f(\varphi)$. In particular this is the case for $v = 0$. Since the model-checking problem for CTL in SOCPs can be easily reduced to the same problem where the initial value $v = 0$ then we have that the model-checking problem for CTL* in OCPs is ExpSpace-hard since the constructions of $\mathcal{G}'$ and $f(\varphi)$ above are polynomial. Note that the CTL* formula is not fixed even if the CTL formula is and therefore the result does not apply for the data complexity of CTL*. □

5.4.2 Lower bound for ATL*

For combined complexity of ATL* we can show that 2ExpSpace is a tight lower bound by a reduction from the word acceptance problem of a doubly-exponential space deterministic Turing machine.

Let $\mathcal{T} = (Q, q_0, \Sigma, \delta, q_F)$ be a deterministic Turing machine that uses at most $2^{2^{|w|^k}}$ tape cells on input w where k is a constant and $|w|$ is the number of symbols in w. Here, Q is a finite set of control states, $q_0 \in Q$ is the initial control state. $\Sigma = \{0, 1, \flat, a, r\}$ is the tape alphabet containing the blank symbol $\flat$ and special symbols a and r such that $\mathcal{T}$ accepts immediately if it reads a and rejects immediately if it reads r. Recall that $\delta : Q \times \Sigma \to Q \times \Sigma \times \{-1, +1\}$ is the transition function and

$q_F \in Q$ is the accepting state. If $\delta(q,a) = (q', a', x)$ we write $\delta_1(q,a) = q', \delta_2(q,a) = a'$ and $\delta_3(q,a) = x$. Let $\Sigma_I = \Sigma \setminus \{\flat\}$.

Now, let $w = w_1 ... w_{|w|} \in \Sigma_I^*$ be an input word. From this we construct an OCG $\mathcal{G}$, an initial state s_0 and an ATL* formula Φ all with size polynomial in $n = |w|^k$ and $|\mathcal{T}|$ such that $\mathcal{T}$ accepts w if and only if $\mathcal{G}, (s_0, 0) \models \Phi$.

An intermediate reduction

We use an intermediate step in the reduction for simplicity. This is done by considering an OCG $\mathcal{H} = (S', \{0,1\}, (S_0', S_1'), R')$ with two players player 0 and player 1 and an initial state s_0' such that player 0 can force the play to reach s_F' if and only if $\mathcal{T}$ accepts w. However, the size of the set S' of states will be doubly-exponential in n. The idea of this construction resembles a reduction from the word acceptance problem for polynomial-space Turing machines to the emptiness problem for alternating finite automata with a singleton alphabet used in [JS07].

Afterwards we will reduce this to model-checking of the ATL* formula Φ in $\mathcal{G}$ where $|S|$ is polynomial in n. This reduction can be performed by considering a more involved formula. We will use a technique similar to [Kup+00; Boz07; BMP10] to simulate a 2^n-bit counter by using LTL properties and alternation between the players. This is the main trick to keep the state-space of $\mathcal{G}$ small.

We start with some notation. We assume that $\mathcal{T}$ uses the tape cells numbered $1, ..., 2^{2^n}$ and that the tape head points to position 1 initially. In addition, suppose for ease of arguments that there are two extra tape cells numbered 0 and $2^{2^n} + 1$ such that $\mathcal{T}$ immediately accepts if the tape head reaches cell 0 or cell $2^{2^n} + 1$. That is, cell 0 and $2^{2^n} + 1$ holds the symbol a initially. Further, assume without loss of generality that if $\mathcal{T}$ halts it always does so with the tape head pointing to cell 1 that contains the symbol a. Since $\mathcal{T}$ is deterministic it has a unique (finite or infinite) run on the word w which is a sequence $C_0^w C_1^w ...$ of configurations. Let $\Delta = \Sigma \cup (Q \times \Sigma)$. Then each configuration C_i^w is a sequence in $\Delta^{2^{2^n}+2}$ containing exactly one element in $Q \times \Sigma$ which is used to specify the current control state and location of the tape head. For instance, the initial configuration C_0^w is given by

$$C_0^w = a(q_0, w_1) w_2 w_3 ... w_{|w|} \flat\flat ....\flat a$$

We use $C_i^w(j)$ to denote the jth element of configuration C_i^w. For a given element $d \in \Delta$ we define the set $\mathrm{Pre}(d)$ of predecessor triples of d as

$$\begin{aligned}
&\mathrm{Pre}(d) = \\
&\quad \{(d_1, d_2, d_3) \in \Sigma^3 \mid d_2 = d\} \\
&\quad \cup \{((q,b), d_2, d_3) \in (Q \times \Sigma) \times \Sigma^2 \mid d = (\delta_1(q,b), d_2) \text{ and } \delta_3(q,b) = +1\} \\
&\quad \cup \{((q,b), d_2, d_3) \in (Q \times \Sigma) \times \Sigma^2 \mid d = d_2 \text{ and } \delta_3(q,b) \neq +1\} \\
&\quad \cup \{(d_1, d_2, (q,b)) \in \Sigma^2 \times (Q \times \Sigma) \mid d = (\delta_1(q,b), d_2) \text{ and } \delta_3(q,b) = -1\} \\
&\quad \cup \{(d_1, d_2, (q,b)) \in \Sigma^2 \times (Q \times \Sigma) \mid d = d_2 \text{ and } \delta_3(q,b) \neq -1\} \\
&\quad \cup \{(d_1, (q,b), d_3) \in \Sigma \times (Q \times \Sigma) \times \Sigma \mid d = \delta_2(q,b)\}
\end{aligned}$$

The idea is that given the three elements $C_i^w(j-1), C_i^w(j)$ and $C_i^w(j+1)$ one can uniquely determine $C_{i+1}^w(j)$ according to the definition of a Turing machine. Pre(d) is then the set of all triples (d_1, d_2, d_3) such that it is possible to have $C_i^w(j-1) = d_1, C_i^w(j) = d_2, C_i^w(j+1) = d_3$ and $C_{i+1}^w(j) = d$.

We now define $\mathcal{H} = (S', \{0,1\}, (S'_0, S'_1), R')$ by

- $S' = (\{0, ..., 2^{2^n}+1\} \times (\Delta \cup \Delta^3)) \cup \{s'_0, s'_z, s'_r, s'_F\}$
- $S'_0 = (\{0, ..., 2^{2^n}+1\} \times \Delta) \cup \{s'_0\}$
- $S'_1 = (\{0, ..., 2^{2^n}+1\} \times \Delta^3) \cup \{s'_z, s'_r, s'_F\}$
- R' contains exactly the elements below
  - $(s'_0, 1, s'_0) \in R'$
  - $(s'_0, 0, (1, (q_F, a))) \in R'$
  - $((j,d), 0, (j, (d_1, d_2, d_3))) \in R'$ for all $j \in \{1, ..., 2^{2^n}\}$ and all $(d_1, d_2, d_3) \in$ Pre(d)
  - For $j \in \{0, 2^{2^n}+1\}$ we have $((j,a), 0, s'_F) \in R'$ and $((j,d), 0, s'_r) \in R'$ when $d \neq a$
  - $((j,d), 0, s'_z) \in R'$ for all (j,d) such that $C_0^w(j) = d$.
  - $(s'_z, 0, s'_F) \in R'$
  - $(s'_z, -1, s'_r) \in R'$
  - $((j, (d_1, d_2, d_3)), -1, (j-1, d_1)) \in R'$ for all $j \in \{1, ..., 2^{2^n}\}$ and all $d_1, d_2, d_3 \in \Delta$
  - $((j, (d_1, d_2, d_3)), -1, (j, d_2)) \in R'$ for all $j \in \{1, ..., 2^{2^n}\}$ and all $d_1, d_2, d_3 \in \Delta$
  - $((j, (d_1, d_2, d_3)), -1, (j+1, d_3)) \in R'$ for all $j \in \{1, ..., 2^{2^n}\}$ and all $d_1, d_2, d_3 \in \Delta$

The different types of transitions are shown in Figure 5.12, 5.13 and 5.14. The intuition is that player 0 tries to show that $\mathcal{T}$ accepts w and player 1 tries to prevent this. Initially, player 0 can increase the counter to any natural number, assume he chooses v. If $\mathcal{T}$ accepts w it does so in a final configuration with the tape head pointing at cell 1 holding the symbol a with the current control state q_F. The game is now played by moving backwards from the state $(1, (q_F, a))$ holding this information. Player 0 can choose a predecessor triple that leads to $(1, (q_F, a))$. Player 1 then chooses one of the elements of the triple, the counter is decreased by one and the play continues like this. Finally, if the counter is 0 in a state (j,d) such that $C_0^w(j) = d$ then player 0 can win by going to s'_z from which player 1 can only go to s'_F. We will argue that player 0 can make sure that this happens if and only if $\mathcal{T}$ accepts w after performing v steps.

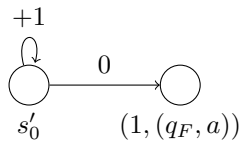


Figure 5.12: From the initial state, player 0 can increase the counter to any natural number before starting the game.

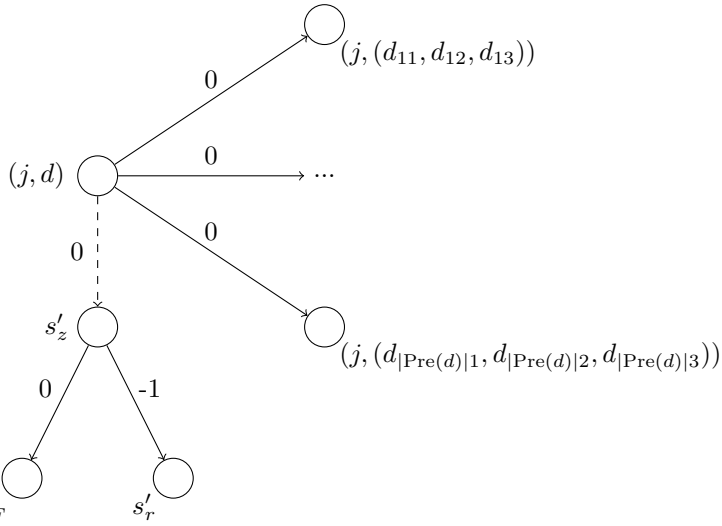


Figure 5.13: From a state $(j, d) \in \{1, ..., 2^{2^n}\} \times \Delta$ player 0 can choose a predecessor triple of d. The dashed transition only exists when $C_0^w(j) = d$. In this case player 0 can be sure to win if the current counter value is 0.

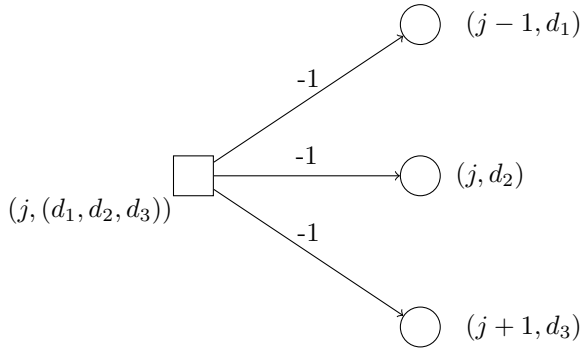


Figure 5.14: From a precedessor triple chosen by player 0, player 1 can choose which predecessor to continue with.

Lemma 5.10 *The configuration* $((j,d),i) \in (\{1,...,2^{2^n}\} \times \Delta) \times \mathbb{N}$ *is winning for player* 0 *if and only if* $C_i^w(j) = d$. *In particular* $((1,(q_F,a)),i)$ *is winning for player* 0 *if and only if* $C_i^w(1) = (q_F,a)$ *if and only if* $\mathcal{T}$ *accepts* w *after* i *steps of computation.*

Proof. The proof is done by induction on i. For the base case $i = 0$ the statement says that $((j,d),0)$ is winning for player 0 if and only if $C_0^w(j) = d$. Indeed, if $((j,d),0)$ is winning for player 0 he must go directly from (j,d) to s'_z because all other paths are blocked after one step since the counter value is 0. If he goes to s'_z then he wins because player 1 can only go to s'_F. However, note that there is only a transition from (j,d) to s'_z if $C_0^w(j) = d$ by construction. Thus, if player 0 is winning from $((j,d),0)$ then $C_0^w(j) = d$. For the other direction, suppose $C_0^w(j) = d$. Then player 0 can make sure to win by going to s'_z.

For the induction step, suppose the lemma is true for i. Now we need to show that $((j,d),i+1)$ is winning for player 0 if and only if $C_{i+1}^w(j) = d$. Suppose first that $((j,d),i+1)$ is winning for player 0. The winning strategy σ cannot consist in going directly to s'_z because then player 1 can go to s'_r. Thus, player 0 must choose a predecessor triple $(d_1,d_2,d_3) \in \mathrm{Pre}(d)$ when playing according to σ. After he chooses this, player 1 chooses one of them and the counter is decreased by one. Thus, player 1 can choose either $((j-1,d_1),i)$, $((j,d_2),i)$ or $(j+1,d_3),i)$. Thus, by the induction hypothesis $C_i^w(j-1) = d_1$, $C_i^w(j) = d_2$ and $C_i^w(j+1) = d_3$ since player 0 is winning. By the definition of predecessor triples, this means that $C_{i+1}^w(j) = d$. For the other direction, suppose $C_{i+1}^w(j) = d$. Then by going to the state $(j,(C_i^w(j-1),C_i^w(j),C_i^w(j+1)))$ he can be sure to win by the induction hypothesis. □

Lemma 5.11 *Starting in configuration* $(s'_0,0)$ *player* 0 *can make sure to reach* s'_F *if and only if* $\mathcal{T}$ *accepts* w.

Simulating the counter using ATL*

We have now reduced the word acceptance problem to a reachability game in an OCG $\mathcal{H}$ with a doubly-exponential number of states. Due to the structure of $\mathcal{H}$ we can reduce this to model-checking an ATL* formula Φ in the OCG $\mathcal{G}$. The difficult part is that we need to store the position of the tape head, which can be of doubly-exponential size. The other features of $\mathcal{H}$ are polynomial in the input.

Note that at each step of the game, the position of the tape head either stays the same, increases by one or decreases by one. This is essential for our ability to encode it using ATL*. We construct $\mathcal{G}$ much like $\mathcal{H}$ but where the position of the tape head is not present in the set of states. Instead, for each transition in the game between states s and s' we have a module in which player 0 encodes the position of the tape head by his choices. This is done much like in the proof of Proposition 5.9. However, we need a 2^n-bit counter in this case rather than just a c-bit counter. For this we need alternation between the players. It is done by giving player 1 the possibility to

challenge if player 0 has not chosen the correct value of the tape head position. This can be ensured by use of the ATL* formula $\Phi = \langle\langle\{0\}\rangle\rangle\varphi$ where φ is an LTL formula. The details of simulating a 2^n-bit counter like this can be obtained from [Kup+00; Boz07; BMP10].

According to the choices of player 1 then player 0 must be able to increase, decrease or leave unchanged the position of the tape head. This can be enforced by a formula with a size polynomial in n. Except for having to implement the position of the tape head in this way, the rules of $\mathcal{G}$ are the same as for $\mathcal{H}$ where player 0 needs to show that $\mathcal{T}$ accepts w by choosing a strategy that ensures reaching a certain state in the game while updating the tape head position correctly. In the end, this means that for the initial state s_0 in $\mathcal{G}$ corresponding to s'_0 in $\mathcal{H}$ we get $\mathcal{G}, s_0, 0 \models \langle\langle\{0\}\rangle\rangle(\varphi \wedge \mathbf{F}s_F)$ if and only if $\mathcal{T}$ halts on w. Here we assume that the play also goes to a halting state s_F corresponding to s'_F if player 1 challenges the counter value incorrectly.

Proposition 5.12 *The combined complexity of model-checking* ATL* *is* 2ExpSpace*-hard in both OCGs and SOCGs.*

Since our lower bound is for formulas of the form $\langle\langle\{0\}\rangle\rangle\varphi$ where φ is an LTL formula and player 0 is a player this means that the complexity of deciding the winner in one-counter games with LTL objectives is 2ExpSpace-complete both in the succinct and non-succinct case. With a fixed formula the complexity of this problem is PSpace-complete in OCGs due to PSpace-hardness of model-checking LTL in finite-state systems [SC85]. For SOCGs the model-checking game from Section 5.3 provides a reduction from data complexity of model-checking CTL in SOCPs to deciding the winner in succinct OCPGs with a fixed number of colors. Such a parity condition can be expressed by a fixed LTL objective.

Corollary 5.13 *Deciding the winner in two-player OCGs and SOCGs with* LTL *objectives are both* 2ExpSpace*-complete. For a fixed* LTL *formula, these problems are* PSpace*-complete and* ExpSpace*-complete respectively.*

5.5 Summary

Complexity results for a number of model-checking problems for one-counter games have been proved for alternating-time temporal logics and branching-time temporal logics including quantitative extensions. In addition to this we have solved the open problem of finding the complexity of deciding the winner in one-counter games and succinct one-counter games with LTL objectives. The 2ExpSpace-completeness closes the gap between the known 2ExpTime-hardness [PR89b] and membership in 3ExpTime [LMS04].

CHAPTER 6

Symmetry reduction in infinite games with finite branching

This chapter is an adapted version of the paper

- [MV14] Nicolas Markey and Steen Vester. "Symmetry Reduction in Infinite Games with Finite Branching". In: *Proceedings of the 12th International Symposium on Automated Technology for Verification and Analysis (ATVA)*. volume 8837. Lecture Notes in Computer Science. Springer, November 2014, pages 281–296

It has been updated to be compatible with the other chapters of the thesis. No major things have been added or removed.

6.1 Introduction

Symmetry reduction techniques have been introduced in model-checking around twenty years ago for combatting the state-space explosion in systems that posses some amount of symmetry [Cla+96; ES96; ID96; Cla+98]. The idea is to merge states of a system that behave in the same way with respect to a given property φ. This provides a smaller model of the system which exhibits the same behaviors as the original model with respect to φ. This yields a more efficient verification procedure since the original model need not be constructed.

While the technique does not guarantee a great efficiency improvement in general, it has been applied to a large number of practical cases with great success [ID96; Cla+96; Cla+98; Hen+03; KNP06; WBE08]. These applications include extensions from traditional model-checking of finite-state transition systems to real-time systems [Hen+03] and probabilistic systems [KNP06]. It seems that many naturally occuring instances of model-checking of concurrent and hardware systems contain symmetry and therefore the technique is very applicable.

Contribution

In this chapter, we extend symmetry reduction for transition systems to symmetry reduction of games. We expect that on practical instances, symmetry reduction in games should be as applicable as it has been in model-checking of temporal logics. Our contribution is to extend the symmetry reduction technique introduced in [ES96; Cla+96] to games. A central result in these papers is a correspondence lemma that describes a correspondence between paths in an original model M and in a reduced model M^G. Here, G is a symmetry group used to perform the reduction. This correspondence is used to conclude that CTL* model-checking can be performed in the reduced model instead of the original model.

In our setting, the correspondence lemma describes a correspondence between *strategies* in an original game $\mathcal{G}$ and in a reduced game $\mathcal{G}^G$. This lemma can then be used to establish a correspondence between winning strategies in the original game and in the reduced game for many different types of objectives. In particular, it follows from this that ATL* model-checking can be performed in the reduced game, and that parity games can be reduced while preserving existence of winning strategies.

The technique, however, is applicable for a much more general set of objectives. The proof that the reduction works for games is technically more involved than for finite-state transition systems, due to the possible irregular behaviours of an opponent player. This phenomenon leads us to apply König's Lemma [Kön36] in order to prove the correspondence between the original game and the reduced game when we assume finite branching. In addition, our approach does not restrict to finite-state games; it also works for games played on infinite graphs, provided that they have finite branching.

Outline

In Section 6.2 the symmetry reduction technique for turn-based games is introduced. In Section 6.3 some applications of the symmetry reduction technique are presented and in Section 6.4 we discuss challenges of automation of symmetry reduction. In Section 6.5 we provide a summary of the results.

6.2 Symmetry Reduction

In this chapter we consider only turn-based games with finite branching. For simplicity of presentation we restrict to two-player games and to games with a left-total transition relation. That is, every state has at least one successor state. However, the techniques can be naturally extended to handle such cases.

For the remainder of the chapter when we write turn-based game we implicitly mean two-player turn-based game with finite branching and a left-total transition relation. The two players are called player 0 and player 1.

Recall that an *objective* is a set $\Omega \subseteq \mathrm{Play}(\mathcal{G})$ of plays. A play ρ *satisfies* an objective Ω iff $\rho \in \Omega$. We say that σ_j is a winning strategy for player $j \in \{0, 1\}$

from state s_0 with objective Ω if $\text{Play}(\mathcal{G}, s_0, \sigma_j) \subseteq \Omega$. If such a strategy exists, we say that s_0 is a winning state for player j with objective Ω. The set of winning states for player j with objective Ω in game $\mathcal{G}$ is denoted $W_j(\mathcal{G}, \Omega)$.

In the following we fix a turn-based game $\mathcal{G} = (S, \{0,1\}, (S_0, S_1), R)$.

Definition 6.1 *A permutation π of S is called a* symmetry *for $\mathcal{G}$ if for all $s, s' \in S$*

1. $(s, s') \in R \Leftrightarrow (\pi(s), \pi(s')) \in R$
2. $s \in S_j \Leftrightarrow \pi(s) \in S_j$ *for* $j \in \{0,1\}$

Let $\text{Sym}_{\mathcal{G}}$ be the set of all symmetries in $\mathcal{G}$. We call a set G of symmetries a *symmetry group* if $(G, \circ)$ is a group, where $\circ$ is the composition operator defined by $(f \circ g)(x) = f(g(x))$. We consider G to be a fixed symmetry group in the rest of this section.

Definition 6.2 *The* orbit $\theta(s)$ *of a state s induced by G is given by*

$$\theta(s) = \{s' \in S \mid \exists \pi \in G.\ \pi(s) = s'\}.$$

The orbits induce an equivalence relation $\sim_G$ defined by $s \sim_G s'$ if and only if $s \in \theta(s')$. The reason for $\sim_G$ being an equivalence relation is that G is a group. The orbit $\theta(s)$ can be thought of as a set of states that have the same behavior as s with respect to the symmetry defined by G. For a path $\rho = s_0 s_1 ...$ in $\mathcal{G}$ we define $\theta(\rho) = \theta(s_0)\theta(s_1)....$ From each orbit $\theta(s)$ we choose a unique state $\text{Rep}(\theta(s)) \in \theta(s)$ as a *representative* of the orbit. For a strategy σ of player $j \in \{0,1\}$, an initial state s_0 and a sequence $t_0 ... t_\ell$ of orbits we choose a unique representative history $\text{Rep}_{s_0,\sigma}(t_0 ... t_\ell) = s_0 ... s_\ell$ that is compatible with σ and such that $s_i \in t_i$ for all $0 \leq i \leq \ell$ given that such a history exists. Otherwise, we define $\text{Rep}_{s_0,\sigma}(t_0 ... t_\ell) = \bot$.

We are now ready to define the notion of a *quotient game*.

Definition 6.3 *Given a game $\mathcal{G} = (S, \{0,1\}, (S_0, S_1), R)$ and a symmetry group G we define the* quotient game $\mathcal{G}^G = (S^G, \{0,1\}, (S_0^G, S_1^G), R^G)$ *by*

- $S^G = \{\theta(s) \mid s \in S\}$
- $R^G = \{(\theta(s), \theta(s')) \mid (s, s') \in R\}$
- $S_j^G = \{\theta(s) \mid s \in S_j\}$ *for* $j \in \{0,1\}$

Notice that $\mathcal{G}^G$ is indeed a game structure: symmetries respects the partition of S into S_0 and S_1, and therefore S_0^G and S_1^G also constitute a partition of S^G. Also, R^G is left-total and has finite branching.

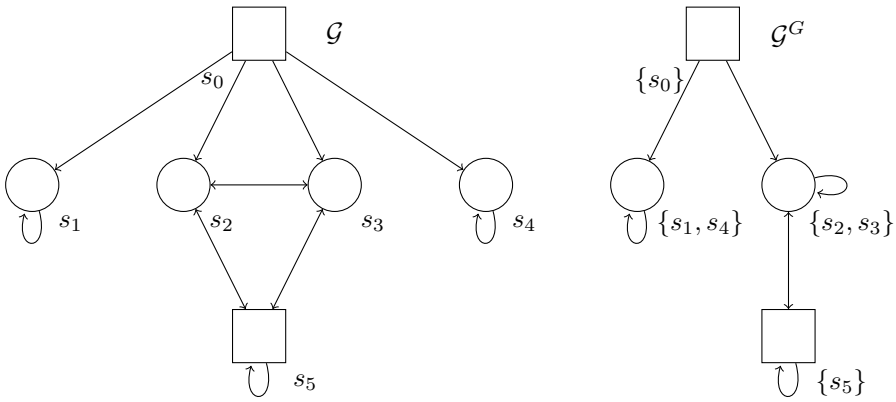


Figure 6.1: A game $\mathcal{G}$ to the left that has symmetric properties and the quotient game $\mathcal{G}^G$ induced by G on the right.

Example 6.4 *Consider the game $\mathcal{G} = (S, \{0,1\}, (S_0, S_1), R)$ to the left in Figure 6.1 and define*

$$G = \left\{ \pi \in \mathrm{Sym}_{\mathcal{G}} \;\middle|\; \begin{array}{ll} \pi(s_0,s_1,s_2,s_3,s_4,s_5) & = (s_0,s_1,s_2,s_3,s_4,s_5) \vee \\ \pi(s_0,s_1,s_2,s_3,s_4,s_5) & = (s_0,s_4,s_2,s_3,s_1,s_5) \vee \\ \pi(s_0,s_1,s_2,s_3,s_4,s_5) & = (s_0,s_1,s_3,s_2,s_4,s_5) \vee \\ \pi(s_0,s_1,s_2,s_3,s_4,s_5) & = (s_0,s_4,s_3,s_2,s_1,s_5) \end{array} \right\}$$

It is easy to see that G is a symmetry group. G now induces the orbits $\{s_0\}$, $\{s_5\}$, $\{s_2, s_3\}$ and $\{s_1, s_4\}$. This gives rise to the quotient game $\mathcal{G}^G$ to the right in Figure 6.1. Note how the construction gives us a smaller game that still has many of the structural properties of the original game.

We can show the following correspondence between plays in the original game and the quotient game.

Lemma 6.5 *Let $\mathcal{G} = (S, \{0,1\}, (S_0, S_1), R)$ be a game and G be a symmetry group. Then*

1. *For each play $\rho \in \mathrm{Play}(\mathcal{G})$ there exists a play $\rho' \in \mathrm{Play}(\mathcal{G}^G)$ such that $\rho'_0 = \theta(\rho_0)$ and $\rho_i \in \rho'_i$ for all $i \geq 0$*
2. *For each play $\rho' \in \mathrm{Play}(\mathcal{G}^G)$ and every $s \in \rho'_0$ there exists a play $\rho \in \mathrm{Play}(\mathcal{G}, s)$ such that $\rho_i \in \rho'_i$ for all $i \geq 0$*

Proof. (1.) Suppose $\rho \in \mathrm{Play}(\mathcal{G})$. Then for every $i \geq 0$ we have $(\rho_i, \rho_{i+1}) \in R$. This implies that $(\theta(\rho_i), \theta(\rho_{i+1})) \in R^G$. Thus, $\theta(\rho) \in \mathrm{Play}(\mathcal{G}^G)$. Since $\rho_i \in \theta(\rho_i)$ the result follows.

(2.) Suppose $\rho' \in \text{Play}(\mathcal{G}^G)$ and let $s \in \rho'_0$. Construct a play ρ as follows. First, let $\rho_0 = s$. Next, suppose that the history $\rho_{\leq i}$ has been constructed for some $i \geq 0$ such that $\rho_j \in \rho'_j$ for all $0 \leq j \leq i$. We have that $(\rho'_i, \rho'_{i+1}) \in R^G$ which implies that there exists states $s', s'' \in S$ such that $(s', s'') \in R$, $s' \in \rho'_i$ and $s'' \in \rho'_{i+1}$. Since $\rho_i \in \rho'_i$ there exists $\pi \in G$ such that $\pi(s') = \rho_i$. Further, we have $(s', s'') \in R$ implies that $(\pi(s'), \pi(s'')) \in R$ since π is a symmetry. Since $\pi(s') = \rho_i$ this means that $(\rho_i, \pi(s'')) \in R$. Since $s'' \in \rho'_{i+1}$ we have $\pi(s'') \in \rho'_{i+1}$. Thus, the history $\rho_{\leq i}$ can be extended to a new history by setting $\rho_{i+1} = \pi(s'')$ and satisfying $\rho_{i+1} \in \rho'_{i+1}$. We can keep extending it in this way to obtain a play ρ with the desired property. □

We now show a correspondence lemma between strategies in the original game $\mathcal{G}$ and the quotient game $\mathcal{G}^G$.

Lemma 6.6 *Let $\mathcal{G} = (S, \{0,1\}, (S_0, S_1), R)$ be a game, G be a symmetry group, $s_0 \in S$ be an initial state, $t_0 = \theta(s_0)$ and $j \in \{0,1\}$. Then*

1. *For any strategy σ of player j in $\mathcal{G}$ there exists a strategy σ' of player j in $\mathcal{G}^G$ such that for all $t_0 t_1 ... \in \text{Play}(\mathcal{G}^G, t_0, \sigma')$ there exists $s_0 s_1 ... \in \text{Play}(\mathcal{G}, s_0, \sigma)$ where $s_i \in t_i$ for all $i \geq 0$;*
2. *For any strategy σ' of player j in $\mathcal{G}^G$ there exists a strategy σ of player j in $\mathcal{G}$ such that for all $s_0 s_1 ... \in \text{Play}(\mathcal{G}, s_0, \sigma)$ there exists $t_0 t_1 ... \in \text{Play}(\mathcal{G}^G, t_0, \sigma')$ where $s_i \in t_i$ for all $i \geq 0$.*

Proof. (1.) Let σ be a strategy for player $j \in \{0,1\}$ in the original game $\mathcal{G}$. From this we construct a strategy σ' for player j in the quotient game $\mathcal{G}^G$ by

$$\sigma'(h) = \theta(\sigma(\text{Rep}_{s_0,\sigma}(h)))$$

for all $h \in \text{Hist}(\mathcal{G}^G)$ such that $\text{Rep}_{s_0,\sigma}(h) \neq \bot$ and arbitrarily when $\text{Rep}_{s_0,\sigma}(h) = \bot$. This strategy is well-defined since when $\text{Rep}_{s_0,\sigma}(h) \neq \bot$ we have

$$\begin{aligned}(\text{last}(\text{Rep}_{s_0,\sigma}(h)), \sigma(\text{Rep}_{s_0,\sigma}(h))) \in R &\Rightarrow (\theta(\text{last}(\text{Rep}_{s_0,\sigma}(h))), \theta(\sigma(\text{Rep}_{s_0,\sigma}(h)))) \in R^G \\ &\Rightarrow (\text{last}(h), \theta(\sigma(\text{Rep}_{s_0,\sigma}(h)))) \in R^G\end{aligned}$$

which means that there is a legal transition to the successor state prescribed by the strategy σ'.

Now, let $\rho = t_0 t_1 ... \in \text{Play}(\mathcal{G}^G, t_0, \sigma')$ be an arbitrary play compatible with σ' in $\mathcal{G}^G$ from t_0. We construct a directed tree T where the root is labelled by s_0 and where the labelling of the infinite paths in T are exactly the plays compatible with σ in $\mathcal{G}$ from s_0. From this tree we obtain a new tree T_ρ by cutting away from T part of the branches labelled $s_0 s_1 ...$ on which there exists $i \geq 0$ such that $s_i \notin t_i$. If k is the smallest number such that $s_k \notin t_k$ then the nodes labelled $s_k s_{k+1} ...$ are removed. The situation is illustrated in Figure 6.2.

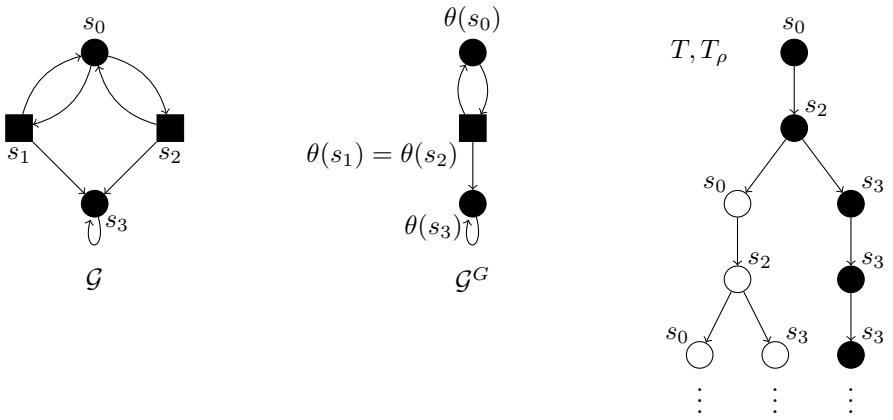


Figure 6.2: From left to right is drawn the original game $\mathcal{G}$, the quotient game $\mathcal{G}^G$ and the trees T, T_ρ where $G = \{(s_0, s_1, s_2, s_3), (s_0, s_2, s_1, s_3)\}, \sigma(h) = s_2$ for all histories h ending in s_0 and $\rho = \theta(s_0)\theta(s_1)\theta(s_3)^\omega$. T and T_ρ are drawn together. T is the whole tree, T_ρ only consists of the filled nodes.

We assume for contradiction that T_ρ has a finite height ℓ. This means that there must be a branch in the tree labelled by the history $\text{Rep}_{s_0,\sigma}(t_0, ..., t_\ell) = s_0...s_\ell$ because if we had $\text{Rep}_{s_0,\sigma}(t_0, ..., t_\ell) = \bot$ then T_ρ would have had height smaller than ℓ. There are now two cases to consider

- Suppose $s_\ell \in S_j$. Then due to the definition of σ' we get

$$\sigma(s_0...s_\ell) = \sigma(\text{Rep}_{s_0,\sigma}(t_0...t_\ell)) \in \sigma'(t_0...t_\ell) = t_{\ell+1}$$

  Since $s_0...s_\ell\sigma(s_0...s_\ell)$ is compatible with σ and $s_i \in t_i$ for $0 \leq i \leq \ell$ then $s_0...s_\ell\sigma(s_0...s_\ell)$ is the labelling of a path in T_ρ which gives a contradiction since it has length $\ell + 1$.

- Suppose $s_\ell \notin S_j$. Since $(t_\ell, t_{\ell+1}) \in R_G$ there must exist $(u, v) \in R$ such that $u \in t_\ell$ and $v \in t_{\ell+1}$. Since $u \in t_\ell = \theta(s_\ell)$ there exists a permutation $\pi \in G$ such that $\pi(u) = s_\ell$. We also have that $(u, v) \in R \Rightarrow (\pi(u), \pi(v)) \in R \Rightarrow (s_\ell, \pi(v)) \in R$ since π preserves the transition relation. Since $(s_\ell, \pi(v)) \in R$ and s_ℓ is not owned by player j then $s_0...s_\ell\pi(v)$ is compatible with σ. But we also have $\pi(v) \in t_{\ell+1}$ since $v \in t_{\ell+1}$. Thus, $s_0...s_\ell\pi(v)$ is the labelling of a path in T_ρ of length $\ell + 1$. This gives a contradiction as well.

This means that T_ρ is infinite. Since it is also finitely branching it has an infinite path according to König's Lemma. Let the labelling of such a path be $s_0s_1...$. Since $s_0s_1...$ is the labelling of an infinite path in T_ρ it is a play compatible with σ since all infinite paths in T_ρ are infinite paths in T. Moreover, since it is an infinite path

in T_ρ it satisfies $s_i \in t_i$ for all $i \geq 0$, because otherwise it would not be present in T_ρ. This proves the first part since $t_0 t_1 ...$ was an arbitrary play compatible with σ'.

(2.) Let σ' be a strategy for player j in $\mathcal{G}^G$. Define σ from this in such a way that

$$\sigma(s_0 ... s_\ell) \in \sigma'(\theta(s_0) ... \theta(s_\ell))$$

for all histories $s_0 ... s_\ell$ in $\mathcal{G}$ with $s_\ell \in S_j$. Note that when $s_0 ... s_\ell$ is a history in $\mathcal{G}$ then $\theta(s_0) ... \theta(s_\ell)$ is a history in $\mathcal{G}^G$. Further, we need to check that there exists a state $s \in \sigma'(\theta(s_0) ... \theta(s_\ell))$ such that $(s_\ell, s) \in R$ in order for the definition to make sense. This can be seen as follows. Since $(\theta(s_\ell), \sigma'(\theta(s_0) ... \theta(s_\ell))) \in R^G$ there exists $(u, v) \in R$ such that $u \in \theta(s_\ell)$ and $v \in \sigma'(\theta(s_0) ... \theta(s_\ell))$. This means that there exists $\pi \in G$ with $\pi(u) = s_\ell$. Now, $(u, v) \in R \Rightarrow (\pi(u), \pi(v)) \in R \Rightarrow (s_\ell, \pi(v)) \in R$. Since $\pi(v) \in \theta(v) = \sigma'(\theta(s_0) ... \theta(s_\ell))$ the state $s = \pi(v)$ satisfies the property.

Now, suppose $s_0 s_1 ... \in \text{Play}(\mathcal{G}, \sigma)$. We will show that $\theta(s_0)\theta(s_1) ... \in \text{Play}(\mathcal{G}^G, \sigma')$. This will prove 2. since $s_i \in \theta(s_i)$ for all $i \geq 0$. For any prefix $\theta(s_0) ... \theta(s_\ell)$ we have that

- If $\theta(s_\ell) \notin S_j^G$ then $(s_\ell, s_{\ell+1}) \in R$ implies that $(\theta(s_\ell), \theta(s_{\ell+1})) \in R^G$.
- If $\theta(s_\ell) \in S_j^G$ then $s_{\ell+1} = \sigma(s_0 ... s_\ell) \in \sigma'(\theta(s_0) ... \theta(s_\ell))$. This implies that $\theta(s_{\ell+1}) = \sigma'(\theta(s_0) ... \theta(s_\ell))$.

This means that $\theta(s_0)\theta(s_1) ...$ is indeed compatible with σ'. □

This lemma leads to desirable properties of the quotient game when certain types of objectives are considered.

Definition 6.7 *A symmetry group G preserves the objective Ω if for any two plays $s_0 s_1 ... \in \text{Play}(\mathcal{G})$ and $s'_0 s'_1 ... \in \text{Play}(\mathcal{G})$ it follows from $s_0 s_1 ... \in \Omega$ and $s_i \sim_G s'_i$ for all $i \geq 0$ that $s'_0 s'_1 ... \in \Omega$.*

If Ω is an objective and G is a symmetry group that preserves it, then we denote by Ω^G the objective in the quotient game $\mathcal{G}^G$ defined as $\Omega^G = \{\theta(s_0)\theta(s_1) ... \mid s_0 s_1 ... \in \Omega\}$. Lemma 6.6 gives us the following.

Theorem 6.8 *Let $\mathcal{G}$ be a game, G be a symmetry group that preserves the objective Ω, $j \in \{0, 1\}$ and $s_0 \in S$. Then*

$$s_0 \in W_j(\mathcal{G}, \Omega) \text{ if and only if } \theta(s_0) \in W_j(\mathcal{G}^G, \Omega^G).$$

Proof. ($\Rightarrow$) Suppose player j has a winning strategy σ in $\mathcal{G}$ with objective Ω from state s_0. Then $\text{Play}(\mathcal{G}, s_0, \sigma) \subseteq \Omega$. According to Lemma 6.6 there is a strategy σ' for player j in $\mathcal{G}^G$ such that for a given play $t_0 t_1 ... \in \text{Play}(\mathcal{G}^G, \theta(s_0), \sigma')$ there exists a play $s_0 s_1 ... \in \text{Play}(\mathcal{G}, s_0, \sigma)$ with $s_i \in t_i$ for all $i \geq 0$. Since G preserves Ω and $\text{Play}(\mathcal{G}, s, \sigma) \subseteq \Omega$ this means that $t_0 t_1 ... \in \Omega^G$. Since $t_0 t_1 ...$ is an arbitrary play

compatible with σ' from $\theta(s_0)$ we have $\text{Play}(\mathcal{G}^G, \theta(s_0), \sigma') \subseteq \Omega^G$ and thus $\theta(s_0) \in W_j(\mathcal{G}^G, \Omega^G)$.

($\Leftarrow$) Suppose player j has a winning strategy σ' in $\mathcal{G}^G$ with objective Ω^G from state $\theta(s_0)$. Then $\text{Play}(\mathcal{G}^G, \theta(s_0), \sigma') \subseteq \Omega^G$. According to Lemma 6.6 there is a strategy σ for player j in $\mathcal{G}$ such that for a given play $s_0 s_1 ... \in \text{Play}(\mathcal{G}, s_0, \sigma)$ there exists a play $t_0 t_1 ... \in \text{Play}(\mathcal{G}^G, \theta(s_0), \sigma')$ with $s_i \in t_i$ for all $i \geq 0$. Since G preserves Ω and $\text{Play}(\mathcal{G}^G, \theta(s_0), \sigma') \subseteq \Omega^G$ this means that $s_0 s_1 ... \in \Omega$. Since $s_0 s_1 ...$ is an arbitrary play compatible with σ from s_0 we have $\text{Play}(\mathcal{G}, s_0, \sigma) \subseteq \Omega$ and thus $s_0 \in W_j(\mathcal{G}, \Omega)$. □

Corollary 6.9 *Let $\mathcal{G}$ be a game, G be a symmetry group that preserves the objective Ω, $j \in \{0, 1\}$ and $s, s' \in S$ be such that $s \sim_G s'$. Then*

$$s \in W_j(\mathcal{G}, \Omega) \text{ if and only if } s' \in W_j(\mathcal{G}, \Omega).$$

We have now shown the main result of this chapter, namely that a winning strategy exists in the original game if and only if it exists in the quotient game. This also implies that there is a winning strategy from a state s in the original game if and only if there is a winning strategy from another state s' that belongs to the same orbit.

For transition systems the correspondence between existence of paths in the original system and the quotient system as shown in Lemma 6.5 was enough to show that model-checking of a CTL* formula in the original system can be reduced to model-checking the same formula in the quotient system if the symmetry group preserves the labelling [ES96; Cla+96].

However, due to the possible behaviors of an opponent player we have had to generalize this result in Lemma 6.6 which directly leads to Theorem 6.8. It will be used in the next section to show that we can extend the symmetry reduction approach to ATL*. Since we apply Königs Lemma in the proof, we have assumed that the games are finitely branching. We leave it as an open problem whether the technique can be generalized to infinitely branching games as well.

6.3 Applications

In this section we provide some examples of applications of Theorem 6.8. We look at symmetry reductions for parity games and games with properties defined in temporal logics (by considering alternating bisimulation). We also consider an example of an infinite-state game with a corresponding finite-state quotient game. This makes it possible for us to decide existence of winning strategies in the original game by using standard techniques on the quotient game.

6.3.1 Parity games

Let $\mathcal{G} = (S, S_0, S_1, R, c)$ be a parity game. From this, the corresponding parity objective is given by $\Omega_j = \{\rho \in \text{Play}(\mathcal{G}) \mid c(\rho) \in \Xi_j\}$ for player j. We say that a symmetry group G *preserves the coloring* c if for all $s, s' \in S$ we have

$$s \sim_G s' \Rightarrow c(s) = c(s')$$

When G preserves c, we define a coloring function c^G on the set of orbits by $c^G(t) = c(\text{Rep}(t))$ for all orbits t. Using Theorem 6.8 we now get the following result for parity games when we have a symmetry group preserving the coloring function.

Proposition 6.10 *Let $\mathcal{G} = (S, S_0, S_1, R, c)$ be a parity game, G a symmetry group that preserves c, $s \in S$ and $j \in \{0, 1\}$. Then*

1. *G preserves the objective Ω_j,*
2. *$\Omega_j^G = \{\theta(s_0)\theta(s_1)... \in \text{Play}(\mathcal{G}^G) \mid c^G(\theta(s_0))c^G(\theta(s_1))... \in \Xi_j\}$,*
3. *$s \in W_j(\mathcal{G}, \Omega_j)$ if and only if $\theta(s) \in W_j(\mathcal{G}^G, \Omega_j^G)$.*

Proof. (1.) Suppose $s_0 s_1... \in \Omega_j$ and $s'_0 s'_1... \in \text{Play}(\mathcal{G})$ satisfy $s_i \sim_G s'_i$ for all $i \geq 0$. Since G preserves the coloring c the two sequences are colored equivalently and we have that $s'_0 s'_1... \in \Omega_j$ as well. Thus, G also preserves the objective Ω_j.

(2.) This can be seen as follows

$$\begin{aligned}
\Omega_j^G &= \{\theta(s_0)\theta(s_1)... \in \text{Play}(\mathcal{G}^G) \mid s_0 s_1... \in \Omega_j\} \\
&= \{\theta(s_0)\theta(s_1)... \in \text{Play}(\mathcal{G}^G) \mid c(s_0)c(s_1)... \in \Xi_j\} \\
&= \{\theta(s_0)\theta(s_1)... \in \text{Play}(\mathcal{G}^G) \mid c(\text{Rep}(\theta(s_0)))c(\text{Rep}(\theta(s_1)))... \in \Xi_j\} \\
&= \{\theta(s_0)\theta(s_1)... \in \text{Play}(\mathcal{G}^G) \mid c^G(\theta(s_0))c^G(\theta(s_1))... \in \Xi_j\}
\end{aligned}$$

(3.) From (1.) we have that G preserves Ω_j and thus, we get the result by applying Theorem 6.8. □

This means that if we have a symmetry group that preserves the coloring function we can decide existence of winning strategies in a parity game by deciding existence of winning strategies in the quotient game. Furthermore, the quotient game is also a parity game and it has the same number of colors as the original game.

Example 6.11 *Consider again the game $\mathcal{G}$ from Example 6.4. Let a coloring function c be defined by $c(s_0) = c(s_1) = c(s_5) = 0$ and $c(s_2) = c(s_3) = c(s_4) = 1$. Then the symmetry group G defined in the example does not preserve c since $s_1 \sim_G s_4$ but $c(s_1) \neq c(s_4)$. However, we can define a (smaller) symmetry group G' that preserves c by*

$$G' = \left\{ \pi \in \text{Sym}_{\mathcal{G}} \;\middle|\; \begin{array}{ll} \pi(s_0, s_1, s_2, s_3, s_4, s_5) & = (s_0, s_1, s_2, s_3, s_4, s_5) \\ \pi(s_0, s_1, s_2, s_3, s_4, s_5) & = (s_0, s_1, s_3, s_2, s_4, s_5) \end{array} \right\}$$

This does not give as great a reduction as G, but on the other hand it preserves the existence of winning strategies for parity conditions defined by c.

6.3.2 Alternating-time temporal logic

Consider a turn-based game $\mathcal{G} = (S, \{0,1\}, (S_0, S_1), R, \mathrm{AP}, L)$ with two players. We will show that the symmetry reduction technique can be applied for model-checking of the alternating-time temporal logic ATL* as well. We say that a symmetry group G *preserves the labelling function* L if for all $s, s' \in S$ we have

$$s \sim_G s' \Rightarrow L(s) = L(s')$$

When G preserves L we define a labelling function L^G on the set of orbits by $L^G(t) = L(\mathrm{Rep}(t))$ for all orbits t. By applying Theorem 6.8 we can now show that the symmetry reduction works for ATL*.

First we define *alternating bisimulation* [Alu+98] which is the analogue of bisimulation extended to games.

Definition 6.12 *Let $\mathcal{G} = (S, \{0,1\}, (S_0, S_1), R, \mathrm{AP}, L)$ be a turn-based game. Two states s and s' of S are* alternating bisimilar *if there exists a binary relation $\mathcal{B}$ over S such that*

- $(s, s') \in \mathcal{B}$;
- *for every $(t, t') \in \mathcal{B}$, it holds that $L(t) = L(t')$;*
- *for every $(t, t') \in \mathcal{B}$, if it holds that $t \in S_0$ if and only if $t' \in S_0$ then*
  - *for every u s.t. $(t, u) \in R$, there exists u' such that $(t', u') \in R$ and $(u, u') \in \mathcal{B}$;*
  - *for every u' s.t. $(t', u') \in R$, there exists u such that $(t, u) \in R$ and $(u, u') \in \mathcal{B}$;*
- *for every $(t, t') \in \mathcal{B}$, if it holds that $t \in S_0$ if and only if $t' \in S_1$ then*
  - *for every u, u' s.t. $(t, u) \in R$ and $(t', u') \in R$ it holds that $(u, u') \in \mathcal{B}$;*

Note in particular the fourth bullet in the definition. It tells us that two states can be alternating bisimilar even if they are controlled by different players. Though, this is only possible if all successors of the states are alternating bisimilar. That is, up to alternating bisimulation there are no different choices available to the players.

Next, we show that s in $\mathcal{G}$ and $\theta(s)$ in $\mathcal{G}^G$ are alternating bisimilar. As a consequence of the results in [Alu+98] this implies that s and $\theta(s)$ satisfy the same ATL* formulas if the game has a finite set of states.

Proposition 6.13 *Let $\mathcal{G} = (S, \{0,1\}, (S_0, S_1), R, \mathrm{AP}, L)$ be a turn-based game. Let G be a symmetry group that preserves L, and L^G be the quotient labelling function for S^G. Then for any $s \in S$, s and $\theta(s)$ are alternating bisimilar.*

Proof. We consider the disjoint union of $\mathcal{G}$ and $\mathcal{G}^G$, and the relation $\mathcal{B}$ defined by

$$(s, s') \in \mathcal{B} \quad \text{if, and only if,} \quad s' = \theta(s).$$

Then the first two conditions in the definition of alternating bisimilarity are fulfilled.

Now, pick $(t, t') \in \mathcal{B}$, assuming that t (hence also $t' = \theta(t)$) belongs to Player 0. First, pick a successor u of t, i.e. $(t, u) \in R$. Then $(\theta(t), \theta(u)) \in R^G$ and since $(u, \theta(u)) \in \mathcal{B}$ the first condition is satisfied. Second, pick a successor u' of t', i.e. $(t', u') \in R^G$. Then there exists $v, w \in S$ such that $(v, w) \in R$, $v \in t'$ and $w \in u'$. Then there exists $\pi \in G$ such that $\pi(v) = t$. This means that $(\pi(v), \pi(w)) = (t, \pi(w)) \in R$. Since $\pi(w) \in u'$ we also have $(\pi(w), u') \in \mathcal{B}$ which means the second condition is satisfied. The proof is the same if t belongs to Player 1.

□

We now show that s in $\mathcal{G}$ and $\theta(s)$ in $\mathcal{G}^G$ also satisfy the same ATL* formulas for infinite-state games with finite branching.

Proposition 6.14 *Let $\mathcal{G} = (S, \{0,1\}, (S_0, S_1), R, \mathrm{AP}, L)$ be a turn-based game and G be a symmetry group that preserves L. Then for every $s \in S$, every $\rho \in \mathrm{Play}(\mathcal{G})$, every* ATL* *state formula φ and every* ATL* *path formula Φ over* AP *we have*

- $\mathcal{G}, s \models \varphi$ *if and only if* $\mathcal{G}^G, \theta(s) \models \varphi$
- $\mathcal{G}, \rho \models \Phi$ *if and only if* $\mathcal{G}^G, \theta(\rho) \models \Phi$

where the satisfaction relation $\models$ in $\mathcal{G}^G$ is defined with respect to the labelling function L^G.

Proof. The proof is done by induction on the structure of the formula. We omit the proof for all the simple cases and focus on $\varphi = \langle\!\langle \{j\} \rangle\!\rangle \Phi$.

Define the objective $\Omega_\Phi = \{\rho \in \mathrm{Play}(\mathcal{G}) \mid \mathcal{G}, \rho \models \Phi\}$ as the set of plays in $\mathcal{G}$ satisfying Φ. We will first show that G preserves Ω_Φ. Suppose $\rho \in \Omega_\Phi$ and $\rho' \in \mathrm{Play}(\mathcal{G})$ is a play such that $\rho \sim_G \rho'$. According to the induction hypothesis, $\mathcal{G}, \rho \models \Phi$ if and only if $\mathcal{G}^G, \theta(\rho) \models \Phi$ but also that $\mathcal{G}, \rho' \models \Phi$ if and only if $\mathcal{G}^G, \theta(\rho') \models \Phi$. Since $\theta(\rho) = \theta(\rho')$ we have that ρ' satisfies Φ since ρ does. Thus, $\rho' \in \Omega_\Phi$ which means that G preserves Ω_Φ. Then by the induction hypothesis we have

$$\begin{aligned}
\Omega_\Phi^G &= \{\theta(\rho) \in \mathrm{Play}(\mathcal{G}^G) \mid \rho \in \Omega_\Phi\} \\
&= \{\theta(\rho) \in \mathrm{Play}(\mathcal{G}^G) \mid \mathcal{G}, \rho \models \Phi\} \\
&= \{\theta(\rho) \in \mathrm{Play}(\mathcal{G}^G) \mid \mathcal{G}^G, \theta(\rho) \models \Phi\}
\end{aligned}$$

Using this and Theorem 6.8 we have for all $s \in S$

$$\begin{aligned} \mathcal{G}, s \models \langle\langle\{j\}\rangle\rangle\Phi \text{ iff } & s \in W_j(\mathcal{G}, \Omega_\Phi) \\ \text{iff } & \theta(s) \in W_j(\mathcal{G}^G, \Omega_\Phi^G) \\ \text{iff } & \mathcal{G}^G, \theta(s) \models \langle\langle\{j\}\rangle\rangle\Phi \end{aligned}$$

□

Remark 6.15 *Even though the result for* ATL* *was only proved in two-player games above, this can easily be extended to handle n-player games for* $n \geq 3$ *as well. This is the case since formulas of the form* $\langle\langle A\rangle\rangle\Phi$ *can be evaluated at a state by letting one player control the players in coalition* A *and let another player control the players in coalition* $\Pi \setminus A$*.*

Remark 6.16 *Notice that the result of Proposition 6.14 does not extend to Strategy Logic [CHP07; MMV10] or* ATL *with strategy contexts [DLM10]. Considering the game depicted on Figure 6.1, assume that* s_2 *and* s_3 *are labelled with* p *and* s_5 *is labelled with* q*. One can notice that there is a strategy of the circle player (namely, playing from* s_2 *to* s_3 *and from* s_3 *to* s_5*) under which the following two propositions hold in* s_0*:*

- *there is a strategy for the square player to end up in a* p*-state after two steps (namely, playing to* s_2*),*
- *there is a strategy for the square player to end up in a* q*-state after two steps (namely, playing to* s_3*).*

This obviously fails in the reduced game.

Example 6.17 *Consider the infinite game illustrated in Figure 6.3 which is played on an infinite grid. Player* 0 *controls the circle states and player* 1 *controls the square states. The games starts in* $(0, 0)$ *and in each state the player controlling the state can move up, down, left or right. The proposition* p *is true exactly when the first coordinate is odd. The game is defined by* $\mathcal{G} = (S, \{0, 1\}, (S_0, S_1), R, \mathrm{AP}, L)$ *where*

- $S = \mathbb{Z}^2$
- $R = \{((x_1, y_1), (x_2, y_2)) \in S \times S \mid |x_1 - x_2| + |y_1 - y_2| = 1\}$
- $S_0 = \{(x, y) \in S \mid \text{ y is even}\}$
- $S_1 = \{(x, y) \in S \mid \text{ y is odd}\}$

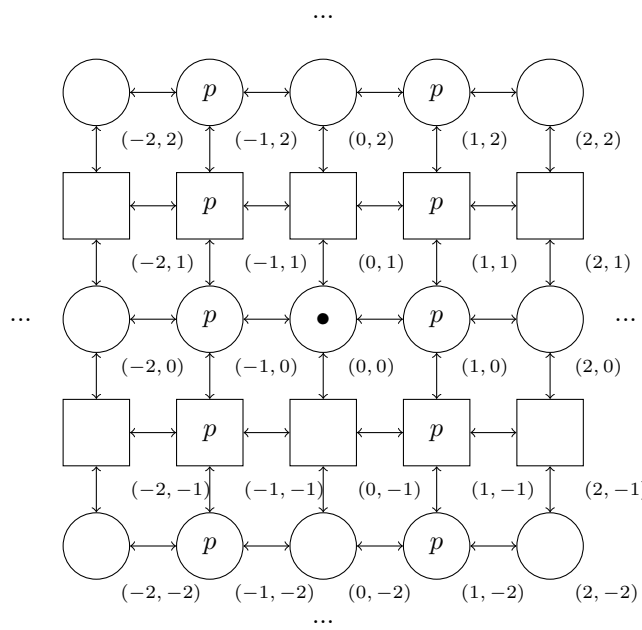


Figure 6.3: Game on an infinite grid.

The labelling is defined by $L((x,y)) = \{p\}$ *if* x *is odd and* $L((x,y)) = \emptyset$ *if* x *is even. Suppose we want to check if some* ATL* *formula* φ *over the set* $\mathrm{AP} = \{p\}$ *is true in* $(0,0)$*. This is not necessarily easy to do in an automatic way since* $\mathcal{G}$ *is infinite. However, we can use symmetry reduction to obtain a finite quotient game as follows. Let us define*

$$G = \{\pi \in \mathrm{Sym}_{\mathcal{G}} \mid \exists a, b \in \mathbb{Z}.\ \forall (x,y) \in S.\ \pi(x,y) = (x' + 2 \cdot a, y' + 2 \cdot b)\}.$$

It is simple to show that G *is a group and also that it preserves the labelling* L*. Further,* G *induces four orbits* $\theta((0,0)), \theta((0,1)), \theta((1,0))$ *and* $\theta((1,1))$*. The corresponding quotient game can be seen in Figure 6.4.*

According to Proposition 6.14 we can just do model-checking in the quotient game since $\mathcal{G}, (0,0) \models \varphi$ *if and only if* $\mathcal{G}^G, \theta((0,0)) \models \varphi$*. This shows an infinite-state game with a finite-state quotient game.*

6.4 Where do the symmetry groups come from?

Until now we have just assumed that a symmetry group G was known, but we have not mentioned how to obtain it. The short answer is that it is not feasible to find

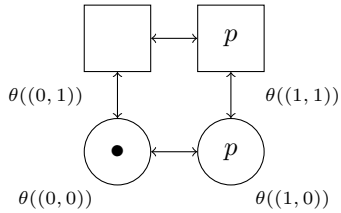


Figure 6.4: Finite-state quotient game.

the symmetry group that gives the largest reduction in general. Indeed, even for the special case of finite transition systems this problem is infeasible [ES96]. Another problem is that the orbit problem is as hard as the Graph Isomorphism problem when the transition system is finite [Cla+96]. The orbit problem is to decide for a given group G generated by a set $\{\pi_1, ..., \pi_n\}$ of permutations whether two states s and s' belong to the same orbit. For many applications it is even the case that we would like to construct the quotient game without building the original game explicitly in the first place because the state spaces are very large.

While this may seem quite negative, the approach has given very large speed-ups on practical verification instances. Here, it is typically the responsibility of the engineer doing the verification to provide the symmetry groups as well as the orbits to the program. The main reason why this is possible is that many natural instances of embedded, concurrent and distributed systems have a number of identical components or processes. This gives rise to symmetry in the model which is quite easy to detect for an experienced human. Another approach is to design modelling languages and data structures where certain forms of symmetry can be detected automatically. For discussions of this in different contexts, see [ID96; Hen+03; KNP06]. We have no reason to believe that the symmetry reduction technique will be less applicable for model-checking properties of games.

6.5 Summary

We have proved that the symmetry reduction technique can be generalized to infinite-state turn-based games with finite branching and provided particular applications of this result in the areas of parity games and model-checking of ATL*. The technique has not yet been implemented and tested on practical examples, but we expect that it should be as applicable as it has been in the context of model-checking of transition systems, model-checking of real-time systems and probabilistic systems. It is still open whether the technique can be generalized to games with infinite branching since our application of König's Lemma requires that the games have finite branching.

CHAPTER 7

Satisfiability in flat fragments of temporal logics

This chapter is an adapted version of the paper

- [GV14] Valentin Goranko and Steen Vester. "Optimal Decision Procedures for Satisfiability in Fragments of Alternating-time Temporal Logics". In: *Advances in Modal Logic 10, invited and contributed papers from the tenth conference on Advances in Modal Logic (AiML)*. 2014, pages 234–253

It has been updated to be compatible with the other chapters of the thesis. In particular, the introduction section has been reformatted and updated. Also, the section defining the flat fragments has been reorganized to be consistent with other chapters of the thesis.

7.1 Introduction

While found to be quite useful and natural for specification in open systems and multi-agent systems the logic ATL* has some problematic features related to the nesting of strategic quantifiers. This is why we propose to study *flat fragments* of ATL* in this chapter.

As an example there may be a conceptual difficulty in understanding the meaning of nested expressions of the type $\langle\langle A\rangle\rangle \ldots \langle\langle B\rangle\rangle\Phi$, especially, when the coalitions A and B share common agents. For instance, what exactly should $\langle\langle A\rangle\rangle\neg\langle\langle A\rangle\rangle\Phi$ mean?

This problem is related to a technical problem built in the semantics of ATL*, where e.g., in the truth evaluation of a formula of the type $\langle\langle A\rangle\rangle \ldots \langle\langle B\rangle\rangle\Phi$ the strategy for A adopted to guarantee the success of the goal $\ldots \langle\langle B\rangle\rangle\Phi$ does not have any effect when evaluating the truth of the subgoal $\langle\langle B\rangle\rangle\Phi$. This, arguably, goes against the intuitive understanding of what a strategy and its execution mean.

Such problems have lead to several proposals of alternative semantics for ATL*, with irrevocable commitment to strategies [ÅGJ07] or with strategy contexts, explicitly controllable within the formulas [Bri+09].

From a practical point of view it is also problematic that the computational complexity of ATL* is very high: it is 2ExpTime-complete for both the model checking [AHK02] and the satisfiability [Sch08] problems.

Thus, there are several good reasons to consider flat fragments of ATL*, where nesting of strategic quantifiers and temporal operators is restricted or completely disallowed, thus avoiding the problems listed above at the cost of reduced expressiveness.

There are two natural kinds of 'flatness' in the language of ATL*: with respect to the temporal operators and with respect to strategic quantifiers. The former comes naturally from purely temporal logics and has been investigated before, see e.g., [Hal95], [DS02], and [SV10] from a more general, coalgebraic perspective. Here we will mainly consider the latter type of flatness.

Contribution

The objective of this chapter is to develop optimal algorithms for solving the satisfiability problem for the variety of naturally definable flat fragments of ATL* and to analyze their computational complexity. Our main results and the contributions of this chapter are as follows:

1. Satisfiability of ATL* where nesting between strategic quantifiers is not allowed is PSpace-complete

2. Satisfiability in the flat fragments of ATL and ATL$^+$ are shown to be Σ_3^P-complete

3. For all logics considered, already the fragments with a nesting depth 2 of strategic quantifiers (respectively, path quantifiers) the satisfiability problem is as hard as in the full logics. Indeed, for a given formula an equi-satisfiable formula with nesting depth 2 and linear size can be computed in linear time.

Compared to 2ExpTime-completeness of satisfiability in ATL* [Sch08] and ATL$^+$ [JL03] and ExpTime-completeness in ATL [Dri03; GD06] the complexity for flat fragments are significantly lower.

We also provide complexity results for flat fragments of the branching-time logics CTL, CTL$^+$ and CTL* where, analogously to strategic quantifiers, path quantifiers cannot be nested. The results are collected in Figure 7.4.

Outline

The structure of the chapter is as follows: In Section 7.2 we introduce the satisfiability problem and various flat fragments of ATL* and discuss their expressiveness. Section 7.3 contains the technical preparation for our algorithms, where we introduce several normal forms for ATL* formulas and obtain some key technical results. In Section 7.5 we provide sound and complete decision procedures as well as matching lower bounds for the flat fragments of ATL*. We end with a summary in Section 7.6.

7.2 Flat fragments

In this section we introduce the flat fragments of temporal logics that will be analyzed for the remainder of the chapter and provide preliminary results for the satisfiability problem.

7.2.1 Flat fragments of LTL, CTL, CTL*

We define the flat fragments LTL_1, CTL_1 and CTL^*_1 respectively as subsets of LTL, CTL and CTL*. In LTL_1 no nesting of temporal operators is allowed. $\mathrm{CTL}^*_1, \mathrm{CTL}^+_1$ and CTL_1 are the fragments of $\mathrm{CTL}^*, \mathrm{CTL}^+$ and CTL where no nesting of path quantifiers is allowed. They are generating by the following grammar

$$\begin{array}{ll}
\mathrm{LTL}_1: & \theta ::= p \mid \neg\theta_1 \mid \theta_1 \wedge \theta_2 \mid \mathbf{X}\beta_1 \mid \beta_1 \mathbf{U} \beta_2 \\
\mathrm{CTL}_1: & \psi ::= p \mid \neg\psi_1 \mid \psi_1 \wedge \psi_2 \mid \mathbf{AX}\beta_1 \mid \mathbf{A}(\beta_1 \mathbf{U} \beta_2) \mid \mathbf{A}(\beta_1 \mathbf{R} \beta_2) \\
\mathrm{CTL}^+_1: & \vartheta ::= p \mid \neg\vartheta_1 \mid \vartheta_1 \wedge \vartheta_2 \mid \mathbf{A}\theta_1 \\
\mathrm{CTL}^*_1: & \varphi ::= p \mid \neg\varphi_1 \mid \varphi_1 \wedge \varphi_2 \mid \mathbf{A}\eta_1
\end{array}$$

where p is a proposition, β_1 and β_2 are Boolean formulas, θ_1 and θ_2 are LTL_1 formulas, η_1 is an LTL formula, ψ_1 and ψ_2 are CTL_1 formulas, ϑ_1 and ϑ_2 are CTL^+_1 formulas and φ_1 and φ_2 are CTL^*_1 formulas. The temporal operators **F** and **G** are defined as usual, for details see Section 2.2.1.

The following are examples of flat and non-flat formulas:

- $p\mathbf{U}q \wedge \mathbf{X}(r \wedge (q \wedge \neg p))$ is in LTL_1 but $p\mathbf{U}(\mathbf{X}q)$ is not.
- $\mathbf{A}(\neg p\mathbf{U}(p \wedge \neg q)) \wedge \neg(\mathbf{EF}(q \wedge \neg p) \wedge \neg\mathbf{AF}\neg(p \wedge q))$ is in CTL_1 (and in CTL^*_1).
- $\mathbf{AGF}p$ is in CTL^*_1 but not in CTL_1; $\mathbf{AGEF}p$ is neither in CTL_1 nor in CTL^*_1.

7.2.2 A hierarchy of flat fragments of ATL*

Here we define some *flat* fragments of ATL* and ATL. Flatness generally means no nesting of non-Boolean operators. There are two natural notions of flatness in the languages of ATL and ATL*: with respect to temporal operators and with respect to strategic quantifiers. We will be mostly concerned with the latter, but the former also applies in the case of ATL.

For the remainder of this chapter we adopt the following notational conventions: we will typically denote Boolean formulas by β, γ; LTL formulas by θ, η, ζ; ATL formulas by φ, ψ; and ATL* formulas – both state and path – by Θ, Φ, Ψ; all possibly with indices.

We will consider the following fragments of ATL*, where p is an atomic proposition, $A \subseteq \Pi$ is a coalition and θ is an LTL formula:

1. **Separated ATL*,** denoted $\mathrm{ATL}^*_{\mathrm{Sep}}$, consists of those formulas of ATL* in which there is no nesting of strategic quantifiers in the scope of temporal operators (but, any nesting of temporal operators within strategic quantifiers or temporal operators is allowed), so the (external) strategic and the (internal) temporal layers are separated. More precisely, the formulas of $\mathrm{ATL}^*_{\mathrm{Sep}}$ are generated as follows:
$$\Phi ::= \theta \mid \neg\Phi_1 \mid \Phi_1 \wedge \Phi_2 \mid \langle\langle A\rangle\rangle\Phi_1$$
where Φ_1 and Φ_2 are $\mathrm{ATL}^*_{\mathrm{Sep}}$ formulas.

2. **Full (strategically) flat ATL*,** denoted ATL^*_1, consists of those formulas of ATL* in which there is no nesting of strategic quantifiers within strategic quantifiers (but, nesting of strategic quantifiers and temporal operators in temporal operators is allowed), formally generated as follows:
$$\Phi ::= p \mid \neg\Phi_1 \mid \Phi_1 \wedge \Phi_2 \mid \langle\langle A\rangle\rangle\theta \mid \mathbf{X}\Phi_1 \mid \Phi_1\mathbf{U}\Phi_2 \mid \Phi_1\mathbf{R}\Phi_2$$
where Φ_1 and Φ_2 are ATL^*_1 formulas. If the restriction $A \neq \emptyset$ is imposed, we denote the resulting fragment $\widehat{\mathrm{ATL}^*_1}$.

3. **State fragment of ATL$^*_1$,** denoted $\mathrm{St}(\mathrm{ATL}^*_1)$, consists of the state formulas of ATL^*_1, i.e. those formulas of ATL* in which there is no nesting of strategic quantifiers in either temporal operators or strategic quantifiers (but, nesting between temporal operators is allowed). The formulas of $\mathrm{St}(\mathrm{ATL}^*_1)$ are explicitly generated as follows:
$$\Phi ::= p \mid \neg\Phi_1 \mid \Phi_1 \wedge \Phi_2 \mid \langle\langle A\rangle\rangle\theta.$$
where Φ_1 and Φ_2 are $\mathrm{St}(\mathrm{ATL}^*_1)$ formulas.

4. **Flat ATL$^+$** (or, **double-flat ATL***), denoted ATL^+_1, consists of those formulas of ATL^+ which are also in $\mathrm{St}(\mathrm{ATL}^*_1)$, e.g., with no nesting of either strategic quantifiers or temporal operators within temporal operators. The formulas of ATL^+_1 are generated as follows:
$$\Phi ::= p \mid \neg\Phi_1 \mid \Phi_1 \wedge \Phi_2 \mid \langle\langle A\rangle\rangle\theta_1$$
where θ_1 is an LTL_1 formula and Φ_1 and Φ_2 are ATL^+_1 formulas.

5. **Flat ATL,** denoted ATL_1, consists of those formulas of ATL^+_1 which are in ATL, i.e., in which strategic quantifiers are followed immediately by temporal operators. The formulas of ATL_1 are generated as follows:
$$\varphi ::= p \mid \neg\varphi_1 \mid \varphi_1 \wedge \varphi_2 \mid \langle\langle A\rangle\rangle\mathbf{X}\beta_1 \mid \langle\langle A\rangle\rangle(\beta_1\mathbf{U}\beta_2) \mid \langle\langle A\rangle\rangle(\beta_1\mathbf{R}\beta_2)$$
where β_1 and β_2 are Boolean formulas and φ_1 and φ_2 are ATL_1 formulas.

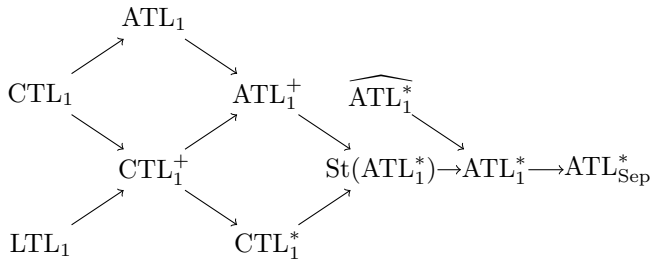


Figure 7.1: Inclusions between flat fragments. An arrow from $\mathcal{L}_1$ to $\mathcal{L}_2$ means that every $\mathcal{L}_1$ formula is an $\mathcal{L}_2$ formula.

Inclusions between the different flat fragments are illustrated in Figure 7.1. All inclusions shown in the figure are strict and there are no inclusions except the ones shown (where transitive closure is implicit). For example:

- $(\langle\langle 1\rangle\rangle \mathbf{G}\neg p \wedge \neg\langle\langle 2\rangle\rangle \mathbf{X}(p \vee \neg q)) \vee \langle\langle 1,2\rangle\rangle (p\mathbf{U}\neg q)$ is in ATL_1;
- $\langle\langle 1\rangle\rangle(\mathbf{G}\neg p \wedge \mathbf{F}q)$, $\neg\langle\langle 1,2\rangle\rangle((p\mathbf{R}\neg q) \vee (\neg p\mathbf{U}q))$ are in ATL_1^+ but not in ATL_1;
- $\langle\langle 1\rangle\rangle(\mathbf{GF}\neg p \vee \mathbf{X}\neg(p\mathbf{U}\neg q))$ is in $\mathrm{St}(\mathrm{ATL}_1^*)$ but not ATL_1^+;
- $\mathbf{G}\langle\langle 1,2\rangle\rangle \mathbf{F}(\neg p \vee q)$ is in $\widehat{\mathrm{ATL}_1^*}$ but not in $\mathrm{St}(\mathrm{ATL}_1^*)$;
- $\langle\langle\emptyset\rangle\rangle \mathbf{G}p \wedge \mathbf{G}\langle\langle 1,2\rangle\rangle \mathbf{GF}\neg p$ is in ATL_1^* but not in $\widehat{\mathrm{ATL}_1^*}$;
- $\langle\langle 2\rangle\rangle\langle\langle 1\rangle\rangle\neg(p\mathbf{U}\neg q)$ is in $\mathrm{ATL}_{\mathrm{Sep}}^*$ but not in ATL_1^*.

7.2.3 Some remarks on the expressiveness of the flat ATL*-fragments

The lower complexity of the satisfiability in the flat fragments of ATL* comes with a price, namely that various properties that require nesting of strategic quantifiers cannot be expressed anymore. However, many interesting and important properties of systems are still expressibe. For instance:

- $\langle\langle \texttt{ctrl}\rangle\rangle \mathbf{G}\neg\texttt{break}$ in ATL_1 specifies that a controller can make sure the system does not break no matter how the environment behaves,
- $\bigwedge_{i=1}^{n}\langle\langle \texttt{proc}_i\rangle\rangle \mathbf{GF}\texttt{db\_access}_i$ in ATL_1^* expresses that each process can ensure database access infinitely often,
- $\langle\langle A\rangle\rangle(\theta_{\mathrm{fair}} \rightarrow \theta)$ in ATL_1^* means that coalition A can make sure that the LTL property θ is satisfied on all fair paths (where fairness is defined by LTL formula θ_{fair}).

The semantics of ATL* is based on *unbounded memory strategies*, but it can be restricted and parameterized with the amount of memory that the proponent agents' strategies can use. The extreme case is the *memoryless semantics*, where the proponents may only use memoryless strategies. It turns out that satisfiability in ATL, is unaffected by such restrictions, but differences occur in ATL* and even in ATL$^+$. For discussion on these see e.g., [BJ13].

In contrast, using the satisfiability procedures developed in Section 7.5, we will show that all semantics based on different memory restrictions yield the same satisfiable (resp., the same valid) formulas in the flat fragment ATL$^*_1$.

7.2.4 The satisfiability problem

We focus on the *satisfiability problem* in this chapter where we distinguish between the *state satisfiability*statesatState satisfiability and *path satisfiability* problems which are defined on a given fragment $\mathcal{L}$ of ATL* as follows:

- Given a state formula φ in $\mathcal{L}$, does there exist a concurrent game $\mathcal{M}$ and a state s in $\mathcal{M}$ such that $\mathcal{M}, s \models \varphi$?
- Given a path formula Φ in $\mathcal{L}$, does there exist a concurrent game $\mathcal{M}$ and a play ρ in $\mathcal{M}$ such that $\mathcal{M}, \rho \models \Phi$?

Note that there are two variants of the satisfiability problem for formulas of ATL*: *tight*, where it is assumed that all agents in the model are mentioned in the formula, and *loose*, where additional agents, not mentioned in the formula, are allowed in the model. It is easy to see that these variants are really different, but the latter one is immediately reducible to the former, by adding just one extra agent a to the language. Furthermore, this extra agent can be easily added superfluously to the formula, e.g., by adding a conjunct $\langle\langle a \rangle\rangle \mathbf{X} \top$, so we hereafter only consider the tight satisfiability version. For further details and discussion on this issue, see e.g., [GS09; Wal+06].

Even though ATL$^*_1$ is included in ATL$^*_{\mathrm{Sep}}$ they have the same expressive power and there is an efficient translation from ATL$^*_{\mathrm{Sep}}$ to ATL$^*_1$.

Proposition 7.1 *Every formula of* ATL$^*_{\mathrm{Sep}}$ *is logically equivalent to a formula of* ATL$^*_{\mathrm{Sep}}$ *which is at most as long and has no nesting of strategic quantifiers. Such a formula is effectively computable in linear time.*

Proof. Because $\langle\langle A \rangle\rangle \Phi \equiv \Phi$ for every state formula Φ and coalition A. □

Thus, deciding satisfiablity in ATL$^*_{\mathrm{Sep}}$ is reducible with no cost to satisfiablity in St(ATL$^*_1$), so we will not discuss ATL$^*_{\mathrm{Sep}}$ hereafter. On the other hand, due to the equivalence above, the fragment ATL$^*_1$ can be extended even further by allowing nesting of strategic quantifiers, as long as there are no occurrences of temporal operators in between them.

The equivalence of $\langle\langle A\rangle\rangle\Phi$ and Φ for state formulas Φ is an example of why nesting of strategic quantifiers in ATL* can be considered unnatural. We note that this phenomenon is avoided in ATL* with strategy context [Bri+09].

Between the full logics and their flat fragments, it is natural to consider the hierarchies of fragments with a bounded nesting depth of strategic quantifiers. However, the next result shows that the fragments with nesting depth 2 are essentially as expressive and computationally hard as the full logics.

Theorem 7.2 *For any logic* $\mathcal{L} \in \{\mathrm{LTL}, \mathrm{CTL}, \mathrm{CTL}^+, \mathrm{CTL}^*, \mathrm{ATL}, \mathrm{ATL}^+, \mathrm{ATL}^*\}$ *and formula* Φ *of* $\mathcal{L}$ *there is an equi-satisfiable formula* Φ' *in* $\mathcal{L}$ *with nesting depth 2 of strategic quantifiers (resp., temporal operators for* LTL*) and length* $|\Phi'| = O(|\Phi|)$ *that can be computed in linear time.*

Proof. In each of the cases, the flattening is done by repeated renaming of state subformulas with fresh atomic propositions. We illustrate the technique on ATL*. Let Φ be an ATL* formula. For any innermost subformula Ψ of Φ beginning with a strategic quantifier we introduce a fresh atomic proposition p_Ψ. Then Φ and $\Phi' = \Phi[p_\Psi/\Psi] \wedge \mathbf{AG}(p_\Psi \leftrightarrow \Psi)$ are equi-satisfiable. By repeated application of such renaming of strategically quantified subformulas we obtain an equi-satisfiable formula of nesting depth 2 that is linear in the size of Φ. Since $\mathbf{AG}(p_\Psi \leftrightarrow \Psi)$ is a CTL formula, this works for each logic $\mathcal{L} \in \{\mathrm{CTL}, \mathrm{CTL}^+, \mathrm{CTL}^*, \mathrm{ATL}, \mathrm{ATL}^+, \mathrm{ATL}^*\}$, while for LTL we use $\mathbf{G}(p_\Psi \leftrightarrow \Psi)$. □

Thus, when restricting the nesting depth of formulas, we only have something to gain complexity-wise by considering a nesting depth of 1 as a nesting depth of 2 already give the same computational complexity as for the full fragments.

7.3 Normal forms and satisfiability of special sets

In this section we introduce several types of normal forms which will be used when we consider algorithms for the satisfiability problems.

7.3.1 Negation normal form of ATL* formulas

The first normal form we introduce is *negation normal form.*

Definition 7.3 *An* ATL* *formula* Φ *is in a* negation normal form *(NNF) if negations in* Φ *may only occur immediately in front of atomic propositions.*

We now define the dual $[[\varphi]]$ to the strategic quantifier $\langle\langle\varphi\rangle\rangle$ as usual. It reads: "There is no strategy for coalition A which ensures that φ is not satisfied along the play". Formally it is defined as follows:

$$[[A]]\varphi := \neg\langle\langle A\rangle\rangle\neg\varphi$$

If we consider $[[\cdot]]$ as a primitive operator in ATL*, then every ATL* formula can be transformed to an equivalent formula in NNF by driving all negations inwards, using the self-duality of $\mathbf{X}$ and the duality between $\mathbf{U}$ and $\mathbf{R}$. However, using $[[\cdot]]$ formally breaks the syntax of the fragments ATL and ATL$_1$ because of inserting a $\neg$ between $\langle\!\langle\cdot\rangle\!\rangle$ and the temporal operator. Yet, this can be easily fixed by equivalently re-defining the applications of $[[\cdot]]$, using the following equivalences:

- $[[A]]\mathbf{X}\varphi \equiv \neg\langle\!\langle A\rangle\!\rangle\mathbf{X}\neg\varphi$
- $[[A]](\varphi\mathbf{U}\psi) \equiv \neg\langle\!\langle A\rangle\!\rangle((\neg\varphi)\mathbf{R}(\neg\psi))$
- $[[A]](\varphi\mathbf{R}\psi) \equiv \neg\langle\!\langle A\rangle\!\rangle((\neg\varphi)\mathbf{U}(\neg\psi))$

Hereafter we assume that the language ATL* and each of its fragments introduced above are formally extended with the operator $[[\cdot]]$ applied just like $\langle\!\langle\cdot\rangle\!\rangle$ in the respective fragments. Due to the equivalences above, the resulting extensions preserve the expressiveness of these fragments. Formally:

Lemma 7.4 *Every formula of* ATL* *extended with the operator* $[[\cdot]]$ *can be transformed to an equivalent formula in NNF. Furthermore, each of the fragments* ATL, ATL$_1$, ATL$_1^+$, St(ATL$_1^*$) *and* ATL$_1^*$, *extended with* $[[\cdot]]$, *is closed under this transformation, i.e. if a formula is in any of these fragments then its NNF-equivalent formula is in that fragment, too.*

7.3.2 Successor normal forms

To define *successor normal form* we first need a few other definitions.

Definition 7.5 (Successor formulas) *An* ATL* *formula is a* successor formula (SF) *if it is of the type* $\langle\!\langle A\rangle\!\rangle\boldsymbol{X}\Phi$ *or* $[[A]]\boldsymbol{X}\Phi$.

Definition 7.6 (Components) *With every set*

$$\Gamma = \{\langle\!\langle A_0\rangle\!\rangle\boldsymbol{X}\Phi_0, \ldots, \langle\!\langle A_{m-1}\rangle\!\rangle\boldsymbol{X}\Phi_{m-1}, [[B_0]]\boldsymbol{X}\Psi_0, \ldots, [[B_{n-1}]]\boldsymbol{X}\Psi_{n-1}\}$$

of ATL* *successor formulas we associate the set of its*

- $\langle\!\langle\cdot\rangle\!\rangle\mathbf{X}$-components*:* $\langle\!\langle\cdot\rangle\!\rangle\boldsymbol{X}(\Gamma) = \{\Phi_0, \ldots, \Phi_{m-1}\}$,
- $[[\cdot]]\mathbf{X}$-components*:* $[[\cdot]]\boldsymbol{X}(\Gamma) = \{\Psi_0, \ldots, \Psi_{n-1}\}$,
- successor components*:* $SC(\Gamma) = \langle\!\langle\cdot\rangle\!\rangle\boldsymbol{X}(\Gamma) \cup [[\cdot]]\boldsymbol{X}(\Gamma)$.

Definition 7.7 (Successor normal form) *Formulas in successor normal form are defined as follows:*

1. *An* LTL *formula is in an* LTL successor normal form *(LSNF) if it is in NNF and is a Boolean combination of literals and successor formulas, i.e.,* LTL *formulas beginning with* $\boldsymbol{X}$.

2. *An* ATL* *formula is in a* successor normal form *(SNF) if it is in NNF and is a Boolean combination of literals and* ATL* *successor formulas.*

Lemma 7.8 *Every* LTL*-formula* ζ *can be effectively transformed to an equivalent formula in* LTL *successor normal form* $LSNF(\zeta)$*, of length at most* $6|\zeta|$.

Proof. We can assume that ζ is already transformed to NNF (of length less than twice the original length). Consider all maximal subformulas of ζ of the types $(\theta \mathbf{U} \eta)$ and $(\theta \mathbf{R} \eta)$. Replace each of them with its LTL-equivalent fixpoint unfolding, respectively $\eta \vee (\theta \wedge \mathbf{X}(\theta \mathbf{U} \eta))$ and $\eta \wedge (\theta \vee \mathbf{X}(\theta \mathbf{R} \eta))$. Then, the same procedure is applied recursively to all respective subformulas θ, η occurring above and not in the scope of $\mathbf{X}$, until all occurrences of $\mathbf{U}$ and $\mathbf{R}$ get in the scope of $\mathbf{X}$. This procedure at most triples the length of the starting formula and the result is clearly a formula in LSNF. □

Definition 7.9 (Conjunctive formulas in SNF) *An* ATL* *formula in SNF is* conjunctive *if it is of the form*

$$\Theta = \Phi \wedge \langle\langle A_0 \rangle\rangle \boldsymbol{X}\Phi_0 \wedge \ldots \wedge \langle\langle A_{m-1} \rangle\rangle \boldsymbol{X}\Phi_{m-1} \wedge [[B_0]] \boldsymbol{X}\Psi_0 \wedge \ldots \wedge [[B_{n-1}]] \boldsymbol{X}\Psi_{n-1}$$

where Φ *is a boolean combination of literals.*

With every such formula Θ *we associate the set of its* successor conjuncts*:*

$$C(\Theta) = \{\langle\langle A_0 \rangle\rangle \boldsymbol{X}\Phi_0, \ldots, \langle\langle A_{m-1} \rangle\rangle \boldsymbol{X}\Phi_{m-1}, [[B_0]] \boldsymbol{X}\Psi_0, \ldots, [[B_{n-1}]] \boldsymbol{X}\Psi_{n-1}\}$$

7.4 Sets of distributed control of ATL* formulas

In this section we consider *sets of distributed control* which will be useful for us when considering the satisfiability question. They are defined as follows.

Definition 7.10 (Set of distributed control) *A set of* ATL* *formulas* Δ *is a* set of distributed control *if* $\Delta = \{\langle\langle A_0 \rangle\rangle \Phi_0, \ldots, \langle\langle A_{m-1} \rangle\rangle \Phi_{l-1}, [[B]]\Psi\}$ *where the coalitions* $A_0, \ldots, A_{l-1}$ *are pairwise disjoint, and* $A_0 \cup \ldots \cup A_{l-1} \subseteq B$.

Lemma 7.11 *A set of* ATL* *successor formulas*

$$\Gamma = \{\langle\langle A_0 \rangle\rangle \boldsymbol{X}\Phi_0, \ldots, \langle\langle A_{m-1} \rangle\rangle \boldsymbol{X}\Phi_{m-1}, [[B_0]] \boldsymbol{X}\Psi_0, \ldots, [[B_{n-1}]] \boldsymbol{X}\Psi_{n-1}, [[\Pi]] \boldsymbol{X}\top\}$$

is satisfiable if and only if every subset of distributed control Δ *of* Γ *has a satisfiable set of successor components.*

PROOF. First, note that the formula $[[\Pi]]\mathbf{X}\top$ is valid, so it plays no role in the satisfiability of Γ; it is only added there in order to enable sufficiently many subsets of distributed control.

Now, suppose Γ is true at a state s of a concurrent game $\mathcal{M}$. Then for every subset of distributed control $\Delta = \{\langle\langle A_0\rangle\rangle\mathbf{X}\Phi_0, \ldots, \langle\langle A_{l-1}\rangle\rangle\mathbf{X}\Phi_{l-1}, [[B]]\mathbf{X}\Psi\}$ consider collective actions for the coalitions $A_0, \ldots, A_{l-1}$ at s that guarantee satisfaction of their respective nexttime objectives in Δ in any of the resulting successor states. Add arbitrarily fixed actions of the remaining agents in B and a respective collective action for $A \setminus B$ dependent on the so fixed actions of the agents in B, that brings about satisfaction of Ψ in the resulting successor state s'. Then all successor components of Δ are true at s'.

Conversely, suppose that $\Delta_1, \ldots, \Delta_d$ are all subsets of Γ of distributed control and they are all satisfiable. For each Δ_i we fix a concurrent game $\mathcal{M}_i$ and a state s_i in it that satisfies $SC(\Delta_i)$. We can assume, w.l.o.g., that $\mathcal{M}_i$ is generated from s_i, i.e. consists only of states reachable by plays starting at s_i.

We will construct a concurrent game satisfying Γ by using a construction from [GD06]. The idea is to first create a root state s and supply all agents with sufficiently many actions at s in order to ensure the existence of all collective actions and respective successor states necessary for satisfying the successor components of Γ. We will show that it suffices to take care of the sets of successor components of each subset of distributed control Γ and then will use the concurrent games satisfying these to complete the construction of the model satisfying Γ.

Now, the construction. Recall that $|\Pi| = k$ and let $r = m+n$ (the numbers of $\langle\langle\cdot\rangle\rangle$- and $[[\cdot]]$-components in Γ). Each agent will have r available actions $\{0, \ldots, r-1\}$ at the root state s, hence $\{0, \ldots, r-1\}^k$ is the set of all possible action profiles at s. The intuition is that every agent's action at s is a choice of that agent of a formula from Γ for the satisfaction of which the agent chooses to act. For every such action profile σ we denote by $N(\sigma)$ the set of agents $\{i \mid \sigma_i \geq m\}$ and then we define the number $\mathbf{neg}(\sigma)$ to be the remainder of $[\sum_{i\in N(\sigma)}(\sigma_i - m)]$ modulo n. (The idea of this definition is that, once all agents in any given proper subset of $N(\sigma)$ choose their actions, the remaining agents in $N(\sigma)$ can act accordingly to yield any value of $\mathbf{neg}(\sigma)$ between 0 and $n-1$ they wish, i.e., to set the "collective action" of all agents in $\mathbf{neg}(\sigma)$ on any $[[\cdot]]\mathbf{X}$-formula in Γ they choose.) Now, we consider the set

$$\Delta_\sigma = \{\langle\langle A_j\rangle\rangle\mathbf{X}\Phi_j \mid j < m \text{ and } \sigma_i = j \text{ for all } i \in A_j\} \cup \{[[B_l]]\mathbf{X}\Psi_l \mid \mathbf{neg}(\sigma) = l \text{ and } \Pi \setminus B_l \subseteq N(\sigma)\}$$

Note that Δ_σ is a subset of Γ of distributed control if it contains a formula $[[B_j]]\mathbf{X}\Psi$, or else can be made a set of distributed control by adding $[[\Pi]]\mathbf{X}\top$ to it. Indeed, all agents in a A_j choose j, so all coalitions A_j must be pairwise disjoint. Besides, if $[[B_l]]\mathbf{X}\Psi_l \in \Delta_\sigma$ then it is clearly a unique $[[\cdot]]$-formula in Δ_σ and no agents from any $A_j \in \Delta_\sigma$ are in $N(\sigma)$, hence $A_j \subseteq B_l$ for each $\langle\langle A_j\rangle\rangle\mathbf{X}\Phi_j \in \Delta_\sigma$. Thus, Δ_σ is one of $\Delta_1, \ldots, \Delta_d$, say Δ_i. Then, we determine the successor state $\delta(s, \sigma)$ to be s_i. To complete the definition of the concurrent game, at each successor state s_i of s we graft a copy of $\mathcal{M}_i$.

We will show that the resulting concurrent game $\mathcal{M}$ satisfies Γ at s. Indeed, for every $\langle\langle A_j\rangle\rangle\mathbf{X}\Phi_j \in \Gamma$ a collective strategy for A_j that guarantees the satisfaction of that formula at s consists in all agents from A_j acting j at s, following their strategy that guarantees in $\mathcal{M}_i$ the satisfaction of the objective Φ_j if the play enters the copy of $\mathcal{M}_i$, and acting in an arbitrarily fixed manner at all other states of $\mathcal{M}$. (Note that, if the strategy for A_j in $\mathcal{M}_i$ is positional, then the above described strategy is positional, too.) Lastly, every $[[B_l]]\mathbf{X}\Psi_l \in \Gamma$ is true at s, too, because if $B_l \neq \Pi$ then for every collective action of all agents from $[[B_l]]\mathbf{X}$ there is a suitable complementary action of $\Pi \setminus B_l$, where all agents choose actions greater than m and such that $\mathbf{neg}(\sigma)$ adds up to l modulo n. (In fact, this can be guaranteed by any agent in $\Pi \setminus B_l$ after all others have chosen their actions.) In the case when $B_l = \Pi$, every subset $\{\langle\langle A_j\rangle\rangle\mathbf{X}\Phi_j, [[B_l]]\mathbf{X}\Psi_l\}$ for $j < m$ is of distributed control, hence Ψ_l is true at the root s_i of $\mathcal{M}_i$ for each $i = 1, ..., d$. Thus, $\mathcal{M}, s \models \Gamma$, which completes the proof. □

A consequence of the proof above is that memoryless and memory-based semantics yield the same satisfiable state formulas in the flat fragments.

Corollary 7.12 *A* $\mathrm{St}(\mathrm{ATL}_1^*)$ *formula* Φ *is satisfiable in the memoryless semantics if and only if it is satisfiable in the memory-based semantics.*

Proof. Lemma 7.11 can be proved for memoryless semantics in the same way, but only for $\mathrm{St}(\mathrm{ATL}_1^*)$ formulas. This is because the successor components are LTL formulas which have the same semantics with and without memory. Further, for both semantics each subformula $\langle\langle A\rangle\rangle\theta$ or $[[A]]\theta$ of Φ with a strategic quantifier as main connective can be converted to SNF by converting θ to LSNF using Lemma 7.8. Then, we can use the memoryless and memory-based version of Lemma 7.11 and obtain that Φ is satisfiable in the memory-based semantics if and only if it is satisfiable in the memoryless semantics since the satisfiable sets of successor components are the same for the two types of semantics. □

7.5 Optimal decision procedures for satisfiability

In this section we provide algorithms for satisfiability in the flat fragments of ATL^*. In addition, we provide matching lower bounds.

7.5.1 Centipede models and satisfiability in LTL_1, CTL_1 and CTL_1^+

Satisfiability of LTL_1 is analyzed in [DS02]. In particular, it is shown that if an LTL_1 formula θ is satisfiable then it is satisfiable in a model of the form $s_0 s_1 ... s_\ell^\omega$ where $\ell = |\theta|$. Consequently, it is shown that satisfiability of LTL_1 is NP-complete. We provide similar results for CTL_1^+ and CTL_1 here.

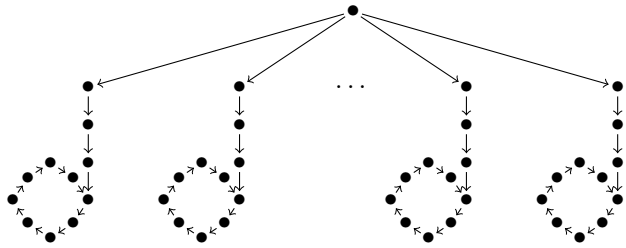


Figure 7.2: A centipede model.

Proposition 7.13 *If a* CTL_1^+ *formula* φ *has a model, then it has a model with at most* $O(|\varphi|^2)$ *states.*

Proof. Suppose $\mathcal{M}, s_0 \models \varphi$ for a CTL_1^+ formula φ, a model $\mathcal{M}$ and a state s_0. Assume w.l.o.g. that φ is in NNF. We generate another model $\mathcal{M}'$ with $O(|\varphi|^2)$ states and a state s_0' such that $\mathcal{M}', s_0' \models \varphi$.

Let Δ_Q be the set of subformulas of φ that has Q as main connective for $Q \in \{\mathbf{E}, \mathbf{A}\}$ and let Δ_{B} be the set of maximal Boolean subformulas of φ that do not occur in the scope of a path quantifier. For each $Z \in \{\mathbf{E}, \mathbf{A}, \mathsf{B}\}$ let $\Delta_Z^\top \subseteq \Delta_Z$ be the subsets satisfied in $\mathcal{M}, s_0$.

Now, for each $\mathbf{E}\psi \in \Delta_{\mathbf{E}}^\top$ let $\rho^\psi = \rho_0^\psi \rho_1^\psi ...$ be a path in $\mathcal{M}$ starting in s_0 such that $\rho^\psi \models \psi$. Since $\mathcal{M}, s_0 \models \varphi$ we have for every $\mathbf{A}\psi' \in \Delta_{\mathbf{A}}^\top$ that $\rho^\psi \models \psi'$ because ψ' is satisfied along all paths from s_0. Further, $\rho_0^\psi = s_0$ implies that $\rho^\psi \models \psi \wedge \bigwedge_{\psi' \in \Delta_{\mathbf{A}}^\top} \psi' \wedge \bigwedge_{\beta \in \Delta_{\mathsf{B}}^\top} \beta$. Since this is an LTL_1 formula of size at most $|\varphi|$ it has a model π^ψ of the form $\pi_0^\psi ... (\pi_{|\varphi|}^\psi)^\omega$ where π_0^ψ is labelled as s_0. Now, by gluing together each path π^ψ (which is made finite by adding a self-loop to the state $\pi_{|\varphi|}^\psi$) in the initial state s_0' we obtain a transition system $\mathcal{M}'$ such that $\mathcal{M}', s_0' \models \varphi$. Since $|\Delta_{\mathbf{E}}^\top| \leq |\varphi|$ there are at most $O(|\varphi|^2)$ states in $\mathcal{M}'$. □

Further, we will see that for satisfiable formulas of CTL_1^*, ATL_1 and $\mathrm{St}(\mathrm{ATL}_1^*)$ there are models that can be obtained by gluing together ultimately periodic paths as in the proof of Proposition 7.13. We call such models *centipede models*, illustrated in Figure 7.2. Note that these models only branch in the initial state.

However, for the flat fragments CTL_1^*, ATL_1, $\mathrm{St}(\mathrm{ATL}_1^*)$ models of polynomial size are not guaranteed to exist as for CTL_1^+. First, the length of the period and the prefix of the ultimately periodic paths can be exponential due to LTL subformulas in the case of CTL_1^* and $\mathrm{St}(\mathrm{ATL}_1^*)$. Second, in the cases of ATL_1 and $\mathrm{St}(\mathrm{ATL}_1^*)$ (but not for CTL_1^*) an exponential branching factor in the initial state may be forced by

a formula. Indeed, consider the following ATL_1 formula

$$\varphi = \bigwedge_{i=1}^{n} \langle\langle i \rangle\rangle \mathbf{X} p_i \wedge \langle\langle i \rangle\rangle \mathbf{X} \neg p_i$$

over the propositions $\{p_1, ..., p_n\}$ and players $\{1, ..., n\}$. For a state s_0 to satisfy this formula there has to be a successor state for each possible truth assignment to the propositions $\{p_1, ..., p_n\}$, of which there are 2^n. This phenomenon does not occur in the branching-time logic CTL_1^*.

Proposition 7.14 *Satisfiability in* CTL_1^+ *is* NP*-complete.*

Proof. NP-hardness follows from Boolean satisfiability. An NP-algorithm for CTL_1^+ works as follows. It takes as input a CTL_1^+ formula φ in NNF, hence a positive Boolean combination of flat CTL_1^+ state formulas, and guesses non-deterministically a centipede model $\mathcal{M}, s_0$ of size $O(|\varphi|^2)$ for φ, as well as the disjuncts in φ that evaluate to true at s_0. (According to Proposition 7.13, if φ has a model then it has a model of this form and size.) After guessing, it checks whether the resulting formula of the form $\varphi' = \beta \wedge \bigwedge_{i=0}^{\ell} \varphi_i'$ is true in the guessed model where β is a Boolean formula and each φ_i' is of the form $\mathbf{A}\theta_i$ or $\mathbf{E}\theta_i$ for an LTL_1 formula θ_i. First, the model-checking of β can be done in linear time. Next, for each of the $O(|\varphi|)$ formulas θ_i it can checked whether it is true in each of the $O(|\varphi|)$ paths of the centipede model in polynomial time since LTL model-checking of an ultimately periodic path of length $O(|\varphi|)$ can be done in polynomial time in $|\varphi|$ [MS03]. Thus, the guess can be verified in polynomial time due to the small model property of Proposition 7.13 and the centipede shape of the model. □

Since CTL_1 is included in CTL_1^+ and CTL_1 satisfiability also includes Boolean satisfiability we have the following corollary.

Corollary 7.15 *Satisfiability in* CTL_1 *is* NP*-complete.*

7.5.2 Lower bound for satisfiability in $\mathbf{ATL}_1$

In this section we show that satisfiability of flat ATL is hard for the complexity class Σ_3^P in the polynomial hierarchy.

Proposition 7.16 ATL_1*-SAT is* Σ_3^P*-hard.*

Proof. The proof is by reduction from the Σ_3^P-SAT problem, which is Σ_3^P-complete. This problem takes as input a quantified Boolean sentence

$$\gamma = \exists x_1, ..., x_m \forall x_{m+1}, ..., x_k \exists x_{k+1}, ..., x_n . \gamma'$$

where γ' is a Boolean formula over the Boolean variables $x_1, ..., x_n$. The problem is to decide whether γ is true.

Given γ, we construct an ATL_1 formula ψ over the set $\mathrm{AP} = \{x_1, ..., x_n\}$ of proposition symbols as follows

$$\psi = \bigwedge_{j=1}^{m} (\mathbf{AX}x_j \vee \mathbf{AX}\neg x_j) \wedge \bigwedge_{i=m+1}^{n} (\langle\!\langle \{i\} \rangle\!\rangle \mathbf{X}x_i \wedge \langle\!\langle \{i\} \rangle\!\rangle \mathbf{X}\neg x_i) \wedge \neg \langle\!\langle \{m+1, \ldots, k\} \rangle\!\rangle \mathbf{X}\neg\gamma'$$

We now claim that γ is true if and only if ψ is satisfiable.

The intuition behind this is as follows. Note first that the formula ψ only expresses properties of the second state during the play. That is, after the first choice of actions of all players in the game.

It is specified that every player i in $\{m+1, ..., n\}$ controls the value of the proposition x_i in the second state. It is also specified that the value of proposition x_j for $1 \leq j \leq m$ is the same in the second state of the game no matter what the players do. That is, these values are specific to a game satisfying this formula. Finally, it is specified that the coalition $\{m+1, ..., k\}$ do not have a strategy to make sure that γ' is false.

It is now possible to show that ψ is satisfiable if and only if there exists a game (with particular truth values of $x_1, ..., x_m$) such that values of $x_{m+1}, ..., x_k$ cannot be chosen by players $\{m+1, ..., k\}$ in a way such that no values of $x_{k+1}, ..., x_n$ (chosen by players $\{k+1, ..., n\}$) makes γ' true. In other words, there exists particular values of $x_1, ..., x_m$ such that for all values of $x_{m+1}, ..., x_k$ there exists values of $x_{k+1}, ..., x_n$ such that γ' is true. This intuition is now shown formally.

First, suppose that γ is true. Then we construct a concurrent game $\mathcal{M} = (S, \Pi, \Sigma, \Gamma, \delta, \mathrm{AP}, L)$ and a state $s_0 \in S$, such that $\mathcal{M}, s_0 \models \psi$, as follows. Let $\Pi = \{m+1, ..., n\}$, $S = \{s_0\} \cup \{s_{v_{m+1},...,v_n} \mid v_i \in \{0,1\} \text{ for } m+1 \leq i \leq n\}$ and $\Sigma = \{0,1\}$. Then, for every player $a \in \Pi$ define $\Gamma(s_0, a) = \{0,1\}$ and $\Gamma(s, a) = \{0\}$ for all $s \neq s_0$. The transitions are defined by $\delta(s_0, (v_{m+1}, \ldots v_n)) = s_{v_{m+1},...,v_n}$ and $\delta(s, \alpha_\Pi) = s$ for all $s \neq s_0$ and all action tuples α_Π. The set of proposition symbols is $\mathrm{AP} = \{x_1, \ldots, x_n\}$. The labelling is given by $L(s_0) = \emptyset$ and for every $i \in \{m+1, \ldots, n\}$ we have $x_i \in L(s_{v_{m+1},...,v_n})$ if and only if $v_i = 1$ when $s_{v_{m+1},...,v_n} \in S \setminus \{s_0\}$. Finally, let $x'_1, ..., x'_m$ be particular values such that $\forall x_{m+1}, \ldots, x_k \exists x_{k+1}, ..., x_n . \gamma'[x_1 \mapsto x'_1, \ldots, x_m \mapsto x'_m]$ is true. Such values exist since γ is true. For $1 \leq i \leq m$ and every $s_{v_{m+1},...,v_n} \in S \setminus \{s_0\}$, let $x_i \in L(s_{v_{m+1},...,v_n})$ if and only if $x'_i = 1$.

Intuitively, for all i such that $m+1 \leq i \leq n$ player i chooses the value of x_i in the successor state and then the play stays in that state forever. The value of x_i for $1 \leq i \leq m$ in the successor state is defined by the values $x'_1, ..., x'_m$. The subformula $\bigwedge_{i=m+1}^{n} (\langle\!\langle \{i\} \rangle\!\rangle \mathbf{X}x_i \wedge \langle\!\langle \{i\} \rangle\!\rangle \mathbf{X}\neg x_i)$ is clearly true at s_0. The same is the case for $\bigwedge_{j=1}^{m} (\mathbf{AX}x_j \vee \mathbf{AX}\neg x_j)$. Next, since γ is true when x_i takes the values x'_i for $1 \leq i \leq m$, then no matter which values of x_i are chosen by players in $\{m+1, ..., k\}$ there exists values of x_i for players in $\{k+1, ..., n\}$ such that γ' is true in the successor state. Thus, coalition $\{m+1, ..., k\}$ does not have a strategy to ensure that γ' is false in the successor state. Thus, $\mathcal{M}, s_0 \models \psi$.

For the converse direction, suppose that ψ is satisfied by some model $\mathcal{M}, s_0$, i.e. $\mathcal{M}, s_0 \models \psi$. For contradiction, suppose that γ is false. Then for all $x_1, ..., x_m$ there exists $x_{m+1}, ..., x_k$ such that γ' is false for all $x_{k+1}, ..., x_n$. In particular, this must be the case when x_i take the unique values x'_i for $1 \leq i \leq m$ that are true in all successors of s_0. These are unique since s_0 satisfies $\bigwedge_{j=1}^{m}(\mathbf{AX}x_j \vee \mathbf{AX}\neg x_j)$. In this case there exists particular values x'_i for $m+1 \leq i \leq k$ such that γ' is false for all $x_{k+1}, ..., x_n$ when x_i take the values x'_i for $m+1 \leq i \leq k$. Consider the strategy for coalition $\{m+1, ..., k\}$ that chooses these values for x_i in the successor state for $m+1 \leq i \leq k$. This strategy ensures that γ' is false in the successor state. However, this contradicts the fact that $\mathcal{M}, s_0 \models \neg\langle\langle\{m+1, \ldots, k\}\rangle\rangle\mathbf{X}\neg\gamma'$ which follows from $\mathcal{M}, s_0 \models \psi$. Thus, γ must be true. This completes the proof. □

Note that the hardness result only requires the use of the temporal operator **X** and neither **U** nor **R**. This is interesting since this lower bound will be shown to be an upper bound for the full ATL_1^+ in the following section. Thus, the $\langle\langle\cdot\rangle\rangle\mathbf{X}$ fragment of ATL_1 is as hard as the full ATL_1^+.

7.5.3 Deciding satisfiability in $\mathrm{St}(\mathrm{ATL}_1^*)$ and ATL_1^+

Here we provide algorithms for satisfiability of $\mathrm{St}(\mathrm{ATL}_1^*)$ and ATL_1^+. However, first we need a lemma.

Lemma 7.17 *Let $\Phi = \langle\langle A\rangle\rangle\Psi$ be an ATL^* formula and let $\mathrm{AP}(\Phi) = \{p_1, \ldots p_r\}$ be the set of atomic propositions occurring in Φ. Consider any mapping $v : \mathrm{AP}(\Phi) \to \{\top, \bot\}$ and let $v[\Phi]$ be the result of substitution of all occurrences of p_i in Φ which are not in the scope of a temporal operator by $v(p_i)$, for each $p_1, \ldots p_r$. Further, let*

$$\delta(v) := \bigwedge_{v(p_i)=\top} p_i \wedge \bigwedge_{v(p_i)=\bot} \neg p_i$$

Then, $\delta(v) \wedge \Phi \equiv \delta(v) \wedge v[\Phi]$.

Proof. Consider any concurrent game $\mathcal{M}$ and a state s in it. If $\delta(v)$ is false at s then both sides are false. Suppose $\mathcal{M}, s \models \delta(v)$. Then $\mathcal{M}, s \models v(p_i) \leftrightarrow p_i$ for each $p_1, \ldots p_r$. Then, Φ and $v[\Phi]$ are equally true or false at s, as they only differ in the occurrences of atomic propositions that are evaluated at s. □

Proposition 7.18 *Satisfiability testing is in*

1. PSpace *for* $\mathrm{St}(\mathrm{ATL}_1^*)$
2. Σ_3^P *for* ATL_1^+ *(and* ATL_1*)*

PROOF. The decision procedures for both $\mathrm{St}(\mathrm{ATL}_1^*)$ and ATL_1^+ will be essentially the same, but in their last phases they work in different computational complexities. First, consider an $\mathrm{St}(\mathrm{ATL}_1^*)$ formula Φ and let $\mathrm{AP}(\Phi) = \{p_1, \dots p_r\}$. The formula Φ is a Boolean combination of atomic propositions and subformulas of the type $\langle\langle A\rangle\rangle\theta$ where $\theta \in \mathrm{LTL}$. By Lemma 7.8, we can assume that each such θ is in a LSNF of linearly increased length, i.e., is a Boolean combination of atomic propositions and $\mathbf{X}$-formulas (formulas beginning with $\mathbf{X}$) of LTL. The algorithm now works as follows:

1. Guess a truth assignment τ for the atomic propositions in $\mathrm{AP}(\Phi)$ at a state s of a concurrent game satisfying Φ, if any. Consider the unique map $v : \mathrm{AP}(\Phi) \to \{\top, \bot\}$ for which $\delta(v)$ is true under τ. By Lemma 7.17, each maximal subformula $\langle\langle A\rangle\rangle\theta$ in Φ can be equivalently replaced by $v[\langle\langle A\rangle\rangle\theta]$, which is $\langle\langle A\rangle\rangle v[\theta]$.

2. After elementary Boolean simplifications (of the type $\top \wedge A \equiv A, \bot \wedge A \equiv \bot$, etc.) each $v[\theta]$ is transformed to a Boolean combination of $\mathbf{X}$-formulas only. Using the LTL validities $\mathbf{X}\eta \wedge \mathbf{X}\zeta \equiv \mathbf{X}(\eta \wedge \zeta)$ and $\mathbf{X}\eta \vee \mathbf{X}\zeta \equiv \mathbf{X}(\eta \vee \zeta)$, it is further equivalently transformed into an $\mathbf{X}$-formula which is at most as long.

Afterwards, the original formula is (non-deterministically) transformed to an equisatisfiable Boolean combination of ATL^* formulas of type $\langle\langle A\rangle\rangle\mathbf{X}\theta$ and $[[A]]\mathbf{X}\theta$.

3. Now, assuming that the resulting formula is satisfiable, we further guess the true disjuncts in every $\vee$-subformula in a satisfying concurrent game and reduce the problem to checking satisfiability of a conjunctive formula of the type

$$\Theta = \langle\langle A_0\rangle\rangle\mathbf{X}\theta_0 \wedge \dots \wedge \langle\langle A_{m-1}\rangle\rangle\mathbf{X}\theta_{m-1} \wedge [[B_0]]\mathbf{X}\eta_0 \wedge \dots \wedge [[B_{n-1}]]\mathbf{X}\eta_{n-1}$$

Let $D(\Theta)$ be the union of the set $C(\Theta)$ of conjuncts of Θ and $\{[[\Pi]]\mathbf{X}\top\}$, i.e.

$$D(\Theta) = \{\langle\langle A_0\rangle\rangle\mathbf{X}\theta_0, \dots, \langle\langle A_{m-1}\rangle\rangle\mathbf{X}\theta_{m-1}, [[B_0]]\mathbf{X}\eta_0, \dots, [[B_{n-1}]]\mathbf{X}\eta_{n-1}, [[\Pi]]\mathbf{X}\top\}$$

4. By Lemma 7.11, the set $D(\Theta)$ is satisfiable iff every subset of distributed control of it has a satisfiable set of successor components. Since each of them is a set of LTL formulas, these checks can be done using standard techniques.

Each check in step 4. of the algorithm can be done in PSPACE when Φ is a $\mathrm{St}(\mathrm{ATL}_1^*)$ formula, since each successor component is an LTL formula. In the case of ATL_1^+ the checks can be done in NP according to [DS02], as in this case each successor component is an LTL_1 formula. Hence, checking that each of the (possibly exponentially many) subsets of distributed control is satisfiable can be done in $\mathrm{coNP}^{\mathrm{PSPACE}} = \mathrm{PSPACE}$ for $\mathrm{St}(\mathrm{ATL}_1^*)$ and in $\mathrm{coNP}^{\mathrm{NP}}$ for ATL_1^+. Thus, the whole procedure can be done respectively in $\mathrm{NP}^{\mathrm{PSPACE}} = \mathrm{PSPACE}$ for $\mathrm{St}(\mathrm{ATL}_1^*)$ and in $\mathrm{NP}^{\mathrm{coNP}^{\mathrm{NP}}}$ for ATL_1^+, by guessing the true propositions in the initial state and the true disjuncts in Φ, and then applying resp. a PSPACE-oracle and $\mathrm{coNP}^{\mathrm{NP}}$-oracle. Since $\mathrm{NP}^{\mathrm{coNP}^{\mathrm{NP}}} = \Sigma_3^P$ the proof is completed. □

This result, combined with Proposition 7.16 and the PSPACE-hardness of LTL satisfiability, yields the following.

Theorem 7.19 *The satisfiability problem of*

1. $\mathrm{St}(\mathrm{ATL}_1^*)$ *is* PSPACE*-complete*
2. CTL_1^* *is* PSPACE*-complete*
3. ATL_1^+ *is* Σ_3^P*-complete*
4. ATL_1 *is* Σ_3^P*-complete*

Here is another consequence of the proof of Proposition 7.18:

Corollary 7.20 *Every satisfiable* $\mathrm{St}(\mathrm{ATL}_1^*)$ *formula* Φ *has a centipede model* $\mathcal{M}$ *with branching factor* $O(2^{|\Phi|})$ *in the root. Further, every ultimately periodic path in* $\mathcal{M}$ *has a prefix of length* $O(2^{|\Phi|})$ *and a period of length* $O(|\Phi| \cdot 2^{|\Phi|})$.

7.5.4 PSPACE decision procedure for the satisfiability in ATL_1^*

The decision procedure for $\mathrm{St}(\mathrm{ATL}_1^*)$ can be extended to a PSPACE-complete decision procedure for the whole ATL_1^*, by combining it with a PSPACE decision procedure for LTL and showing that every path-satisfiable ATL_1^* formula can be satisfied in a special type of concurrent games described below. The proof of the latter is rather lengthy (see brief discussion further), so we only state and prove here the easier case of the slightly smaller fragment $\widehat{\mathrm{ATL}_1^*}$. This is the fragment where no strategic quantifiers of the form $\langle\langle\emptyset\rangle\rangle$ (i.e, fully universal path quantifiers) are allowed. We only note that the procedure for the full ATL_1^* is essentially the same.

First, recall that every satisfiable LTL formula has an ultimately period linear model with prefix and period that both have length exponential in the size of the formula [SC85]. Further, according to Corollary 7.20, every satisfiable $\mathrm{St}(\mathrm{ATL}_1^*)$ formula can be satisfied at the root state of a centipede model of exponentially bounded number and length of legs. Combining these results leads to a new type of concurrent games which we call *Lasso of Centipedes (LoC) models*. Such models consist of an ultimately periodic path (the lasso) where each state is the root of a centipede model. An illustration of a model like this is shown in Figure 7.3.

Proposition 7.21 *Every satisfiable* $\widehat{\mathrm{ATL}_1^*}$ *formula* Φ *is satisfied in a LoC model with size bounded exponentially in* $|\Phi|$.

PROOF. Given an $\widehat{\mathrm{ATL}_1^*}$ formula Φ in NNF we define its LTL *skeleton* $\mathrm{Sk}_{\mathrm{LTL}}(\Phi)$ as follows: Let the state subformulas of Φ of type $\langle\langle A\rangle\rangle\theta$ or $[[A]]\theta$ be $\Psi_1, \ldots, \Psi_n$. For each of them Ψ we introduce a new (not in AP) atomic proposition p_Ψ. Then we

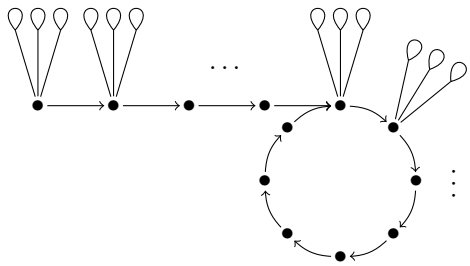


Figure 7.3: A Lasso of Centipedes (LoC) model.

produce the LTL formula $\widehat{\Phi}$ by replacing every occurrence of such a subformula Ψ in Φ by p_Ψ. Now, define

$$\mathrm{Sk}_{\mathrm{LTL}}(\Phi) ::= \widehat{\Phi} \wedge \bigwedge_{i}^{n} \mathbf{G}(p_{\Psi_i} \to \Psi_i)$$

We claim that any concurrent game $\mathcal{M}$ and a path π in it on which Φ is true can be expanded to a concurrent game $\widehat{\mathcal{M}}$ and a path $\widehat{\pi}$ in it satisfying $\mathrm{Sk}_{\mathrm{LTL}}(\Phi)$, by evaluating each new atomic proposition p_Ψ to be true at exactly those states of π at which Ψ is true in $\mathcal{M}$. Conversely, for any concurrent game $\mathcal{M}$ and a path π in it on which $\mathrm{Sk}_{\mathrm{LTL}}(\Phi)$ is true, the formula Φ is true on π, too, because all atomic propositions p_{Ψ_i} occur only positively in $\widehat{\Phi}$, so replacing them with the respective Ψ_i's will preserve the truth.

Thus, it suffices to show that if $\mathrm{Sk}_{\mathrm{LTL}}(\Phi)$ is path-satisfiable then it can be satisfied on the lasso path in some LoC model of size bounded exponentially in $|\Phi|$. Indeed, take any concurrent game $\mathcal{M}$ and a path π in it on which $\mathrm{Sk}_{\mathrm{LTL}}(\Phi)$ is true. Then, in particular, the path π alone is a linear model for $\widehat{\Phi}$. Now, take an ultimately periodic linear model $\widehat{\pi}$ of length bounded exponentially in $|\widehat{\Phi}|$, hence in $|\Phi|$. Such a model can be obtained from π by cutting its tail off at a suitable position and looping back to a suitable previous state. Thus, every state in $\widehat{\pi}$ has the label of a prototype state in π. Now, for every state $\widehat{s}$ on $\widehat{\pi}$, let s be its prototype in π. We do the following.

- Consider the set $\Gamma(s)$ of state subformulas Ψ of Φ such that p_Ψ is in the label of s in π. Since $\mathrm{Sk}_{\mathrm{LTL}}(\Phi)$ is true on π, every formula in $\Gamma(s)$ is true at s in $\mathcal{M}$. Thus, $\Gamma(s)$ is satisfiable, hence by Corollary 7.20, it can be satisfied at the root state of a centipede model $\mathcal{M}(\Gamma(s))$ of exponentially bounded in $|\Phi|$ number and length of legs.
- Now, we graft a copy of $\mathcal{M}(\Gamma(s))$ at the state $\widehat{s}$ in $\widehat{\pi}$ by identifying its root with $\widehat{s}$ and keeping all other states disjoint from $\widehat{\pi}$.
- Next, we add a special new action for every agent at the state $\widehat{s}$ and define the successor of the resulting action profile to be the successor of $\widehat{s}$ on the path $\widehat{\pi}$,

while every other action profile involving some (but not all) of these special new actions leads to a successor of $\widehat{s}$ in the grafted copy of $\mathcal{M}(\Gamma(s))$, chosen so as not to affect the truth of any of the formulas from $\Gamma(s)$ at $\widehat{s}$. We omit the easy but tedious details of this construction.

After completing this procedure for each state of $\widehat{\pi}$, the result is a LoC model $\widetilde{\mathcal{M}}$ which, by construction, satisfies the formula $\widehat{\Phi}$ on $\widehat{\pi}$ and satisfies at each state $\widehat{s}$ on $\widehat{\pi}$ the set $\Gamma(s)$. Therefore, $\widetilde{\mathcal{M}}, \widetilde{\pi} \models \mathrm{Sk}_{\mathrm{LTL}}(\Phi)$, hence $\widetilde{\mathcal{M}}, \widetilde{\pi} \models \Phi$. □

We briefly indicate the additional complication in extending this result to ATL_1^*: if a subformula $\Psi = \langle\langle\emptyset\rangle\rangle\theta$ is true at some state of the path π in the concurrent game satisfying Φ, its effect cannot be constrained only on the centipede model grafted at the respective state of $\widetilde{\mathcal{M}}$, as done above, but it propagates through the path $\widetilde{\pi}$ to all centipede models grafted at all further states on $\widetilde{\pi}$. So, additional description in LTL is needed to describe and preserve this effect when converting π into the lasso $\widetilde{\pi}$. That is why we state the next result relativized to only what we have proved here.

Proposition 7.22 *The path-satisfiability problem for* ATL_1^* *formulas where only LoC models of exponentially bounded size are considered is in* PSPACE.

PROOF. The algorithm begins like the PSPACE decision procedure for LTL satisfiability that guesses the lasso on the fly for an LTL input formula θ [SC85]. First, the length of the prefix and the length of the period are guessed. At each step around the lasso, the subformulas that are true from the current state are guessed non-deterministically and a local consistency check as well as a one-step consistency check are performed. Further, a set Δ (of at most polynomial size) of eventuality formulas is kept to make sure that all eventualities that are needed for θ to be true are actually true further on the lasso.

The algorithm for ATL_1^* works in the same way on an ATL_1^* formula Φ, but treats strategically quantified subformulas of Φ as atomic propositions and, at each step of the procedure, the local consistency check includes verifying these subformulas that have to be true at the current state. This amounts to checking satisfiability of an $\mathrm{St}(\mathrm{ATL}_1^*)$ formula and can be done in PSPACE, by Theorem 7.19. For the formulas of the form $\langle\langle A\rangle\rangle\theta$ where $A \neq \emptyset$ this can be done independently of the rest of the lasso, by ensuring that when agents in A commit to satisfying θ then the play goes into the centipede (and stays there). But, when $A = \emptyset$ then θ has to be true on all paths from the current state. This includes both the path around the lasso and those that enter one of the centipedes at some point. Note that the original set Δ of formulas we are keeping only needs to be satisfied around the lasso. To keep track of this we keep, in addition to Δ, an extra set of formulas Γ which must be satisfied both around the lasso and on paths that exits to a centipede. Thus, the formulas in Γ must be included in the $\mathrm{St}(\mathrm{ATL}_1^*)$ satisfiability check at each step. But since Γ is polynomial in size at each step, this check can still be performed in PSPACE. This means that the entire procedure can be performed in PSPACE. □

$\mathcal{L}$	SAT($\mathcal{L}$)	SAT($\mathcal{L}_1$)
LTL	PSPACE [SC85]	NP [DS02]
CTL	EXPTIME [Eme90]	NP (Cor. 7.15)
CTL$^+$	2EXPTIME [JL03]	NP (Prop. 7.14)
CTL*	2EXPTIME [Eme90]	PSPACE (Theo. 7.19)
ATL	EXPTIME [Dri03]	Σ_3^P (Theo. 7.19)
ATL$^+$	2EXPTIME [Sch08][JL03]	Σ_3^P (Theo. 7.19)
ATL*	2EXPTIME [Sch08]	PSPACE (Theo. 7.19, Cor. 7.23)

Figure 7.4: Complexity of satisfiability. All results are completeness results. In the case of ATL* the results refer to St(ATL$_1^*$) and $\widehat{\text{ATL}_1^*}$.

Corollary 7.23 *Satisfiability of* $\widehat{\text{ATL}_1^*}$ *is* PSPACE*-complete.*

7.6 Summary

We have developed optimal decision procedures for the satisfiability problems in flat fragments of ATL*, and in particular CTL* and have obtained exact complexity results for all of them. A summary of the main complexity results obtained in this chapter is provided in the table in Fig. 7.4. It shows that these complexities are much lower than those for the full languages while, in view of Theorem 7.2, they are very tight with respect to syntactic extensions in terms of nesting depth of formulas.

Part II

Games with partial observation

CHAPTER 8

Introduction and background

In Part I we have seen how a number of verification and synthesis problems for computing systems can be modelled in a meaningful way using games with full observation. This includes, in particular, the modelling of open systems.

In many cases the assumption of full observation is too strict to model systems realistically, however. This includes most distributed systems where it is an inherent property that different processes have local state which is not immediately available to the other processes. This also includes a number of open systems where a system is interacting with the environment and only receives partial information about the state of the world, for instance by having imprecise sensors. To expand our scope to such applications we therefore extend our models to incorporate *partial observation*.

These new applications also lead us to solution concepts that are different from the antagonistic games considered in Part I. There, the goal of a player was always the opposite of the opponent. Or, in some cases, the goal of a coalition of players was the opposite of the goal of the remaining players.

In many multi-agent systems it makes more sense to assume that every player has his own objectives that he tries to fulfill. This view resembles a more classical game-theoretical and economical approach [OR94]. This assumption naturally leads to *non-zero sum objectives*. In particular, in Chapter 9 we study the well-known concept of *Nash equilibria* [Nas50; OR94] which is a strategy profile (containing one strategy for each player) such that no player can deviate and improve his own payoff, provided that all the other players stick to the strategy profile.

In Part II we will introduce two new formalisms for modelling systems with partial observation. In both cases these models are games in which each player has his own local module with the local states of the player. Further, each player only partially observes the local states of the other players. The players do have complete information about the rules of the game and the structure of each of the local modules of the other players, but during the play of the game they only partially observe what is going on.

The first model, introduced in Chapter 9, is a special class of concurrent game with partial observation which is built from local modules. With the goal of modelling systems with many similar processes we further work with symmetry constraints on the structure of the game. Indeed, our main focus is the search for a single

strategy which, when followed by all players, constitute a Nash equilibrium. Such Nash equilibria where all players follow the same strategy are called *symmetric Nash equilibria*.

The second model, introduced in Chapter 10, takes a different direction. In this model we incorporate continuous time as well as probabilistic transition functions as used in the paradigm of continuous-time Markov chains and continuous-time Markov decision processes. In addition, we incorporate explicit handshake-communication on shared actions as known from CSP [BHR84]. Indeed, every local module is an *interactive Markov chain* [Her02] and the game is a result of a composition of such modules. As these models are quite complex we tackle simpler objectives than in previous chapters and focus on reachability problems.

The more expressive models of Part II result in a higher computational complexity of the problems considered. Indeed, there are many undecidability results in games with partial information (see e.g. [PR90; FS05; BK10]). The title of a classical paper on distributed synthesis by Pnueli and Rosner says it quite well: "Distributed Reactive Systems Are Hard to Synthesize" [PR90].

Remark 8.1 *What we call* partial observation games *in this thesis is also known by a number of other names in the litterature. For instance, they are called incomplete information games (see e.g. [AHK02; Rei84]) and imperfect information games (see e.g. [OR94]).*

To be precise, in the partial observation games we consider all players have complete information about the rules of the game and the preferences of the other players before the game begins. What they do not have is full observation *of the state of the game during the play.*

CHAPTER 9

Nash Equilibria in Symmetric Games with Partial Observation

This chapter is based on the paper

- [BMV14] Patricia Bouyer, Nicolas Markey, and Steen Vester. "Nash Equilibria in Symmetric Games with Partial Observation". In: *Proceedings 2nd International Workshop on Strategic Reasoning (SR)*. Electronic Proceedings in Theoretical Computer Science. Grenoble, France, 2014, pages 49–55

and, in particular, the longer journal version

- [BMV16] Patricia Bouyer, Nicolas Markey, and Steen Vester. "Nash Equilibria in Symmetric Graph Games with Partial Observation". To appear in a special issue of *Information & Computation*. 2016

The journal version has been adapted to a form compatible with the other chapters of the thesis. This includes removing definitions that are already present in the thesis, moving some examples and adding a few new ones, reorganization and renaming of sections and subsections and the addition of more in-depth explanation of some of the content. All results in this chapter are also present in [BMV16].

Some of the contents and ideas of this chapter were developped during a Master project at LSV, ENS Cachan, France in 2012 with Patricia Bouyer and Nicolas Markey as supervisors. This work was published in the technical report

- [Ves12] Steen Vester. *Symmetric Nash equilibria*. Research Report LSV-12-23. Laboratoire Spécification et Vérification, ENS Cachan, France, December 2012

This report introduced a model containing many of the ingredients of the symmetric game networks of this chapter, but in a less developed form. In particular, the proofs of Proposition 9.16 and Theorem 9.19 were originally done for that setting and could simply be adapted to the setting of this chapter. Also, some of the ideas behind the algorithms of Theorems 9.21 and 9.25 were presented in that report.

9.1 Introduction

In games for formal verification there has until recently been most focus on the study of purely antagonistic games (a.k.a. zero-sum games) which conveniently represent systems evolving in a hostile environment: the aim of one player is to prevent the other player from achieving his own objective. This was also the case in the settings studied in Part I.

Games with *non-zero sum objectives* are interesting as they allow for conveniently modelling complex infrastructures where each individual system tries to fulfill its own objectives, while being subject to uncontrollable actions of the surrounding systems.

As an example, consider a wireless network in which several devices try to send data: each device can modulate its transmitting power, in order to maximize its bandwidth or reduce energy consumption as much as possible. In that setting, focusing only on optimal strategies for one single device is too narrow.

For another example, consider a server granting access to a printer and connected to several clients. The clients may send requests to the server, and the server grants access to the printer depending on the requests it receives. Here, the goal of each client is to get its own printing jobs processed as quickly as possible while the goal of the server might be to provide as fair a division of the printing resource as possible.

Game-theoreticians have defined and studied many solution concepts for non-zero sum settings of which *Nash equilibrium* [Nas50] is a prominent one. A Nash equilibrium is a strategy profile where no player can improve the outcome of the game by unilaterally changing his strategy.

Focus in this chapter is on handling the special case where many of the interacting systems have identical abilities and objectives. This encompasses many situations involving computerized systems over a network. We provide a convenient way of modelling such situations, and develop algorithms for synthesizing a single strategy that, when followed by all the players, leads to a global Nash equilibrium. To be meaningful, this requires symmetry assumptions on the structure of the game. We also include *partial observation* of the players which is relevant in such a setting.

Recent works have considered multi-player non-zero-sum games, including computation of constrained equilibria in turn-based and in concurrent games [CMJ04; UW11; Bou+15] and the development of temporal logics geared towards non-zero-sum objectives [CHP10; Mog+14; LM15]. None of those works distinguish symmetry constraints in strategy profiles nor in game description.

Still, symmetry has been studied in the context of normal-form games [Nas51; DM86]: in such games, each player has the same set of actions, and the utility function of a player only depends on his own action and on the number of players who played each action (it is independent on 'who played what').

Contribution

We propose a model for representing large interacting systems, which we call a *game network*. A game network is made of multiple copies of a one-player concurrent game;

each player plays on his own copy. As mentioned earlier, the players only partially observes the global state of the game. For instance, they may have a perfect view on some of their "neighbours", but may be blind to some other players.

In *symmetric game networks*, we additionally require that any two players are in similar situations: for every pair of players (a, b), we are able to map each player c to a corresponding player d with the informal meaning that 'player d is to b what player c is to a'. Of course, winning conditions and partial observation should respect that symmetry.

We present several examples illustrating the model, and argue why it is a relevant model. In these systems, we are interested in symmetric pure Nash equilibria, which are special Nash equilibria where all players follow the same deterministic strategy.

We show several undecidability results, in particular that the *parameterized synthesis problem* (aiming to obtain one policy that forms a Nash equilibrium when applied to any number of participants) is undecidable even for memoryless strategies. We then characterize the complexity of computing constrained pure symmetric Nash equilibria in symmetric game networks, when objectives are given as LTL formulas, and when restricting to memoryless and bounded-memory strategies. This problem with no memory bound is then proven undecidable.

Outline

In Section 9.2 we introduce Nash equilibria and symmetric game networks. In Section 9.3 we introduce the problems considered and provide reductions among some of these problems. Several different existence problems in symmetric game networks are analyzed in Section 9.4 and in Section 9.5 we consider similar problems in succinct symmetric game networks as well as the parameterized synthesis problem. In Section 9.6 we summarize the contents of the chapter.

9.2 Nash equilibria and symmetric game networks

In this section we introduce the solution concept of Nash equilibria. Further, we motivate and introduce symmetric game networks and analyze some of their structural properties.

9.2.1 Nash equilibria

Let $\mathcal{G} = (S, \Pi, \Sigma, \Gamma, \delta)$ be a concurrent game. Recall that a winning condition for player j is a set Ω_j of plays of $\mathcal{G}$. We say that a play $\rho \in \Omega_j$ yields *payoff* 1 to player j, and a play $\rho \notin \Omega_j$ yields payoff 0 to player j. Winning conditions are usually infinite, but will typically be given symbolically as the set of plays satisfying a given property. For instance, given a formula ϕ of some logic (like LTL) using the set S of states in $\mathcal{G}$ as atomic propositions, we write $\Omega(\phi)$ for the set of plays satisfying ϕ.

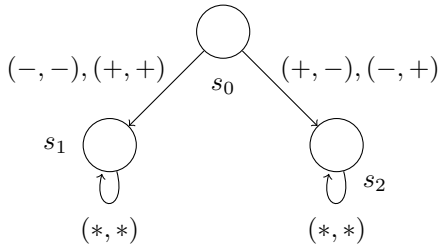


Figure 9.1: Matching pennies game modelled as a concurrent game.

If $\sigma = (\sigma_j)_{j \in A}$ is a strategy for coalition A and σ'_a is a strategy for player $a \in A$ then we write $\sigma[a \mapsto \sigma'_a]$ to denote the strategy for coalition A where a plays according to σ'_a and player j plays according to σ_j for all $j \in A \setminus \{a\}$.

A strategy σ of a coalition C is *winning for player j* from state s if $\mathrm{Play}(\mathcal{G}, s, \sigma) \subseteq \Omega_j$. A strategy profile (that is, a strategy for each player in the game) σ is a *Nash equilibrium* from state s if, for every $j \in \Pi$ and every strategy σ'_j, if σ is losing for player j from s, then so is $\sigma[j \mapsto \sigma'_j]$. In other terms, no player can individually improve his payoff.

In more general settings where the payoff can take more values than just 0 and 1, a Nash equilibrium is a strategy profile such that no player can improve his payoff by changing his strategy if the other players stick to the strategies prescribed in the strategy profile. This can also be generalized to randomized strategies and games with built-in randomness by requiring that no player can improve his expected payoff.

Remark 9.1 *In this chapter, we restrict to* pure Nash equilibria*. That is, Nash equilibria where players cannot choose actions according to a probability distribution. This corresponds to deterministic programs for the players. When we write Nash equilibria for the rest of the paper we mean pure Nash equilibria.*

Considering randomized strategies would clearly be of interest in presence of symmetry and would be a natural extension of this work. However, some restrictions will be required since the existence of a randomized Nash equilibrium in concurrent games with reachability objectives is undecidable already with three players [BMS14].

In Figure 9.1 the *matching pennies game* is modelled as a concurrent game where two players 0 and 1 independently and concurrently choose between actions $+$ and $-$. Player 0 wants both to choose the same action whereas player 1 wants both to choose different actions.

In this game there is no (pure) Nash equilibrium as no matter which player wins in a given pure strategy profile, the losing player can change his strategy to win. If the players were allowed to use randomization, they could both player $+$ and $-$ with probability $\frac{1}{2}$. Then no player would be able to improve their probability of winning given that the other player sticks to this strategy.

For another example, consider a similar game, but with three players. All players can choose either $+$ or $-$ as in the matching pennies game. In this game a player wins if and only if there is exactly one other player that chooses the same action. Let (x, y, z) denote a strategy profile where player 1 chooses x, player 2 chooses y and player 3 chooses z.

First, observe that all players cannot win in this game. Next, observe that the strategy profiles given by the tuples

$$(+,+,-),(-,-,+),(+,-,+),(-,+,-),(-,+,+),(+,-,-)$$

are all Nash equilibria in which two of the players win and no player can improve his own payoff by deviating. Finally, there are two less desirable Nash equilibria in which all players lose, namely $(+,+,+)$ and $(-,-,-)$. Here, no player can improve by deviating given that the other players stick to their strategy.

Even though Nash equilibria are some of the most well-studied solution concepts both in normal-form games and extensive-form games [OR94], there are situations where they are not sufficient to prescribe rational behavior, for instance, in situations with non-credible threats. For a discussion, see e.g. [OR94].

To handle phenomena like this other kinds of equilibria, such as subgame-perfect equilibria [Osb04], have been defined. Studying such equilibrium concepts in our setting would also be a natural continuation of our work. Though, despite the issues, Nash equilibria are still sufficient in many cases. As a practical example, most of the succesful programs in the AAAI Computer Poker Competition are based on Nash equilibrium computation [San10].

9.2.2 Game networks

Our aim is to propose a convenient way of modelling situations where all the interacting systems have identical abilities and objectives, and to develop algorithms for synthesizing symmetric strategy profiles in that setting. Intuitively, a symmetric strategy profile is a strategy profile where the same single strategy is played by all the players.

The model we propose is made of a one-player concurrent game (called an arena), together with an observation relation. Intuitively, each player plays in his own copy of the one-player arena; the global system is the (synchronous) product of all the local copies, but each player observes the state of the global system only through an observation relation. This way of defining partial observation is just like in concurrent games with partial observation. It is in particular needed for representing large networks of systems, in which each player may only observe some of his neighbours.

Example 9.2 *Consider a set of identical devices (e.g. cell phones) connected on a local area network. Each device can modulate its emitting power. In order to increase its bandwidth, a device tends to increase its emitting power; but besides consuming more energy, this also adds* noise *over the network, which decreases the other players' bandwidth and encourages them to in turn increase their power.*

We can model a device as an n-state arena where state i corresponds to some power p_i, with $p_0 = 0$ representing the device being off. A device would not know the exact state of the other devices, but would be able to evaluate the surrounding noise; this can be modelled using our observation relation, where all configurations with the same level of noise would be equivalent. Based on this information, the device can decide whether it should increase or decrease its emitting power, resulting in a good balance between bandwidth and energy consumption.

Despite the global game being described as a product of identical arenas, not all games described this way are symmetric: the observation relation should also be symmetric, and we have to impose extra conditions on that relation in order to capture an adequate notion of symmetry. Moreover, the observation relation relates global states of the system, and an explicit description of it will most often not be practical. We thus consider compact representations of this relation, as we now explain.

We first define our notion of an n-player *game network*, which describes a game as a product of n identical one-player games.

For every $k \in \mathbb{N} \cup \{\infty\}$, we write $[k]$ for the set $\{i \in \mathbb{N} \mid 0 \leq i < k\}$ and $[\infty] = \mathbb{N}$.

Definition 9.3 *An* n-player game network $\mathcal{G}$ *is a tuple* $(G, (\equiv_i)_{i\in[n]}, (\Omega_i)_{i\in[n]})$ *such that*

- $G = (S, \{a\}, \Sigma, \Gamma, \delta)$ *is a* one-player concurrent game (arena)*;*
- *for each* $i \in [n]$, $\equiv_i$ *is an* equivalence relation *on* S^n *extended in a natural way to sequences of states of* S^n*. Two* $\equiv_i$*-equivalent elements of* S^n *are indistinguishable to player* i*. This models* partial observation *for player* i*. If* $\equiv_i$ *is the identity, then we say player* i *has full observation;*
- *for each* $i \in [n]$, $\Omega_i \subseteq (S^n)^\omega$ *is the* objective *of player* i*. We require that for all* $\rho, \rho' \in (S^n)^\omega$*, if* $\rho \equiv_i \rho'$ *then* $\rho \in \Omega_i$ *if and only if* $\rho' \Omega_i$*.*

The semantics of a game network $\mathcal{G} = (G, (\equiv_i)_{i\in[n]}, (\Omega_i)_{i\in[n]})$ is defined in terms of the concurrent game $\mathcal{G}' = (S', [n], \Sigma, \Gamma', \delta', (\equiv_i)_{i\in[n]})$ with partial observation (see Definition 2.7) where

- $S' = S^n$ is the set of states
- $\Gamma'((s_0, \ldots, s_{n-1}), i) = \Gamma(s_i, i)$ for $i \in [n]$ specifies the actions available
- and the transition function is defined as

$$\delta'((s_0, \ldots, s_{n-1}), (m_i)_{i\in[n]}) = (\delta(s_0, m_0), \ldots, \delta(s_{n-1}, m_{n-1})).$$

for all $(s_0, ..., s_{n-1}) \in S'$ and legal action tuples $(m_i)_{i\in[n]}$ in $(s_0, ..., s_{n-1})$

An element of S^n is called a *configuration* of $\mathcal{G}$. Notice that we do not fix an initial state for the one-player arena as we want to be able to model cases where the players start in different states. For example, this makes us capable of modelling settings where not all players start playing at the same time.

Example 9.4 *Consider the cell-phone game again. It can be modelled as a game network where each player observes everything. That is, the equivalence relations $\equiv_i$ are the identity. A more realistic model for the system can be obtained by assuming that each player only gets precise information about his close neighbours, and less precise information, or no information at all, about the devices that are far away.*

9.2.3 Symmetric game networks

If we impose no restriction on the observation relation, n-player game networks do not fully capture symmetries in a system. Besides playing on similar arenas, we will add the extra requirement that all the players are in similar situations w.r.t. the other players.

Given a mapping $f\colon A \to B$, $a \in A$, and $b \in B$, we write $f[a \mapsto b]$ for the mapping g such that $g(a) = b$ and $g(\alpha) = f(\alpha)$ for all $\alpha \in A \setminus \{a\}$. Given a mapping $f\colon A \to B$ and a mapping $g\colon B \to C$, the composition of g and f is the mapping $g \circ f\colon A \to C$ defined as $g \circ f(a) = g(f(a))$. Seeing a sequence $\rho = (\rho_i)_{i\in[n]}$ as a mapping with domain $[n]$, and given a mapping $f\colon [m] \to [n]$, we write $\rho \circ f$ for the sequence $(\rho_{f(j)})_{j\in[m]}$.

Given a permutation π of $[n]$, for a configuration $t = (s_i)_{i\in[n]}$ we define $t(\pi) = (s_{\pi(i)})_{i\in[n]}$; similarly, for a path $\rho = (t_j)_{j\in\mathbb{N}}$, we define $\rho(\pi) = (t_j(\pi))_{j\in\mathbb{N}}$.

We now refine the previous definition for a game network to capture symmetries in the system. To do so we introduce permutations $\pi_{i,j}$ for players i and j to define the symmetry of the game. The idea is that $\pi_{i,j}(k) = l$ means that player l plays vis-à-vis player j the role that player k plays vis-à-vis player i. To be symmetric the game has be the same from the point of view of every player. Thus, the permutation $\pi_{i,j}$ can be used to map every player from the point of view of player i to every player from the point of view of player j. This intuition is reflected in the following definition.

Definition 9.5 *A game network $\mathcal{G} = (G, (\equiv_i)_{i\in[n]}, (\Omega_i)_{i\in[n]})$ is symmetric whenever there are permutations $\pi_{i,j}$ of $[n]$ for every two players $i, j \in [n]$ such that $\pi_{i,j}(i) = j$ and satisfying the following conditions: for every $i, j, k \in [n]$,*

1. *$\pi_{i,i}$ is the identity, and $\pi_{k,j} \circ \pi_{i,k} = \pi_{i,j}$; hence $\pi_{i,j}^{-1} = \pi_{j,i}$.*
2. *the observation made by the players is compatible with the symmetry of the game: for all configurations t and t', $t \equiv_i t'$ if and only if $t(\pi_{i,j}^{-1}) \equiv_j t'(\pi_{i,j}^{-1})$;*
3. *objectives are compatible with the symmetry of the game: for every play ρ, $\rho \in \Omega_i$ if and only if $\rho(\pi_{i,j}^{-1}) \in \Omega_j$.*

In that case, $\pi = (\pi_{i,j})_{i,j\in[n]}$ is called a symmetric representation *of $\mathcal{G}$.*

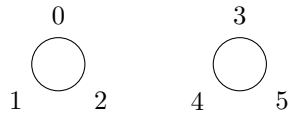


Figure 9.2: A card-game tournament.

We give the intuition why we apply $\pi_{i,j}^{-1}$ in the definition above, and not $\pi_{i,j}$. Assume configuration $t = (s_0, \ldots, s_{n-1})$ is observed by player i. The corresponding configuration for player j is $t' = (s'_0, \ldots, s'_{n-1})$ where player-$\pi_{i,j}(k)$ state should be that of player k in t. That is, $s'_{\pi_{i,j}(k)} = s_k$, so that $t' = t(\pi_{i,j}^{-1})$. For a discussion of the subtleties of the definition of symmetry in the context of normal-form games, see [Ves12].

These mappings define how symmetry must be used in strategies: let $\mathcal{G}$ be a symmetric n-player game network with symmetric representation π. We say that a strategy profile $\sigma = (\sigma_i)_{i\in[n]}$ is *symmetric* for the representation π if it is realizable (i.e., each player only plays according to what he can observe) and if for all $i, j \in [n]$ and every history ρ, it holds $\sigma_i(\rho) = \sigma_j(\rho(\pi_{i,j}^{-1}))$.

Symmetric Nash equilibria are the special kinds of Nash equilibria which are also symmetric strategy profiles. This means that a symmetric Nash equilibrium is a Nash equilibrium where all players apply the same strategy.

Example 9.6 *Consider a card game tournament with six players, three on each table (see Figure 9.2). Here each player has a left neighbour, a right neighbour, and three opponents at a different table. To model this, one could assume player* 0 *knows everything about himself, and has some information about his right neighbour (player* 1*) and his left neighbour (player* 2*). But he knows nothing about players* 3*,* 4 *and* 5*.*

Now, the role of player 2 *vis-à-vis player* 1 *is that of player* 1 *vis-à-vis player* 0 *(he is his right neighbour). Hence, we can define the symmetry as* $\pi_{0,1}(0) = 1$*,* $\pi_{0,1}(1) = 2$*,* $\pi_{0,1}(2) = 0$*, and* $\pi_{0,1}(3, 4, 5) = (3, 4, 5)$ *(any choice is fine here). As an example, the observation relation in this setting could be that player* 0 *has perfect knowledge of his set of cards, but only knows the number of cards of players* 1 *and* 2*, and has no information about the other three players. Notice that other observation relations would have been possible, for instance, giving more information about the right player.*

Example 9.7 *Consider again the cell-phone example. In this model, the noise depends on the relative positions of the devices, and in that sense this game is not symmetric. The model of the cell-phone could include information about the relative positions of the other devices, by including several disjoints copies of the model, in which the neighbour devices have different influences over the noise. The initial state for each player would then depend on the topology of the network.*

Example 9.8 *Finally, let us mention that even though it is not fully symmetric, it is possible to model a client-server architecture in our framework. Let S be a model for the server and C be a model for the client. The arena G will then be the disjoint union of S and C, and the equivalence $\equiv_i$ is defined such that "if player i is in part S, then he has full observation of all players, and if player i is in part C, then he sees his own states and the state of players in part S", having in mind that in the initial configuration it is ensured the one player starts in part S (being the server player) and all other players will be clients starting in part C.*

9.2.4 Properties of symmetric representations

First note that symmetric representations are not unique in general. We discuss here the impact of the symmetric representation on Nash equilibria.

We first define for each player a partitioning of the set of all the players, which will define what the players cannot distinguish. We then prove various indistinguishability properties, give examples and give conditions over the representations which ensure identical Nash equilibria.

For every player $i \in [n]$, we let $\cong_i$ be the following equivalence relations on the set of players $[n]$: $j \cong_i k$ if and only if for every configuration t, $t \equiv_i t(\pi_{j\leftrightarrow k})$, where $\pi_{j\leftrightarrow k}$ is the permutation such that $\pi_{j\leftrightarrow k}(j) = k, \pi_{j\leftrightarrow k}(k) = j$ and $\pi_{j\leftrightarrow k}(i) = i$ for all $i \notin \{j, k\}$. It means that player i cannot distinguish between the players j and k. We then define for every $i \in [n]$, the partition $\mathcal{P}_i$ of $[n]$ which is induced by $\cong_i$. We denote $\mathcal{P} = (\mathcal{P}_i)_{i\in[n]}$.

Lemma 9.9 *For every symmetric representation π of $\mathcal{G}$, for every $i, j \in [n]$, for every $P \in \mathcal{P}_i$, it holds $\pi_{i,j}(P) \in \mathcal{P}_j$.*

Proof. Assume for contradiction that it is not the case. As $\mathcal{P}_j$ is a partition there are two possibilities. Either $\pi_{i,j}(P)$ is a proper subset of some $Q \in \mathcal{P}_j$. Otherwise, there are two different sets $Q, Q' \in \mathcal{P}_j$ such that there are elements of $\pi_{i,j}(P)$ in both of them. The two cases are handled separately below.

- First, suppose $\pi_{i,j}(P) \subsetneq Q$ for some $Q \in \mathcal{P}_j$. Take $k_j \in Q \setminus \pi_{i,j}(P)$ and $p_j \in \pi_{i,j}(P)$. For every configuration t, we have that $t \equiv_j t(\pi_{k_j\leftrightarrow p_j})$. We define $p_i = \pi_{i,j}^{-1}(p_j)$ (which is then in P) and $k_i = \pi_{i,j}^{-1}(k_j)$ (which is then not in P). As π is a symmetric representation of $\mathcal{G}$, we have that $t(\pi_{i,j}^{-1}) \equiv_i t(\pi_{i,j}^{-1} \circ \pi_{k_j\leftrightarrow p_j})$. We can now notice that $t(\pi_{i,j}^{-1} \circ \pi_{k_j\leftrightarrow p_j}) = t(\pi_{k_i\leftrightarrow p_i} \circ \pi_{i,j}^{-1})$, which then implies $t(\pi_{i,j}^{-1}) \equiv_i t(\pi_{k_i\leftrightarrow p_i} \circ \pi_{i,j}^{-1})$. For every t', we therefore get $t' \equiv_i t'(\pi_{k_i\leftrightarrow p_i})$. This contradicts the fact that $p_i \in P$ and $k_i \notin P$. This case is not possible.
- Second, suppose there exists two sets $Q, Q' \in \mathcal{P}_j$ such that $Q \neq Q'$, and $\pi_{i,j}(P) \cap Q \neq \emptyset$ and $\pi_{i,j}(P) \cap Q' \neq \emptyset$. The reasoning is similar to above.

□

Example 9.10 *We now consider an example where players cannot observe which players are in what states, but where players can see how many players are in a certain state. Such an assumption is realistic for example in a setting such as in the cell-phone example. Here it might be possible to tell how many other phones are turned on by measuring the surrounding noise. However, it is not possible to decide which phones are turned on.*

For a configuration $t = (s_i)_{i\in[n]}$ *and a subset* P *of players, we define* t_P *as the subsequence* $(s_i)_{i\in P}$*, and its* Parikh image $\mathrm{Parikh}(t_P)$ *as the function mapping each state* s *to its number of occurrences in* t_P*. Now, we define the observation relation* $\mathrm{Parikh}(P)$ *as follows:*

$$(t,t') \in \mathrm{Parikh}(P) \quad \textit{if and only if} \quad \mathrm{Parikh}(t_P) = \mathrm{Parikh}(t'_P).$$

Similarly, we define the observation relation $\mathrm{Id}(P)$ *as*

$$(t,t') \in \mathrm{Id}(P) \quad \textit{if and only if} \quad t[i] = t'[i] \textit{ for all } i \in P.$$

Using these relations and the fact that the intersection of two equivalence relations is an equivalence relation, we can define various observation relations, for instance the relation $\equiv_i$ *defined by*

$$\mathrm{Id}(\{i\}) \cap \mathrm{Parikh}(\{i+1, i+2, i+3\}) \cap \mathrm{Parikh}(\{i+3, i+4, i+5\})$$

where all indices are taken modulo n*. With such an observation, player* i *has perfect information about his own state, and knows the Parikh images for players* $i+1$*,* $i+2$ *and* $i+3$ *and for players* $i+3$*,* $i+4$ *and* $i+5$*. One can check that the partition* $\mathcal{P}_i$ *is then* $(\{i\}, \{i+1, i+2\}, \{i+3\}, \{i+4, i+5\})$ *where player* $i+3$ *plays a special role as he appears in the two Parikh conditions.*

There are two reasons why a symmetric game network may admit several symmetric representations. For instance, in a three-player game where each player only observes the Parikh image of the other two players, mappings π can either be defined as $\pi_{0,1}(1) = 2$ and $\pi_{0,1}(2) = 0$, or $\pi_{0,1}(1) = 0$ and $\pi_{0,1}(2) = 2$. Such distinctions are harmless in general, and those will generate the same symmetric behaviours. More precisely:

Lemma 9.11 *Let* $\mathcal{G}$ *be a symmetric* n*-player game network. Take two symmetric representations* π *and* $\widetilde{\pi}$ *for* $\mathcal{G}$*. Assume that for every* $i \in [n]$*, for every part* $P \in \mathcal{P}_i$*,* $\pi_{i,j}(P) = \widetilde{\pi}_{i,j}(P)$*. Then, a strategy profile* σ *is symmetric for* π *if and only if it is symmetric for* $\widetilde{\pi}$*.*

Proof. It is sufficient to show that for every configuration t, $t(\pi_{i,j}^{-1}) \equiv_j t(\widetilde{\pi}_{i,j}^{-1})$.

Let π be a permutation of $[n]$ that preserves partition $\mathcal{P}_i$ such that $\widetilde{\pi}_{i,j} = \pi_{i,j} \circ \pi$. Let t be a configuration. As π preserves $\mathcal{P}_i$, $t \equiv_i t(\pi^{-1})$. This implies, if we apply the symmetry condition for $\pi_{i,j}$: $t(\pi_{i,j}^{-1}) \equiv_j t(\pi^{-1} \circ \pi_{i,j}^{-1})$, that is, $t(\pi_{i,j}^{-1}) \equiv_j t(\widetilde{\pi}_{i,j}^{-1})$.

Therefore the symmetry condition for the strategy profile does not depend on the choice of the symmetry mappings. □

Symmetric representations might however differ more 'dramatically'. Assume for instance that $n = 6$, and that $\equiv_i$ is defined for every $i \in [6]$ as 'Id($\{i\}$) $\cap$ Parikh($\{i+1, i+2\}$) $\cap$ Parikh($\{i+3, i+4\}$)' (taken modulo 6). Then the partition $\mathcal{P}_i$ is equal to $(\{i\}, \{i+1, i+2\}, \{i+3, i+4\}, \{i+5\})$, and the mappings $\pi_{i,j}(i+k) = j+k \bmod 6$ properly define the symmetry. But there are other mappings that define the symmetry, for instance:

$$\pi'_{2i,2i+1}: \begin{array}{rl} 2i & \mapsto 2i+1 \\ 2i+1 & \mapsto 2i+4 \\ 2i+2 & \mapsto 2i+5 \\ 2i+3 & \mapsto 2i+2 \\ 2i+4 & \mapsto 2i+3 \\ 2i+5 & \mapsto 2i \end{array} \qquad \pi'_{2i+1,2i+2}: \begin{array}{rl} 2i & \mapsto 2i+1 \\ 2i+1 & \mapsto 2i+2 \\ 2i+2 & \mapsto 2i+5 \\ 2i+3 & \mapsto 2i \\ 2i+4 & \mapsto 2i+3 \\ 2i+5 & \mapsto 2i+4 \end{array}$$

The other mappings are obtained by composition. This also properly represents the symmetry, but generates different symmetric strategy profiles. Under additional technical conditions, we can prove that Nash equilibria coincide for two symmetric representations of a given symmetric game network. First we realize that a symmetric strategy profile is fully determined by an $\equiv_0$-realizable strategy for player 0 (recall that a strategy is realizable if it is compatible with the observation relation).

Lemma 9.12 *Fix a symmetric representation π for $\mathcal{G}$. If σ_0 is an $\equiv_0$-realizable strategy for player 0, then the strategy profile σ defined by $\sigma_i(\rho) = \sigma_0(\rho(\pi_{i,0}^{-1}))$ defines a realizable and symmetric strategy profile.*

Proof. Symmetry is straightforward:

$$\sigma_j(\rho(\pi_{i,j}^{-1})) = \sigma_0(\rho(\pi_{j,0}^{-1} \circ \pi_{i,j}^{-1})) = \sigma_0(\rho(\pi_{i,0}^{-1})) = \sigma_i(\rho).$$

Assume that σ_i is not $\equiv_i$-realizable: this means that there are two runs $\rho \equiv_i \rho'$ such that $\sigma_i(\rho) \neq \sigma_i(\rho')$. By the symmetry of the game, it holds that $\rho(\pi_{i,0}^{-1}) \equiv_0 \rho'(\pi_{i,0}^{-1})$, which implies that $\sigma_0(\rho(\pi_{i,0}^{-1})) = \sigma_0(\rho'(\pi_{i,0}^{-1}))$. However this precisely means $\sigma_i(\rho) = \sigma_i(\rho')$. Hence strategy σ_i is $\equiv_i$-realizable. □

We can now show the following result which says that under mild assumptions, the choice of symmetric representation does not affect which Nash equilibria are symmetric.

Lemma 9.13 *Assume that a symmetric representation π of game network $\mathcal{G} = (G, (\equiv_i)_{i\in[n]}, (\Omega_i)_{i\in[n]})$ satisfies the following additional constraint: if there exist permutations $(\kappa_i)_{i\in[n]}$ of $[n]$ such that:*

(i) $\kappa_i(i) = i$ *for every* i

(ii) for every two configurations t *and* t'*,*

$$t \equiv_i t' \Leftrightarrow t(\kappa_i \circ \pi_{j,i} \circ \kappa_j^{-1} \circ \pi_{i,j}) \equiv_i t'(\kappa_i \circ \pi_{j,i} \circ \kappa_j^{-1} \circ \pi_{i,j})$$

(iii) for every run ρ*, we have* $\rho \in \Omega_i$ *if and only if* $\rho(\kappa_i \circ \pi_{j,i} \circ \kappa_j^{-1} \circ \pi_{i,j}) \in \Omega_i$

then for every configuration t*,* $t \equiv_i t(\kappa_i \circ \pi_{j,i} \circ \kappa_j^{-1} \circ \pi_{i,j})$*. Under that additional constraint, the choice of the representations does not affect Nash equilibria. More precisely: if* π *and* $\widetilde{\pi}$ *are two symmetric representations of game* $\mathcal{G}$ *that satisfy the above hypothesis, then a realizable strategy profile* σ *is a symmetric Nash equilibrium from* t *in* $\mathcal{G}$ *for representation* π *if and only if it is a symmetric Nash equilibrium from* t *in* $\mathcal{G}$ *for representation* $\widetilde{\pi}$*.*

Proof. Let $(\pi_{i,j})_{i,j}$ and $(\widetilde{\pi}_{i,j})_{i,j}$ be two different representations (for the pieces of the canonical partitioning). Following Def. 9.5, those mappings are uniquely characterized by $(\pi_{0,i})_i$ and $(\widetilde{\pi}_{0,i})_i$. Assume κ_i is the permutation of $[n]$ such that $\widetilde{\pi}_{0,i} = \kappa_i \circ \pi_{0,i}$ (in particular w.l.o.g. κ_i swaps pieces of $\mathcal{P}_i$). We notice that $\widetilde{\pi}_{i,j} = \kappa_j \circ \pi_{i,j} \circ \kappa_i^{-1}$, again applying the properties listed in Def. 9.5.

It is not difficult to prove all the conditions for the κ_i's:

- using $\widetilde{\pi}_{0,i} = \kappa_i \circ \pi_{0,i}$, we get $\widetilde{\pi}_{0,i}(0) = \kappa_i \circ \pi_{0,i}(0)$, which implies $\kappa_i(i) = i$.
- it holds $t \equiv_i t'$ if and only if $t(\widetilde{\pi}_{i,j}^{-1}) \equiv_j t'(\widetilde{\pi}_{i,j}^{-1})$, which in turn is equivalent to $t(\widetilde{\pi}_{i,j}^{-1} \circ \pi_{j,i}^{-1}) \equiv_i t'(\widetilde{\pi}_{i,j}^{-1} \circ \pi_{j,i}^{-1})$. This directly implies the second property of the lemma.
- the third property is proven by applying the same argument: $\rho \in \Omega_i$ is equivalent to $\rho(\widetilde{\pi}_{i,j}^{-1} \circ \pi_{j,i}^{-1}) \in \Omega_i$, from which the property follows.

We therefore get that $t \equiv_i t(\kappa_i \circ \pi_{j,i} \circ \kappa_j^{-1} \circ \pi_{i,j})$ for every i and j, and in particular, taking $j = 0$, we get that $t \equiv_i t(\kappa_i)$ (since κ_0 is the identity).

Fix a strategy σ_0 for player 0 (which is $\equiv_0$-realizable). It defines two strategy profiles σ and $\widetilde{\sigma}$. For every i, we compute:

$$\widetilde{\sigma}_i(\rho) = \sigma_0(\rho(\widetilde{\pi}_{i,0}^{-1})) = \sigma_0(\rho(\kappa_i \circ \pi_{i,0}^{-1})) = \sigma_i(\rho(\kappa_i)) = \sigma_i(\rho) \quad (\text{since } \rho \equiv_i \rho(\kappa_i))$$

In particular, $\widetilde{\sigma}$ and σ yield the same payoff to all players. Now if one of the players can improve his payoff, say player i can use strategy σ_i' to get better payoff than with his strategy σ_i, then he can also improve his payoff by playing the same strategy σ_i' in place of strategy $\widetilde{\sigma}_i$. □

In the sequel, we always assume that the symmetric representation is given. Note however that a symmetric representation can be computed in space polynomial in the number of players, by just enumerating the permutations and checking that they satisfy the constraints. As we show later, the problems we consider have higher complexity (when decidable), so that this assumption does not alter our results.

Encoding of symmetric game networks. One motivation for the definition of this model is to represent large networks of identical systems in a rather compact way. To this aim, we need a succinct representation of game networks, in particular for the relations $\equiv_i$. Notice that representing those equivalence relations explicitly as $|S|^n \times |S|^n$ tables is not practical. We therefore allow equivalence relations to be given *symbolically*, for instance as a polynomial-time program (or Turing machine) taking two integers $i \leq n$ and two configurations t and t' in S^n, and returning 1 if and only if $t \equiv_i t'$. Examples of such functions are Parikh(P) and Id(P) defined previously.

9.3 Problems and reductions

We are interested in the computation of Nash equilibria and symmetric Nash equilibria in symmetric game networks. More precisely, we are interested in the following two problems:

Problem 9.14 (Existence of a symmetric NE) *Given a symmetric game network $\mathcal{G}$, a symmetric representation π, and a configuration t, the* existence problem *asks whether there is a symmetric Nash equilibrium in $\mathcal{G}$ from t for the representation π.*

Note that there might not exist a pure Nash equilibrium in a symmetric game network. Figure 9.3 shows how one can simulate the *matching penny game*, which is known not to have pure Nash equilibria. We assume there are two players, and they both have full observation. Player 0 starts from p_0 whereas player 1 starts from q_0. The objective is the same for player 0 and for player 1 and is written:

$$\Big(p_0 \Rightarrow (\mathbf{F}((p_+ \wedge q_+) \vee (p_- \wedge q_-)))\Big) \wedge \Big(q_0 \Rightarrow (\mathbf{F}((p_+ \wedge q_-) \vee (p_- \wedge q_+)))\Big).$$

This reads as follows: "if you are in p_0, then you have to eventually visit both p_+ and q_+, or both p_- and q_-, and if you are in q_0, you have to eventually visit both p_+ and q_-, or both p_- and q_+". It is not hard to be convinced that it is symmetric, and that there is no pure Nash equilibrium from (p_0, q_0) in that game network.

Figure 9.3: Matching pennies as a symmetric game network. Self-loops in p_+, p_-, q_+ and q_- are omitted.

Problem 9.15 (Constrained existence of a symmetric NE) *Given a symmetric game network $\mathcal{G}$, a symmetric representation π, a configuration t, a set $L \subseteq [n]$ of losing players, and a set $W \subseteq [n]$ of winning players, the* constrained existence

problem *asks whether there is a symmetric Nash equilibrium σ in $\mathcal{G}$ from t for the representation π, such that all players in L lose and all players in W win. If $W = [n]$, the problem is called the* positive existence problem.

Note that for the constrained problem, the input sets L and W do not need to cover the entire set of players. Thus, L and W constitute a partial specification of which players should win and lose.

9.3.1 From Nash equilibria to symmetric Nash equilibria

In this section we show that even though symmetric Nash equilibria are Nash equilibria satisfying special properties, they are in some sense at least as hard to find as Nash equilibria. This unfortunately means that we cannot in general hope to have an algorithm with better complexity for the symmetric problem by using properties of symmetry. Furthermore, it allows us to infer hardness results from the framework with standard Nash equilibria to the framework with symmetric Nash equilibria. The result is formalized in the following proposition which will be proved in this section.

Proposition 9.16 *From a symmetric game network $\mathcal{G}$, we can construct in polynomial time a symmetric game network $\mathcal{H}$ such that there exists a symmetric Nash equilibrium in $\mathcal{H}$ if and only if there exists a Nash equilibrium in $\mathcal{G}$. Furthermore the construction only changes the arena, but does not change the number of players nor the objectives or the resulting payoffs.*

The overall idea of the proof is to take the arena in $\mathcal{G}$ that each player plays in and combine n disjoint copies of this arena to obtain the arena of $\mathcal{H}$. Then we will show that deciding whether there exists a Nash equilibrium in $\mathcal{G}$ can be reduced to deciding whether there exists a symmetric Nash equilibrium in $\mathcal{H}$ where each player starts in different disjoint copies of the arena. This construction, in practice, makes it possible for players to play differently even in a symmetric strategy profile as they all start in different states.

The overall idea is thus quite simple, but the execution of it is a bit technical due to the nature of the symmetry constraints and the partial observation.

Let $\mathcal{G} = (G, (\equiv_i)_{i\in[n]}, (\Omega_i)_{i\in[n]})$ be a symmetric game network with symmetric representation $\pi = (\pi_{i,j})_{i,j\in[n]}$ and $(s_0, ..., s_{n-1})$ be an initial configuration.

We show how to build a symmetric game network $\mathcal{H}$ which has a symmetric Nash equilibrium from some particular configuration if and only if $\mathcal{G}$ has a Nash equilibrium from $(s_0, ..., s_{n-1})$. The construction is as follows.

First, let $G = (S, \{a\}, \Sigma, \Gamma, \delta)$. We define $\mathcal{H} = (H, (\sim_i)_{i\in[n]}, (\Theta_i)_{i\in[n]})$. Here, $H = (S_H, \{a\}, \Sigma, \Gamma_H, \delta_H)$ is defined by

- $S_H = S \times [n]$
- $\Gamma_H((s, j), a) = \Gamma(s, a)$ for all $(s, j) \in S_H$

- $\delta_H((s,j),a) = (\delta(s,a),j)$

We define $\sim_i$ such that for all $i \in [n]$, for all configurations $(u_0,...,u_{n-1}),(v_0,...,v_{n-1})$ in $\mathcal{G}$ and for all $m_0,...,m_{n-1},j_0,...,j_{n-1} \in [n]$, it holds that:

$$((u_0,m_0),...,(u_{n-1},m_{n-1})) \sim_i ((v_0,j_0),...,(v_{n-1},j_{n-1}))$$

if and only if

$$(u_0,...,u_{n-1}) \equiv_i (v_0,...,v_{n-1}) \wedge m_i = j_i.$$

That is, none of the players can observe which copy the other players are in, but can observe which copy they are in themselves.

The objectives are defined such that for every player i and every infinite play $\rho = (s_0^1,...,s_{n-1}^1)(s_0^2,...,s_{n-1}^2)...$ in $\mathcal{G}$ and all $j_0,...,j_{n-1} \in [n]$ we define

$$((s_0^1,j_0),...,(s_{n-1}^1,j_{n-1}))((s_0^2,j_0),...,(s_{n-1}^2,j_{n-1}))... \in \Theta_i \quad \Leftrightarrow \quad \rho \in \Omega_i$$

That is, the objectives do not distinguish between which copy a player is in.

Finally, we let the initial configuration of $\mathcal{H}$ be $((s_0,0),...,(s_{n-1},n-1))$. That is, all players starts in different copies of G.

The rest of the proof is done by showing that

1. $\sim_i$ is an equivalence relation for every $i \in [n]$ implying that $\mathcal{H}$ is a game network
2. π is also a symmetric representation of $\mathcal{H}$ implying that $\mathcal{H}$ is a symmetric game network
3. There exists a Nash equilibrium in $\mathcal{G}$ from $(s_0,...,s_{n-1})$ if and only if there exists a Nash equilibrium in $\mathcal{H}$ from $((s_0,0),...,(s_{n-1},n-1))$

The first two points can be shown directly from the construction of $\mathcal{H}$ and the definition of a symmetric representation. For the third point there are two directions which are quite similar.

For the first direction, the idea is to show that for a strategy profile σ in $\mathcal{G}$ and an initial configuration $(s_0,...,s_{n-1})$ one can construct a strategy profile σ' in $\mathcal{H}$ that plays similarly starting in configuration $((s_0,0),...,(s_{n-1},n-1))$.

Moreover, one can show that such a strategy profile σ' can be defined which is symmetric. This is possible since the players start in different states in $\mathcal{H}$ and therefore do not need to play in the same way in order to be symmetric (symmetry only requires players to play similarly when they are in similar situations).

Finally, it is shown that if σ is a Nash equilibrium then so is σ'. The idea is that for every deviation a player can make from σ' in $\mathcal{H}$ with initial configuration $((s_0,0),...,(s_{n-1},n-1))$ there is a similar deviation for the same player from σ in $\mathcal{G}$ with initial configuration $(s_0,...,s_{n-1})$. When σ is a Nash equilibrium this implies that σ' is also a Nash equilibrium since no player can gain anything by deviating from σ in $\mathcal{G}$.

For the other direction a similar proof is done, but a strategy profile in $\mathcal{G}$ is constructed from a strategy profile in $\mathcal{H}$.

The details of the proof are omitted here, but can be found in the journal paper [BMV16].

There are n times as many states in the arena H as in the arena G. In addition, there are n times as many equivalence classes, which implies that the size of $\mathcal{H}$ is polynomial in the size of $\mathcal{G}$. This concludes the proof of Proposition 9.16.

The trick used in the proof where one makes several copies of the arena and let the players start in different copies is very similar to the trick we used in Example 9.8. There each arena consists of two disconnected parts: a server part and a client part. Then, the server player starts in the server part and the client players start in the client part. Then, in a symmetric strategy profile only the clients are required to behave in the same way as only the clients face the same situations.

9.3.2 From positive existence to existence

Before turning to our decidability and undecidability results, we begin by showing that existence of positive Nash equilibria (i.e. Nash equilibria where all players win) is not harder than existence when the objectives of the players are given by LTL formulas. This is quite natural as all players have to win and there is no need to look for improvements: positive existence is equivalent to finding a path along which all objectives are fulfilled.

Proposition 9.17 *Deciding the symmetric existence problem in symmetric game networks is at least as hard as deciding the positive symmetric existence problem. The reduction doubles the number of players and uses* LTL *objectives, but does not change the nature of the strategies (memoryless, bounded-memory, or general).*

This result is a consequence of the following lemma, which we prove below.

Lemma 9.18 *Let $\mathcal{G}$ be an n-player symmetric game network and t_0 be an initial configuration in $\mathcal{G}$. We can construct in polynomial time a $(2n)$-player symmetric game network $\mathcal{G}'$ and a configuration t'_0 in $\mathcal{G}'$ such that there is a Nash equilibrium from t_0 along which all players win in $\mathcal{G}$ if and only if there is a Nash equilibrium from t'_0 in $\mathcal{G}'$. Moreover, this equivalence also holds for memoryless and bounded-memory equilibria.*

Proof. Let $\mathcal{G} = (G, (\equiv_i)_{i\in[n]}, (\Omega_i)_{i\in[n]})$ be a symmetric game network with n players and assume the symmetric representation $(\pi_{i,j})_{i,j\in[n]}$. Let $t_0 = (s_0, ..., s_{n-1})$ be the initial configuration.

The overall idea is that players in $\{0, ..., n-1\}$ still play in the same arena G, but before entering G each player i in this set plays a matching pennies game against player $i + n$. Players in $\{n, ..., 2n-1\}$ only play the matching pennies game. This is accomplished by using disconnected parts of the arena as shown in Figure 9.4.

Thus, the initial configuration t'_0 in $\mathcal{G}'$ is given the initial state (s_i, p_0) for player $i \in \{0, ..., n-1\}$ and the initial state q_0 for player $i \in \{n, ..., 2n-1\}$.

The objectives of each player $i \in \{n, ..., 2n-1\}$ is simply to win the matching pennies game he plays against player $i-n$. The objectives of each player $i \in \{0, ..., n-1\}$ is to either win the matching pennies game or accomplishing the same objective as he has in $\mathcal{G}$. Note that this transformation of objectives can easily be done using a slightly larger LTL formula than the one expressing the objective in $\mathcal{G}$.

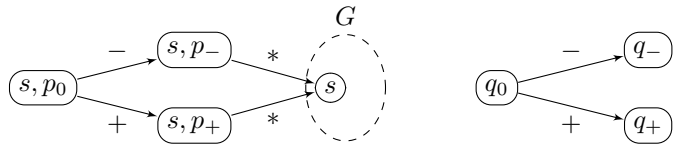


Figure 9.4: The arena G'. Each player $i \in \{0, ..., n-1\}$ will start in a module like the one to the left with initial state (s_i, p_0). Each player $i \in \{n, ..., 2n-1\}$ will start in the module to the right in q_0..

It can be shown that there exists a symmetric representation for $\mathcal{G}'$ that extends π. For details see the journal paper [BMV16].

We are now ready to show that there is a positive Nash equilibrium in $\mathcal{G}$ from t_0 if and only if there is a Nash equilibrium in $\mathcal{G}'$ from t'_0.

For the first direction suppose that there is a positive Nash equilibrium σ in $\mathcal{G}$ from t_0. Then there is a (positive) Nash equilibrium σ' in $\mathcal{G}'$ where players in $\{0, ..., n-1\}$ play like in σ when the play reaches G and all players play in the matching pennies game such that every player $i \in \{n, ..., 2n-1\}$ wins his matching pennies game against player $i-n$. As all players win in σ' it is a positive Nash equilibrium.

For the other direction suppose that there is a Nash equilibrium σ' in $\mathcal{G}'$ from t'_0.

First suppose for contradiction that there is a player i that has payoff 0 in σ' from t'_0. Then this player can deviate in the initial matching pennies game and receive payoff 1 if all other players play according to σ'. This is a contradiction since σ' is a Nash equilibrium. Thus, every player has payoff 1 in σ' from t'_0.

As every player has payoff 1 this means that all players in $\{n, ..., 2n-1\}$ win their matching pennies game and thereby all players in $\{0, ..., n-1\}$ lose their matching pennies game. Thus, as players in $\{0, ..., n-1\}$ all win in σ' from t'_0 this means that they play in a way in the G modules such that they all win. They can use that same strategy to obtain a positive Nash equilibrium in $\mathcal{G}$ from t_0.

□

9.4 Existence in symmetric game networks

Recent works have considered the computation of Nash equilibria in standard concurrent or turn-based games. In particular, the abstraction of suspect games described in [Bou+12] has allowed the development of efficient algorithms for computing Nash equilibria in concurrent games, for various classes of objectives. However those algorithms cannot be applied to our framework for the following reasons:

- each player has only *partial observation* of the global state of the game;
- the *symmetry requirement* induces non-local constraints in the concurrent game resulting from the product of the one-player arenas.

Notice that even in the case of symmetric games with full observation, an approach using *Strategy Logic* [MMV10], which can express Nash equilibria and impose several players to play the same strategy, would not work out-of-the-box, as in our setting strategies are equal *up to a permutation of the states*.

We now list the results we have obtained about computing Nash equilibria in symmetric game networks. We begin with undecidability results for the following cases:

- non-regular objectives (for two players, full observation)
- partial observation (for three players, LTL objectives)
- parametrized number of players (LTL objectives, partial observation, memoryless strategies)

We prove decidability when the number of players is given in the input and there is a restriction to bounded memory strategies.

9.4.1 Undecidability with non-regular objectives

Our games allow for arbitrary Boolean objectives, defined for each player as a set of winning plays. We prove that it is too general to get decidability of our problems even with full observation.

Theorem 9.19 *The existence of a symmetric Nash equilibrium for non-regular objectives in two-player symmetric game networks is undecidable (even with full observation).*

Proof. We do a reduction of the halting problem for a deterministic two-counter machine, which is known to be undecidable [Min61]. Let M be a deterministic two-counter machine and let (q_0, c_0, d_0) be an initial configuration. From this we create a symmetric game network with two players $\mathcal{G} = (G, (\equiv_i)_{i\in[2]}, (\Omega_i)_{i\in[2]})$ where $(s_0, s_1) \equiv_i (s'_0, s'_1)$ if and only if $s_0 = s'_0$ and $s_1 = s'_1$ for $i = 0, 1$. The arena G consists of two disconnected parts. It is shown in Fig. 9.5, but without G'.

Figure 9.5: Illustration of G where G' simulates the two-counter machine M.

The idea is that player 0 starts in $s_{0,0}$ and player 1 starts in $s_{1,0}$. They first play a matching pennies game and then player 1 plays in G' which simulates the counter machine M. We design the objectives so that player 1 wins if he acts according to the rules of the counter machine and reaches a halting state. If he does not reach a halting state, he wins if he chose an action different from that of player 0 in the initial matching pennies game; otherwise player 0 wins. This way, if there is a legal, halting run of the counter machine, then there is a Nash equilibrium where player 1 wins and player 0 loses. If there is no legal halting run then the game is essentially reduced to a matching pennies game which has no Nash equilibrium.

We do this by letting G' have the control states of M as states with the state connected to $s_{1,+}$ and $s_{1,-}$ being the initial control state q_0 of M. Then for all control states q_i, q_j such that there is a transition from q_i to q_j increasing counter C in M there is an action C^+ in G' taking the play from q_i through an intermediate state C^+_{ij} to q_j as illustrated in Figure 9.6 to the left.

Figure 9.6: Constructions of the incrementation and decrementation modules.

Similarly, for a transition in M that goes from state q_i and

- decreases counter C if the value is positive and changes control state to q_j
- leave counter C unchanged if the value is 0 and changes control state to q_k

we have actions C^- and C^0 in G' respectively taking the play from q_i through intermediate states C^-_{ij} to q_j and C^0_{ik} to q_k as shown in Figure 9.6.

Additionally, we add a self-loop to the halting state q_F in G'. For a finite path ρ and a counter C we now define its value by the end of ρ by

$$C_\rho = \left|\{k \mid \rho_k = C^+_{ij} \text{ for some } i,j\}\right| - \left|\{k \mid \rho_k = C^-_{ij} \text{ for some } i,j\}\right|$$

When given an initial value c_0 and d_0 of the two counters of M we then define the objectives such that player 1 loses in all plays ρ that contains a prefix $\rho_{\leq k}$ such that state $\rho_k = C_{ij}^-$ and $C_0 - C_{\rho_{\leq k}} < 0$ for some C, i and j to make sure player 1 plays according to the rules of the counter machine and does not subtract from a counter with value zero. In addition, he loses in all plays ρ that contains a prefix $\rho_{\leq k}$ such that $\rho_k = C_{ij}^0$ and $C_0 - C_{\rho_{\leq k}} \neq 0$ to make sure player 1 does not follow the true branch of a zero test when the value of the counter being tested is not zero. Finally, player 1 wins if he does not violate any of these restrictions and reaches q_F. He also wins if he wins the matching pennies game initially, no matter whether he violates the restrictions or not. Player 0 simply wins whenever player 1 does not win.

In total this means that there is a Nash equilibrium where player 1 wins and player 0 loses if M halts with initial counter values c_0 and d_0. If M does not halt with initial values c_0 and d_0 the game is reduced to a matching pennies game which has no Nash equilibrium. Thus, there is a Nash equilibrium in $\mathcal{G}$ if and only if M halts, implying that the existence problem is undecidable.

The partially defined strategies specified for the two players in the reduction can trivially be extended to symmetric strategies which makes the symmetric existence problem undecidable as well. □

9.4.2 Undecidability with partial observation

We already mentioned an undecidability proof in Theorem 9.19 for two players and full observation. However, the objectives used for achieving the reduction are quite complex. We explain here how partial observation also leads to undecidability, but for LTL objectives, and with only three players.

To show this, we can slightly alter a proof from [PR90]. There, synthesis of distributed reactive systems with LTL objectives is shown undecidable in the presence of partial observation. The situation used in that proof, where two processes (players 0 and 1) with an LTL objective φ play against a hostile environment (player 2), can be modelled in our framework. The idea is that φ is built from a deterministic Turing machine $\mathcal{M}$ in such a way that the processes can win if and only if $\mathcal{M}$ halts on the empty input tape.

On top of this reduction, we add an initial matching-pennies module between player 0 and 1, and slightly change the LTL objectives as follows:

- Players 0 and 1 still win if φ is true, but each player can also win by winning the initial matching-pennies game.
- Player 2 still wins if φ is not true.

Now, if $\mathcal{M}$ halts on the empty input tape, then there is a Nash equilibrium where players 0 and 1 play in such a way that φ is true; they both win, while player 2 loses and has no winning deviation.

On the other hand, suppose $\mathcal{M}$ does not halt on the empty input tape. Let $\sigma = (\sigma_0, \sigma_1, \sigma_2)$ be a given strategy profile. If player 2 is losing along the play of σ, then he can change his strategy and improve, since $\mathcal{M}$ does not halt on the empty input tape; thus σ is not a Nash equilibrium. On the other hand, if player 2 is winning along the play of σ, then one of players 0 and 1 is losing. But then, this player can improve by changing his strategy in order to win the initial matching pennies game. Thus, σ is not a Nash equilibrium in this case either. This implies that there exists a Nash equilibrium if, and only if, $\mathcal{M}$ halts on the empty input tape.

Theorem 9.20 *Deciding the existence of a symmetric Nash equilibrium for* LTL *objectives in symmetric game networks is undecidable for $n \geq 3$ players.*

9.4.3 Decidability for memoryless strategies

In this section we prove that the existence of a memoryless symmetric Nash equilibrium is decidable for LTL objectives, and that this problem is PSPACE-complete. Notice here that the input of the observation relations $\equiv_i$ are already of size $|S|^n \times |S|^n$. In the next section we consider more succinct encodings for these relations.

We first observe that PSPACE-hardness is a direct consequence of the proof of PSPACE-hardness of model-checking of LTL in finite-state transition systems [SC85].

We now explain our algorithm for deciding the (constrained) existence of symmetric Nash equilibria restricted to memoryless strategies. The algorithm is as follows: it first guesses a memoryless strategy for one player, from which it deduces the strategies to be played by the other players. It then looks for the players that are losing, and checks if they alone can improve their payoff.

More formally, we fix a symmetric game network $\mathcal{G} = (G, (\equiv_i)_{i\in[n]}, (\Omega_i)_{i\in[n]})$ with symmetric representation $\pi = (\pi_{i,j})_{i.j\in[n]}$ and an initial configuration t_0. We assume that each objective Ω_i is given by an LTL formula ϕ_i.

The first step is to guess and store an $\equiv_0$-realizable memoryless strategy σ_0 for player 0. Such a strategy is a mapping from S^n to the set Σ of actions in G; following our remark above about the size of the input, such a strategy has size polynomial in the size of the input. We intend player 0 to play according to σ_0, and every player i to play according to $\sigma_0(\pi_{i,0}^{-1}(s_0, ..., s_{n-1}))$ for every configuration $(s_0, ..., s_{n-1})$. From Lemma 9.12, we know that all symmetric memoryless strategy profiles can be characterized by such an $\equiv_0$-realizable memoryless strategy for player 0.

The algorithm first checks which players are winning and losing in the unique play ρ induced by the strategy profile σ from t_0. Since all players use memoryless strategies ρ is an ultimately periodic path, i.e. of the form $\alpha \cdot \beta^\omega$ for two finite sequences α, β of states which are both polynomial in the size of the input. For the constrained problem it is checked whether the constraints on which players should win and lose are satisfied.

Next, the algorithm checks if σ is a Nash equilibrium. This is done by checking that no player can improve his payoff. The way to do this is, for each player i losing in ρ, to build a transition system $\mathcal{T}$ representing all the possible plays starting in

t_0 given that all players except player i play according to σ. That is, $\mathcal{T}$ represents exactly the different ways that player i can deviate given that all other players play according to σ.

It is now checked whether there is a play in this transition system which is winning for player i using the standard model-checking algorithm for LTL. If there is, he can deviate and improve his payoff. If there is no such play he cannot deviate and improve. If none of the losing players can deviate and improve then σ is a Nash equilibrium, otherwise it is not.

As the state space S^n is polynomial in the size of the input, this procedure can be performed in polynomial space.

Theorem 9.21 *The constrained existence of a memoryless symmetric Nash equilibrium for* LTL *objectives in symmetric game networks is* PSPACE*-complete.*

Notice that this algorithm can be extended to bounded-memory strategies. The algorithm would then require exponential space if the bound on the memory is given in binary, and still polynomial space otherwise.

Notice also that the algorithm above could be adapted to handle non-symmetric bounded-memory equilibria in non-symmetric game networks: it would just guess all the strategies, and check the satisfaction of the LTL objectives in the product automaton obtained by applying the strategies.

The algorithm could also be adapted, still with the same complexity, to handle richer objectives as in the semi-quantitative setting of [Bou+12], where players have several preordered objectives. Instead of guessing the winners, the algorithm would guess, for each player, which objectives are satisfied, and check that no individual improvement is possible. This can be done by listing all possible improvements and checking that none of them can be accomplished.

9.5 Succinct symmetric game networks

As our goal is to represent large networks of components it is not feasible to store the entire observation relation explicitly for all players since this can be very large. In this section we investigate a succinct representation for symmetric game networks that can be represented symbolically in a succinct way.

A *succinct symmetric game network* is a tuple $\mathcal{P} = (G, (\alpha_j)_{j\in[k]}, \equiv, \phi)$ where G is a one-player arena with the set S of states, $\alpha_j \colon \mathbb{N} \to (\mathbb{N} \to \mathbb{N})$ indicate the k *neighbours* of each player, and $\equiv$ and ϕ are *templates* for defining $\equiv_i$ and ϕ_i for each player. We now explain how a symmetric game network $\mathcal{G} = \mathcal{P}^n$ can be obtained from a succinct symmetric game network $\mathcal{P}$ and an integer $n \geq k$.

For a given n, the state space of $\mathcal{P}^n$ is S^n. Then each $\alpha_j(n)$ is a mapping $[n] \to [n]$; the integer $\alpha_j(n)(i)$ represents the j-th neighbour of player i. We require that the mappings α_j are represented symbolically, e.g. as arithmetic expressions involving j and the arguments n and i. We explain below how this partially defines the symmetric representation for $\mathcal{P}^n$.

The equivalence relation $\equiv$ is a relation over $(S^k \times \mathbb{N}^S)$: the first component deals with the k neighbours of each state, while the second component contains information about the Parikh image of the configurations[1]. That is, information about how many players are in certain states (but not about which players are in those states). For any $i \in [n]$, and for any two configurations t and t' in S^n, we let

$$t \equiv_i t' \Leftrightarrow (t_{\alpha_j(i)})_{j\in[k]} \times (\#_s(t))_{s\in S} \equiv (t'_{\alpha_j(i)})_{j\in[k]} \times (\#_s(t'))_{s\in S}.$$

where $\#_s(t)$ is the number of occurences of s in the configuration t.

For instance, in order to define exact observation of the left- and right neighbours, we would define $\alpha_1(n)(i) = i-1 \pmod n$, $\alpha_2(n)(i) = i+1 \pmod n$, and let $\equiv$ relate any two tuples as soon as their first two items (t_1, t_2) and (t'_1, t'_2) match.

Similarly, ϕ is an LTL formula from which the objectives of the players can be derived: the formula is built on two types of atomic propositions:

- For each atomic proposition p appearing in G, and for any $j \in [k]$, p_k is an atomic proposition;
- Formulas of the form $\#_s \sim c$ and $\#_s - \#_{s'} \sim c$ are atomic propositions, where $\sim \in \{<, \leq, =, \geq, >\}$, $c \in \mathbb{N}$ and s and s' are states.

For each $i \in [n]$, formula ϕ_i is then obtained by replacing p_k with $p_{\alpha_k(n)(i)}$. The semantics of these atomic propositions is defined as follows:

- $p_{\alpha_k(n)(i)}$ holds true in configuration t if the label of state $t(\alpha_k(n)(i))$ contains p;
- $\#_s - \#_{s'} \sim c$ holds true in t if, writing n_s for the number of occurrences of s in t and $n_{s'}$ for the number of occurrences of s' in t, it holds $n_s - n_{s'} \sim c$. Similarly for $\#_s \sim c$.

It remains to be seen under which conditions the resulting game network $(G, (\equiv_i)_{i\in[n]}, (\phi_i)_{i\in[n]})$ is a symmetric game network: for this, we need to prove the existence of a symmetric representation π. This puts contraints on $(\alpha_j)_{j\in[k]}$, depending on $\equiv$ and ϕ. In the general case (omitting trivial cases where e.g. $\equiv$ is the identity relation, or ϕ is always true), the condition $t \equiv_i t' \Leftrightarrow t(\pi_{i,j}^{-1}) \equiv_j t'(\pi_{i,j}^{-1})$ might give rise to conditions $\pi_{i,j}(\alpha_l(n)(i)) = \alpha_l(n)(j)$ on the symmetric representation. This corresponds to our intuition that the role of player $\alpha_l(n)(j)$ w.r.t. j (namely, being his l-th neighbour) is the same as the role of $\alpha_l(n)(i)$ w.r.t. i. In particular, this in general implies that if $\alpha_l(n)(i) = \alpha_{l'}(n)(i)$ for some i, then $\alpha_l(n)(j) = \alpha_{l'}(n)(j)$ for all $j \in [n]$.

Finally, the initial configuration of a succinct game network is given as a function mapping each integer $n \geq k$ to a configuration in S^n. This can for instance be given as a sequence of pairs (s_j, ψ_j) where $s_j \in S$ and ψ_j is a boolean valued function

[1]Notice that this slightly differs from the Parikh condition we used in Example 9.10: there several conditions would be imposed on Parikh images of different subsets of neighbours. The setting defined here could easily be extended to this case.

taking n and i as argument. Then, in $\mathcal{P}^n$, player i would have initial state s_j for the smallest j for which $\psi_j(n,i)$ is true (requiring that $\psi_l \equiv \top$ for some l, so that such a j always exists).

9.5.1 Undecidability of parameterized existence

In this section we study the problem of deciding whether a succinct symmetric game network admits a symmetric Nash equilibrium when the number of players is large enough. More precisely, we aim at deciding the existence of a one-player strategy σ_0, and of an integer n_0, such that the strategy profile obtained by making all n_0 players follow strategy σ_0 (each player having its own observation) is a Nash equilibrium. This is called the *parameterized existence problem*. We show that this problem is undecidable for LTL objectives, even when considering only memoryless strategies.

Theorem 9.22 *The parameterized existence problem for* LTL *objectives in succinct symmetric game networks is undecidable (even for memoryless strategies).*

We first give a proof for undecidability of the parameterized existence problem for positive symmetric Nash equilibria and then describe how to do a similar construction without the positivity constraint.

The proof is done by reduction from the halting problem for deterministic Turing machines. Let $\mathcal{M} = (Q_{\mathcal{M}}, q_0, \Sigma_{\mathcal{M}}, \delta_{\mathcal{M}}, q_F)$ be a deterministic Turing machine. We build a succinct symmetric game network $\mathcal{P}$ that captures the behaviour of $\mathcal{M}$. We intend to enforce that $\mathcal{M}$ halts if, and only if, there exists n such that $\mathcal{P}^n$ has a positive symmetric Nash equilibrium from some initial configuration. Moreover, we will show that there is a positive Nash equilibrium in $\mathcal{P}^n$ if and only if there is a memoryless one.

Intuitively, the idea is to have each player control a cell of the Turing machine. Thus, in each state of the game each player can alter the contents of the cell. Each player can in each step also claim that the tape head currently points to his cell and that the current control state of the Turing machine is some state q.

Each player can observe the local states of the two players controlling the cell to the left and right respectively. This information is enough to be able to update the cell correctly according to the rules of the Turing machine (assuming that the other players play according to the rules of the Turing machine as well).

Using LTL objectives for the players it can be expressed how a player should play in order to follow the rules of the Turing machine. Included in this objective is also that from some point onwards each player will keep playing the same action forever. If all players follow the rules of the Turing machine this can only be accomplished if the Turing machine has an accepting run.

If $\mathcal{M}$ halts on the empty input tape then the smallest number n of players such that there exists a symmetric positive Nash equilibrium in $\mathcal{P}^n$ can be shown to be equal to $n_1 + 2$, where n_1 is the number of cells used by $\mathcal{M}$ during the halting run. If

$\mathcal{M}$ does not halt it can be shown that there is no symmetric positive Nash equilibrium for any n.

We first define the one-player arena $G = (S, \{A\}, \Sigma, \Gamma, \delta)$, which is depicted on Fig. 9.7, as follows:

- $S = ((Q_{\mathcal{M}} \times \Sigma_{\mathcal{M}}) \cup \Sigma_{\mathcal{M}} \cup \{q_F\}) \times \{L, \#, R\}$
- $\Sigma = \Sigma_{\mathcal{M}} \cup Q_{\mathcal{M}}$
- For $q \in Q_{\mathcal{M}}, a \in \Sigma_{\mathcal{M}}$ and $\bullet \in \{L, \#, R\}$ let
  - $\Gamma(((q, a), \bullet), A) = \Sigma_{\mathcal{M}}$ if $q \neq q_F$
  - $\Gamma((a, \bullet), A) = Q_{\mathcal{M}} \cup \{a\}$
  - $\Gamma(((q_F, a), \bullet), A) = \Gamma(q_F) = \{q_F\}$
- For $q \in Q_{\mathcal{M}}, a, b \in \Sigma_{\mathcal{M}}$ and $\bullet \in \{L, \#, R\}$ let
  - $\delta(((q, a), \bullet), b) = (b, \bullet)$ if $q \neq q_F$
  - $\delta((a, \bullet), q) = ((q, a), \bullet)$
  - $\delta((a, \bullet), a) = (a, \bullet)$
  - $\delta(((q_F, \bullet), a), q_F) = \delta((q_F, \bullet), q_F) = (q_F, \bullet)$

Each state in the arena is marked with a special symbol in $\{L, \#, R\}$: letters L and R are used to indicate the left-most and right-most cells of the tape, while $\#$ identifies all other cells.

In this reduction, we let $k = 3$ (each player observes two neighbours plus himself), with $\alpha_1(n)(i) = i - 1 \pmod n$, $\alpha_2(n)(i) = i$, and $\alpha_3(n)(i) = i + 1 \pmod n$. The observation relation $\equiv$ is the identity on S^3, with no condition on the Parikh images. This defines a ring topology where each player has full observation of himself and of his left and right neighbours.

We now define the objectives of the players, by describing an LTL formula φ. For the sake of readability, we use atomic propositions p_{-1}, p and p_{+1} (instead of p_1, p_2 and p_3, respectively), representing the value of atomic proposition p for players $\alpha_1(n)(i)$, $\alpha_2(n)(i)$ and $\alpha_3(n)(i)$. The LTL formula φ is given in Fig. 9.8.

The formula describes how the player should update the content of his cell according to what is observed about the right and left neighbour and the current content of his own cell. Next, it is described that eventually the player will keep playing the same action forever. Finally, the two cells at the left border and right border of the tape must always be blank.

All cells of the tape initially contains the blank symbol $\flat$, we set the initial configuration of the network to be $(\flat, \#)$ for all players, except for players 0, 1 and 2, starting respectively in states $(\flat, R)$, $(\flat, L)$ and $((q_0, \flat), \#)$. We write γ^n for this initial configuration.

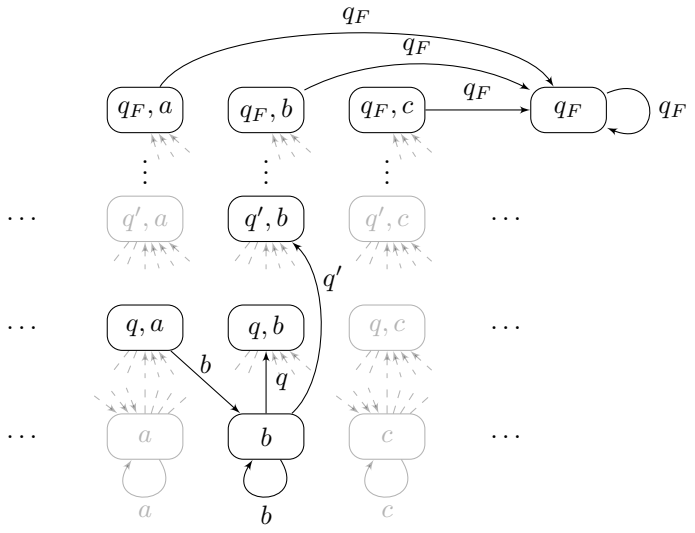


Figure 9.7: The one-player arena (with transitions for one 4-tuple (q, q', a, b)). Note that the second component $\bullet \in \{L, \#, R\}$ of the states are omitted.

Lemma 9.23 *If $\mathcal{M}$ halts, then there exists n and a memoryless strategy σ that induces a positive symmetric memoryless Nash equilibrium in $\mathcal{P}^n$ from γ^n.*

Proof. Suppose $\mathcal{M}$ halts. Then the unique finite run ρ of $\mathcal{M}$ uses only a finite number of tape cells. Let this number be n_1, and consider the game $\mathcal{P}^n$, for any $n \geq n_1 + 2$.

In this game, define a memoryless strategy such that action $\flat$ is always chosen from states $(\flat, L)$ and $(\flat, R)$ (since the tape head never points to these positions). For states with second component $\#$, let the choice of action for player $i \in \{1, ..., n_1\}$ in round k of the game correspond to the content of cell i in the k-th step of the run ρ. It also means playing the control state of $\mathcal{M}$ when the tape head in ρ moves to cell i. As player i can see the contents of the two cells $i+1$ and $i-1$, as well as cell i, he can derive from the current state what to play next in this strategy profile. Thus, this can be done using a memoryless strategy. As every player can use this strategy, this induces a symmetric memoryless strategy profile.

As φ_i expresses that player i plays exactly according to the rules of the Turing machine, that some player eventually reaches the q_F state (because it requires all players to eventually stay in the same state), and that the state of players beginning in $(\flat, L)$ and $(\flat, R)$ never change states, this strategy profile ensures that every player wins as ρ is a halting run.

Thus, the strategy profile defined is a memoryless positive symmetric Nash equilibrium in $\mathcal{P}^n$. □

$$\varphi = \bigwedge_{\substack{(q,a)\in Q\times\Sigma \text{ s.t.}\\ \delta(q,a)=(q',b,r),\ c,d\in\Sigma\\ \gamma_1,\gamma_2,\gamma_3\in\{L,\#,R\}}} \left[\begin{array}{l} \mathbf{G}\left[\begin{array}{l}\Big(((q,a),\gamma_1)\wedge(c,\gamma_2)_r\wedge(d,\gamma_3)_{-r}\Big)\Rightarrow\\ \qquad \mathbf{X}\Big((b,\gamma_1)\wedge((q',c),\gamma_2)_r\wedge(d,\gamma_3)_{-r}\Big)\end{array}\right]\\ \wedge\\ \mathbf{G}\left[\begin{array}{l}\Big((c,\gamma_1)\wedge(d,\gamma_2)_r\wedge((q,a),\gamma_3)_{-r}\Big)\Rightarrow\\ \qquad \mathbf{X}\Big(((q',c),\gamma_1)\wedge(d,\gamma_2)_r\wedge(b,\gamma_3)_{-r}\Big)\end{array}\right]\end{array}\right]$$

$$\wedge \bigwedge_{\substack{a,b,c\in\Sigma\\ \gamma_1,\gamma_2,\gamma_3\in\{L,\#,R\}}} \mathbf{G}\Big[\Big((a,\gamma_1)_{-1}\wedge(b,\gamma_2)\wedge(c,\gamma_3)_{+1}\Big)\Rightarrow \mathbf{X}(b,\gamma_2)\Big]$$

$$\wedge\ \mathbf{F}\bigvee_{s\in S}\mathbf{G}s\ \wedge \bigwedge_{\gamma\in\{L,R\}}\Big((\flat,\gamma)\Rightarrow\mathbf{G}(\flat,\gamma)\Big)$$

Figure 9.8: Formula φ. It expresses that a player should play according to the rules of the Turing machine and that eventually it should keep staying in the same state forever. Finally, the left-most and right-most cell should always contain the blank symbol $\flat$.

Lemma 9.24 *If $\mathcal{M}$ does not halt, then there exists no n for which there is a positive Nash equilibrium in $\mathcal{P}^n$ from γ^n.*

Proof. We do the proof by contraposition. Suppose that there exists n such that there is a positive symmetric Nash equilibrium σ in $\mathcal{P}^n$ from γ^n. Then the unique play ϱ of the associated strategy profile from γ^n satisfies φ_i for all $0 \leq i \leq n-1$. In particular, player 0 and player 1 always choose action $\flat$ and stay in the states $(\flat, L)$ and $(\flat, R)$ respectively.

Further, as φ_i is satisfied in ϱ for all other players, the topmost conjuncts in the definitions of the formulas imply that the players must play according to the unique run ρ of $\mathcal{M}$. The truth of the formula also implies that one of the players eventually plays the halting state as all players eventually keep staying in the same state. This means that $\mathcal{M}$ does in fact halt. □

Theorem 9.22 now follows from Lemmas 9.23 and 9.24. In particular, note that they imply undecidability both with and without the restriction to memoryless strategies.

Note also that the proof above is only for the restriction to positive equilibria. However, using techniques similar to the proof of Proposition 9.17, the proof can be adapted to handle unconstrained Nash equilibria.

9.5.2 Decidability with bounded memory

In this section, we keep the setting of succinct representations for the observation relation and for the LTL objectives, but fix the number of players. We prove that the existence of a memoryless (or even bounded-memory) symmetric Nash equilibrium is decidable, and that it is ExpSpace-complete. Notice that we assume that the number of players is given in binary, so that the state space S^n is actually doubly-exponential in the size of the input.

We first notice that ExpSpace-hardness is a direct consequence of the proof of Theorem 9.22; the only difference is that we have to consider exponential-space Turing machines. The reason that the size of the tape is exponential in this reduction is that there is one cell for each player. Here, the crucial point is that the number n of players is given in binary in the succinct representation.

The algorithm follows the same line as in Section 9.4.3: it guesses a memoryless strategy to be stored on the tape, and checks that no player has a profitable deviation by guessing paths step-by-step. The strategy maps each observation set to an action. The number of observation sets is the number of different equivalence classes in $\equiv$: the number of different Parikh images of size n over S is bounded by n^S, and the number of different configurations for the k neighbours is S^k. Here k can be assumed to be given in unary, since the input contains one function α_j for each $0 \leq j \leq k-1$. Hence the number of observation sets is exponential, and the strategy can be guessed and stored using exponential space.

Checking whether a player meets his objective or has an incentive to deviate from the guessed strategy can be achieved in exponential space, following the same ideas as in Section 9.4.3.

Theorem 9.25 *Deciding the constrained existence of a memoryless symmetric Nash equilibrium for* LTL *objectives in succinct symmetric game networks is* ExpSpace*-complete.*

As for the case of non-succinct symmetric game networks, this algorithm can be lifted to handle finite-memory strategies. Here, the problem remains ExpSpace-complete, even when the memory bound is given in binary.

The algorithm can also be adapted to handle non-symmetric equilibria, by guessing and storing exponentially many memoryless strategies (one for each player).

9.6 Summary

In this chapter, we have proposed a model of games for large networks of identical devices. This model of games is composed of a single arena, which is duplicated (one

copy for each player), and each player only partially observes the global state of the system. To fully represent large networks of identical devices, we added symmetry constraints, which yields non-local constraints in the system.

For this model, we have studied several problems related to the computation of symmetric pure Nash equilibria where the objective of each player is given by an LTL formula. We have fully characterized the complexity of the existence and constrained existence problems for bounded-memory strategies. That is, to decide respectively whether there exists a bounded-memory symmmetric Nash equilibrium and whether there exists a bounded-memory symmetric Nash equilibrium where certain players are constrained to be winning and certain players are constrained to be losing. Further, we have proved several undecidability results when the memory of the strategies is unbounded.

This work opens many interesting directions of research. Besides solving the questions left open in this chapter, these directions include the study of randomized Nash equilibria in such networks of games. Other possibilities for further work include extended quantitative objectives, or stronger solution concepts, like sub-game perfect equilibria [Osb04] or secure equilibria [CHJ06]. Restriction to interesting subclasses of observation relations and network topologies is also important to find meaningful special cases with lower complexity.

Finally, it would be interesting to investigate whether the symmetry reduction technique of Chapter 6 could be adapted to this setting. Indeed, there are some challenges with respect to partial observation and the symmetry constraints on strategies in symmetric strategy profiles. Also, it is not clear how to deal with Nash equilibria rather than winning strategies .

CHAPTER 10

Distributed synthesis in continuous time

This chapter is an adapted version of the paper

- [HKV16] Holger Hermanns, Jan Krcál, and Steen Vester. "Distributed Synthesis in Continuous Time". In: *Foundations of Software Science and Computation Structures (FOSSACS) - 19th International Conference.* 2016, pages 353–369

It has been updated to be compatible with the other chapters of the thesis. In particular, the introduction has been updated and extended with more explanation of the semantics of the modelling formalism. Some of the more technical proof details have been relegated to the appendix or left out for increased readability. In these cases proof sketches are included in the main text.

10.1 Introduction

Distributed systems interact in real time, and one very general way to reason about their timing behaviour is to assume that arbitrary continuous probability distributions govern the timing of local steps as well as communication steps. We are interested in how foundational properties of such distributed systems differ from models where timing is abstracted away as was the case in all previous chapters.

We also model communication explicitly in this chapter. As principal means of communication we consider symmetric handshake communication, since it can embed other forms of communication faithfully [Mil83; BK08] including asynchronous and input/output-separated communication.

As an example, consider the problem of leaking a secret from a sandboxed malware to an attacker. The behaviour of attacker and malware (and possibly other components) are prescribed in terms of states, private transitions, labelled synchronization transitions, and delay transitions which model both local computation times and synchronization times. The delays are governed by arbitrary continuous probability distributions over real time. Handshake synchronization is assumed to take place if all devices able to do so agree on the same transition label. Otherwise the components run fully asynchronously. The sandboxing can be thought of as restricting the set of labels allowed to occur on synchronization transitions.

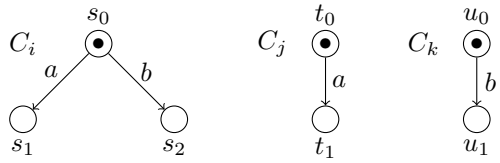


Figure 10.1: Possibility of synchronizing on both a and b.

The question we focus on is how to synthesize the component control strategies so that they reach their target almost surely or with at least a given probability p. In this example, by synthesizing control strategies for malware and attacker with a high probability of leaking the secret.

In this chapter we consider a parallel composition of n modules synchronizing via handshake communication. The modules are modelled by *interactive Markov chains* (IMCs) [Her02; HK09], a generalization of labelled transition systems and of continuous time Markov chains, equipped with a well-understood compositional theory. Each module may in each state enable private actions, as well as synchronization actions.

It is natural to view such a *distributed IMC* as a game with $n+1$ players, where the last player (the scheduler) controls the interleaving of the modules. Each of the other n players controls the decisions in a single module, only based on its local timed history containing only transitions that have occurred within the module.

On entering a state of its module, each player selects and commits to executing one of the actions available. A private action is executed immediately while a synchronization action requires a CSP-style handshake [BHR84], it is executed once all modules able to perform this action have committed to it.

For representing delay distributions, we make one decisive and one technical restriction. First, we assume that each distribution is continuous. This for instance disallows deterministic delays of, say, 3 time units. It is an important simplification assumed along our explorations of continuous-time distributed control. Second, we restrict to exponential distributions. This is a pure technicality, since (a) our results can be developed with general continuous distributions, at the price of excessive notational and technical overhead, and (b) exponential distributions can approximate arbitrary continuous distributions arbitrarily close [Neu81]. Together, these assumptions enable us to work in a setting close to interactive Markov chains.

Example 10.1 *Consider the example in Figure 10.1. Here there are three players i, j and k each with their own module. Initially, j has to commit to action a and k has to commit to action b. However, i has the possibility to choose either a or b. If i chooses a then the next configuration of the system is (s_1, t_1, u_0). It is not possible for player k to take a transition, because all players with the action b in their alphabet must synchronize on b for the transition to be taken.*

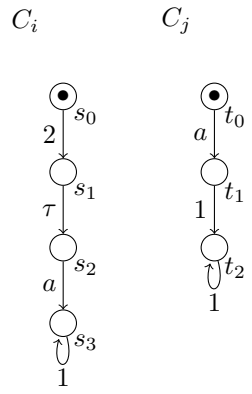


Figure 10.2: A distributed interactive Markov chain $\mathcal{G}$.

Example 10.2 *In Figure 10.2 we show a distributed IMC with two players i and j. In this example there are, in addition to synchronization transitions labelled a also an internal transition labelled τ. Finally, there are several* delay transitions *labelled with rates.*

Initially, j commits to a, but nothing happens as i is not yet ready to synchronize. Then, a random length delay occurs according to an exponential distribution with rate 2, say after time 1.42 has passed. Then the configuration becomes (s_1, t_0). Now, player i must commit to the internal transition which must occur immediately yielding configuration (s_2, t_0). Next, player i can only choose action a and immediately the two players synchronize on a resulting in configuration (s_3, t_1).

After this sequence, two numbers are chosen randomly according to exponential distributions both with rate 1, since both the delay transitions from s_3 to s_3 and t_1 to t_2 have rate 1. The lowest number d decides which transition occurs. Say, this is the transition from t_1 to t_2. The length of the delay until this transition occurs is then d. The history played so far is then

$$(s_0,t_0)(\bot,a) \overset{i,1.42}{\to} (s_1,t_0)(\tau,a) \overset{\tau,1.42}{\to} (s_2,t_0)(a,a) \overset{a,1.42}{\to} (s_3,t_1)(\bot,\bot) \overset{j,1.42+d}{\to} (s_3,t_2)$$

Note that the history contains both information about the actions committed to by players (and $\bot$ when no action is available) and the absolute time of transitions. Also, arrows are labelled with an action if an action occured and labelled with a player if a delay transition occured.

Finally, each player only observes the timed history of the behavior in his own module. Thus, for the history above, player j observes the following sequence

$$t_0\, a \overset{a,1.42}{\to} t_1 \bot \overset{j,1.42+d}{\to} t_2$$

Thus, while he cannot observe it, he can deduce that the delay transition in the module of player i occured at time 1.42.

In this example the interleaving scheduler did not have any choices since at no point were several different actions enabled during this history. We will see such examples later.

Apart from running in continuous time, the concepts behind distributed IMCs are rather common. Closely related are models based on probabilistic automata [Seg95] and (partially observable) Markov decision processes [Paz71; Ber+02]. In these settings, the power of the interleaving scheduler is a matter of ongoing debate [Che+06; Can+08; PD12]. The problem is that without additional (and often complicated) assumptions this player is too powerful to be realistic, and can for instance leak information between the other players. This is a problem, e.g. in the security context, making model checking results overly pessimistic [GDF14].

In sharp contrast to the discrete-time settings, in our distributed IMCs the interleaving scheduler does not have decisive influence on the resulting game. The reason is that the interleaving scheduler can only affect the order of transitions between two delays, but neither *which transitions* are taken nor what the different players *observe*. This is rooted in the common alphabet synchronization and especially the continuous-time nature of the game: the probability of two local modules changing state at the same time instant is zero, except if synchronizing.

Example 10.3 *We consider the model displayed in Figure 10.3 where the delay transitions are labelled by some rate λ. It displays a very simplistic malicious* App *trying to communicate a secret to an outside attacker, despite being sandboxed. Innocently looking actions* login, logout *and* lookup *synchronize* App *and* Att*, while the unlabelled transitions denote some private actions of the respective module.*

Initially, the App *can only let time pass. The attacker player has no other choice than committing to handshaking on action* login*. A race of the delay transitions will occur that at some point will lead to either state $(t_1, \bar{c}_1)$ or $(b_1, \bar{c}_1)$, with equal probability. Say in $(t_1, \bar{c}_1)$, the* App *player can only commit to action* login*. The synchronization will happen immediately since the attacker is committed to* login *already, leading to $(t_2, \bar{c}_2)$. Now the* App *player either has to commit to action* lookup *or* logout*. The latter will induce a deadlock due to a mismatch in players' commitments. Instead assuming the earlier, the state synchronously and immediately changes to $(t_3, \bar{c}_3)$. The attacker player can now use its local timed history to decide which of the private actions to pick. Whatever it chooses, an interleaving of private actions of the two modules follows in zero time. Unless the reachability condition considers transient states such as $(t_3, \bar{t}_4)$ where no time is spent, the player resolving the interleaving has no influence on the outcome.*

Now, assume the reachability condition is the state set $\{(t_4, \bar{t}_4), (b_4, \bar{b}_4)\}$. This corresponds to the attacker player correctly determining the initial race of the App*, and can be considered as a leaked secret. However, according to the explanations provided, it should be obvious that the probability of guessing correctly (by committing properly in state $\bar{c}_3$) is no larger than 0.5, just because the players are bound to decide only based on the local history.*

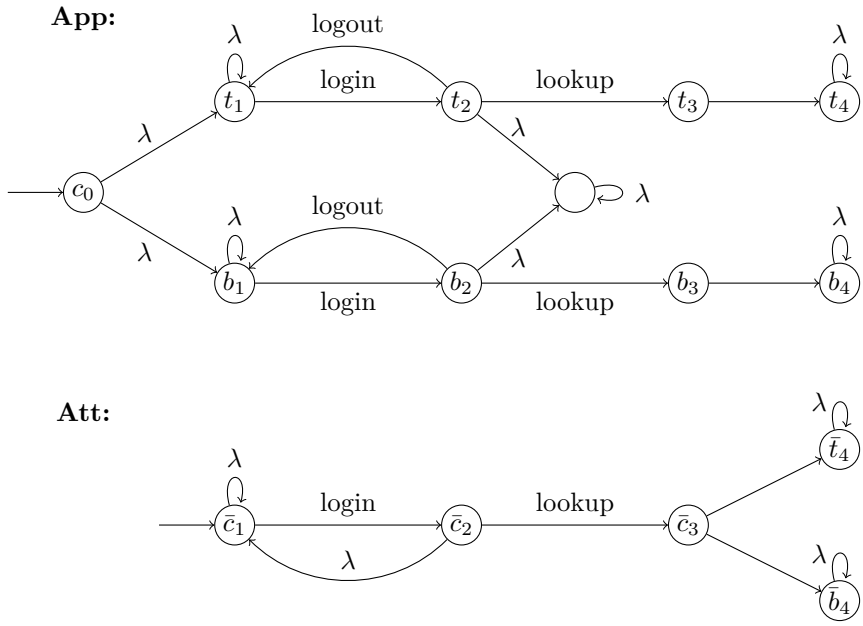


Figure 10.3: Model of malicious app and outside attacker.

Contribution

This chapter presents a new approach to explore distributed cooperative reachability games with continuous-time flow modelled explicitly. The formalism we study is based on interactive Markov chains and is called *distributed interactive Markov chains*.

We aim at synthesizing local strategies for the players to reach a specified set of goal states with at least a given probability. If this probability is 1 we call the problem *qualitative*, otherwise *quantitative*. We consider *existential* problems, asking for the existence of strategies with these properties, and *value* problems, asking for strategies approximating the given probability value arbitrarily close.

We have three main results:

1. We show that, under mild assumptions on the winning condition, in continuous-time distributed synthesis the interleaving scheduler has no power.

2. In general, we establish that the quantitative problems are undecidable for two or more players, the qualitative value problem is undecidable for two or more players and the qualitative existence problem is EXPTIME-hard for two players and undecidable for three or more players.

3. However, when focusing on the subclass of 2-player *non-urgent* distributed IMCs, the quantitative value problem can be solved in exponential time. Non-urgency

enables changing the decisions committed to after some time. Thus, it empowers the players to reach a distributed consensus about the next handshake to perform by observing the only information they jointly have access to: the advance of time.

The qualitative undecidability comes from a novel result we provide about decentralised partially observable Markov decision processes (DEC-POMDP), a multi-player extension of partially observable Markov decision processes (POMDP).

While qualitative existence is decidable for POMDP [BGB12], we show that qualitative existence is undecidable for DEC-POMDP already for 2 players.

By a reduction from DEC-POMDP to distributed IMCs that adds one player, we get undecidability of qualitative existence for 3 or more players in distributed IMCs.

Outline

In Section 10.3 we introduce the new formalism of distributed interactive Markov chains. Schedulers are shown not to have too much power in Section 10.4. In Section 10.5 we show several undecidability results for distributed IMCs and a new undecidability result for DEC-POMDPs. Decidability of the value problem is shown for the subclass of 2-player non-urgent models in Section 10.6. A summary of the chapter can be found in Section 10.7.

10.2 Preliminaries

We denote by $\mathbb{R}_{\geq 0}$ and $\mathbb{Q}_{>0}$ the sets of non-negative real numbers and positive rational numbers respectively. Furthermore, for a finite set X, we denote by $\Delta(X)$ the set of *discrete probability distributions* over X, i.e. functions $f : X \to [0,1]$ such that $\sum_{x \in X} f(x) = 1$. Finally, for a tuple x from a product space $X_1 \times \cdots \times X_n$ and for $1 \leq i \leq n$, we use functional notation $x(i)$ to denote the ith element of the tuple.

10.2.1 Markov decision proces (MDP)

A *discrete-time Markov decision proces* (see e.g. [Put09]) is a formalism for modelling systems with decision making and stochastic behavior. It can be seen as a one-player concurrent game where the transition function is randomized. In this thesis we only deal with finite-state Markov decision processes and therefore just refer to these as Markov decision processes.

Definition 10.4 *A Markov decision processes MDP is a tuple* $\mathcal{M} = (S, \Sigma, P, s^{in})$ *where*

- S *is a finite set of states*
- Σ *is a finite set of actions*

- $P : S \times \Sigma \to \Delta(S)$ *is a partial probabilistic transition function*
- $s^{in} \in S$ *is an initial state*

A *play* in $\mathcal{M}$ is an infinite sequence $\rho = s_0 a_0 s_1 a_1 \dots$ of states and actions such that $s_0 = s^{in}$ and $P(s_i, a_i)(s_{i+1}) > 0$ for every $i \geq 0$. A *history* is a prefix of a play. A *strategy* is a function σ that to every history h assigns a probability distribution over actions such that if an action a is assigned a non-zero probability, then $P(\text{last}(h), a)$ is defined. A strategy σ is *pure memoryless* if for any history it assigns probability 1 to some action and its choice depends only on the last state of the history. When we fix a strategy σ, we obtain a probability measure $\Pr^\sigma$ over the set of plays. For further details, see [Put09].

10.2.2 Decentralized POMDP (DEC-POMDP)

A *decentralized partially-observable Markov decision process* (DEC-POMDP) [Ber+02] is a multi-player extention of MDPs that incorporates partial observation of the players. The one-player case of DEC-POMDP is simply called a *partially-observable Markov decision process.*

Definition 10.5 *A DEC-POMDP is a tuple* $(S, \Pi, (\Sigma_i, \mathcal{O}_i)_{1 \leq i \leq n}, P, O, s^{in})$ *where*

- S *is a finite set of* global states *with* initial state $s^{in} \in S$*,*
- $\Pi = \{1, ..., n\}$ *is a finite set of* players*,*
- Σ_i *is a finite set of* local actions *of player* i *with* $\Sigma_i \cap \Sigma_j = \emptyset$ *if* $j \neq i$*, (by* $\Sigma = \Sigma_1 \times \cdots \times \Sigma_n$ *we denote the set of global actions),*
- $\mathcal{O}_i$ *is a finite set of* local observations *for player* i*,*
  (by $\mathcal{O} = \mathcal{O}_1 \times \cdots \times \mathcal{O}_n$ *we denote the set of* global observations*),*
- $P : S \times \Sigma \to \Delta(S)$ *is the* transition function *which assigns to a state and a global action a probability distribution over successor states, and*
- $O : S \times \Sigma \times S \to \Delta(\mathcal{O})$ *is the* observation function *which assigns to every transition a probability distribution over global observations.*

A DEC-POMDP starts in the initial state s^{in}. Assuming that the current state is s, one discrete step of the process works as follows:

- First, each player j chooses an action a_j.
- Then the next state s' is chosen according to the probability distribution $P(s, a)$ where $a = (a_1, \dots, a_n)$.
- Finally, each player j receives an observation $o_j \in \mathcal{O}_j$ such that the observations $o = (o_1, ..., o_n)$ are chosen with probability $O(s, a, s')(o)$.

Repeating this indefinitely, we obtain a *play* which is an infinite sequence $\rho = s_0 a_0 o_0 s_1 a_1 o_1 \cdots$ where $s_0 = s^{in}$ and for all $i \geq 0$ it holds that $s_i \in S$, $a_i \in \Sigma$, and $o_i \in \mathcal{O}$.

Note that the players can only base their decisions on the sequences of observations they receive rather than the actual sequence of states which is not available to them. For a more complete coverage of DEC-POMDPs, see [Ber+02].

10.3 Distributed Interactive Markov Chains

We first give a definition of a (local) module based on the formalism of *interactive Markov chains* (IMC) [Her02]. Then we introduce (global) distributed IMC.

IMCs are extensions of continuous-time Markov chains that incorporates composition in a similar way to in process algebra. The explicit actions, the continuous-time nature and stochastic behavior of IMCs are important ingredients for us when we model distributed settings using *distributed IMCs*.

Definition 10.6 *An* IMC *(module) is a tuple* $(S, \Sigma, \hookrightarrow, \rightsquigarrow, s^{in})$ *where*

- S *is a finite set of states with an initial state* s^{in}*,*
- Σ *is a finite set of actions,*
- $\hookrightarrow \subseteq S \times \Sigma \times S$ *is the* action *transition relation,*
- $\rightsquigarrow \subseteq S \times \mathbb{Q}_{>0} \times S$ *is the finite* delay *transition relation.*

We write $s \overset{a}{\hookrightarrow} s'$ when $(s, a, s') \in \hookrightarrow$ and $s \overset{\lambda}{\rightsquigarrow} s'$ when $(s, \lambda, s') \in \rightsquigarrow$ (λ being the *rate* of the transition). We say that action a is *available* in s if $s \overset{a}{\hookrightarrow} s'$ for some s'.

Definition 10.7 *A* distributed IMC *is a tuple*

$$\mathcal{G} = ((S_i, \Sigma_i, \hookrightarrow_i, \rightsquigarrow_i, s_i^{in}))_{1 \leq i \leq n}$$

of modules for players $\Pi = \{1, ..., n\}$*. Furthermore, by* $\Sigma = \bigcup_i \Sigma_i$ *we denote the set of all actions, and by* $S = S_1 \times ... \times S_n$ *the set of (global) states.*

Intuitively, a distributed IMC moves in continuous-time from a (global) state to a (global) state using transitions with labels from $L = \Sigma \cup \Pi$:

- An action transition with label $a \in \Sigma$ corresponds to *synchronous* communication of all players in $\text{Move}(a) = \{j \in \Pi \mid a \in \Sigma_j\}$ and can only be taken when it is *enabled*, i.e. when all these players choose their local transitions with action a at the same time. It is called a *synchronization* action if $|\text{Move}(a)| \geq 2$ and a *private* action if $|\text{Move}(a)| = 1$.

- A delay transition of any player $j \in \Pi$ is taken *independently* by the player after a random delay, i.e. the set of players that synchronize over label j is $\mathrm{Move}(j) = \{j\}$.

Formally, the (local) *choices of player* j range over $\mathcal{C}_j = \hookrightarrow_j \cup \{\bot\}$. When in (local) state s, the player may pick only a choice *available in* s. That is, either an action transition of the form $s \overset{a}{\hookrightarrow} s'$ or $\bot$ if there is no such action transition. We define *global choices* as $\mathcal{C} = \mathcal{C}_1 \times \cdots \times \mathcal{C}_n$. A global choice c induces the set $\mathrm{En}(c) = \{a \in \Sigma \mid \forall j \in \mathrm{Move}(a) : c(j) = (\cdot, a, \cdot)\}$ of actions *enabled* in c.

To avoid that time stops by taking infinitely many action steps in zero time, we pose a standard assumption prohibiting cycles [HKK13; HJ08; Guc+12; Kat+11; KKN09]: we require that for every action $a \in \Sigma$ there is a player $j \in \mathrm{Move}(a)$ such that the labelled transition system $(S_j, \hookrightarrow_j)$ does not have any cycle involving action a.

The behaviour of a distributed IMC is given by a *play* which is an infinite sequence

$$\rho = s_0 c_0 \overset{a_1,t_1}{\to} s_1 c_1 \overset{a_2,t_2}{\to} s_2 c_2 \cdots$$

where each $s_i \in S$ is the state after i moves, $c_i \in \mathcal{C}$ is the choice of the players in the state s_i, and $a_{i+1} \in L$ and $t_{i+1} \in \mathbb{R}_{\geq 0}$ are the label and the absolute time of the next transition taken. By Play we denote the set of all plays. Which play is taken depends on the *strategies* of the players, on the *scheduler* which resolves interleaving of communication whenever multiple actions are enabled, and on the rules (involving randomness) given later.

10.3.1 Schedulers and strategies

First we define strategies and schedulers basing their decision on the current local and global history, respectively. A *(global) history* is a finite prefix

$$h = s_0 c_0 \overset{a_1,t_1}{\to} \cdots \overset{a_i,t_i}{\to} s_i$$

of a play ending with a state. For given h, we get the *local history* of player j as

$$\pi_j(h) = s'_0(j) c'_0(j) \overset{a'_1,t'_1}{\to} \cdots \overset{a'_\ell,t'_\ell}{\to} s'_\ell(j)$$

where $s'_0 c'_0 \overset{a'_1,t'_1}{\to} \cdots \overset{a'_\ell,t'_\ell}{\to} s'_\ell$ is the subsequence of h omitting all steps not visible for player j, i.e. all $\overset{a_m,t_m}{\to} s_m c_m$ with $j \notin \mathrm{Move}(a_m)$. The set of all global histories is denoted by Hist; the set of local histories of player j by Hist_j.

Example 10.8 *Consider again Example 10.3. Let* App *be controlled by player 1 and* Att *by player 2. For the following history we get corresponding local histories*

$$h = (c_0, \bar{c}_1)(\bot, \mathrm{login}) \overset{1,0.42}{\to} (t_1, \bar{c}_1)(\mathrm{login}, \mathrm{login}) \overset{\mathrm{login},0.42}{\to} (t_2, \bar{c}_2),$$
$$\pi_1(h) = c_0 \bot \overset{1,0.42}{\to} t_1 \mathrm{login} \overset{\mathrm{login},0.42}{\to} t_2, \qquad \pi_2(h) = \bar{c}_1 \mathrm{login} \overset{\mathrm{login},0.42}{\to} \bar{c}_2$$

Note that the attacker can neither observe the Markovian transition nor the local state of the App. *The* App *cannot observe the local state of the attacker either, but it can be deduced from the local history of the* App.

A *strategy* for player j is a measurable function $\sigma : \mathrm{Hist}_j \to \Delta(\mathcal{C}_j)$ that assigns to any local history h of player j a probability distribution over choices available in the last state of h. We say that a strategy σ for player j is *pure* if for all h we have $\sigma(h)(c) = 1$ for some c; and *memoryless* if for all h and h' with equal last local state we have $\sigma(h) = \sigma(h')$.

A *scheduler* is a measurable function $\delta : \mathrm{Hist} \times \mathcal{C} \to \Delta(\Sigma) \cup \{\bot\}$ that assigns to any global history h and global choice c a probability distribution over actions enabled in c; or a special symbol $\bot$ again denoting that no action is enabled.

Example 10.9 *The available local choices in* $(t_2, \bar{c}_2)$*, the last state of* h *from above, are* $\{(t_2, \mathit{lookup}, t_3), (t_2, \mathit{logout}, t_1)\}$ *for* App *and solely* $\{(\bar{c}_2, \mathit{lookup}, \bar{c}_3)\}$ *for* Att. *Let the strategy of* App *select either choice with equal probability. If* $(t_2, \mathit{lookup}, t_3)$ *is chosen, lookup is enabled and must be picked by the scheduler* σ*. If* $(t_2, \mathit{logout}, t_1)$ *is chosen, no action is enabled and* δ *must pick* $\bot$*, waiting for a delay transition.*

10.3.2 Probability of plays

Let us fix a strategy profile $\sigma = (\sigma_1, \ldots, \sigma_n)$ for individual players, and a scheduler δ. The play starts in the initial state $s_0 = (s_1^{in}, \ldots, s_n^{in})$ and inductively evolves as follows. Let the current history be $h = s_0 c_0 \overset{a_1,t_1}{\to} \cdots \overset{a_i,t_i}{\to} s_i$.

- For the next choice c_i, only players $P_i = \mathrm{Move}(a_i)$ involved in the last transition freely choose (we assume $P_0 = \Pi$). Hence, independently for every $j \in P_i$, the choice $c_i(j)$ is taken randomly according to $\sigma_j(\pi_j(h))$. All remaining players $j \notin P_i$ stick to the previous choice $c_i(j) = c_{i-1}(j)$ as for them, no observable event happened.

- After fixing c_i, there are two types of transitions:

  1. If $\mathrm{En}(c_i) \neq \emptyset$, the next synchronization action $a_{i+1} \in \mathrm{En}(c_i)$ is chosen randomly according to $\delta(h, c_i)$ and taken immediately at time $t_{i+1} = t_i$. The next state s_{i+1} satisfies for every $j \in \Pi$:

$$s_{i+1}(j) = \begin{cases} \mathrm{target}(c_i(j)) & \text{if } j \in \mathrm{Move}(a_{i+1}), \\ s_i(j) & \text{if } j \notin \mathrm{Move}(a_{i+1}). \end{cases}$$

     where $\mathrm{target}(c_i(j))$ denotes the target of the transition chosen by player j. In other words, players involved in synchronization move according to their choice, the remaining players stay in their previous states.

2. If $\mathrm{En}(c_i) = \emptyset$, a local delay transition is taken after a random delay, chosen as follows. Each delay transition $s_i(j) \overset{\lambda}{\rightsquigarrow} \cdot$ outgoing from the current local state of any player j is randomly assigned a real-valued delay according to the exponential distribution with rate λ. This results in a collection of real numbers. The transition $s_i(\ell) \overset{\lambda}{\rightsquigarrow} s$ with the minimum delay d in this collection is taken. Hence, $a_{i+1} = \ell$ (denoting that player ℓ moves), $t_{i+1} = t_i + d$, and the next state s_{i+1} satisfies for every $j \in \Pi$:

$$s_{i+1}(j) = \begin{cases} s & \text{if } j \in \mathrm{Move}(a_{i+1}) = \{\ell\}, \\ s_i(j) & \text{if } j \notin \mathrm{Move}(a_{i+1}). \end{cases}$$

All these rules induce a probability measure $\Pr^{\sigma,\delta}$ over the set of all plays by a standard cylinder construction. For details, see Appendix 11.

10.3.3 Distributed synthesis problems

We study the following two fundamental reachability problems for distributed IMCs. Let $\mathcal{G}$ be a distributed IMC, $T \subseteq S$ be a target set of states, and p be a rational number in $[0, 1]$. Denoting by $\diamond T$ the set of plays ρ that reach a state in T and stay there for a non-zero amount of time, we focus on:

Existence Does there exist a strategy profile σ s.t. for all schedulers δ,

$$\Pr^{\sigma,\delta}(\diamond T) \geq p \,?$$

Value Can the value p be arbitrarily approached, i.e. do we have

$$\sup_{\sigma} \inf_{\delta} \Pr^{\sigma,\delta}(\diamond T) \geq p \,?$$

We refer to the general problem with $p \in [0, 1]$ as *quantitative*. When we restrict to $p = 1$, we call the problem *qualitative*.

Note that the existence and value problems are related in the following way. If there exists a strategy σ' such that for all schedulers δ' we have

$$\Pr^{\sigma',\delta'}(\diamond T) \geq p$$

then we also have

$$\sup_{\sigma} \inf_{\delta} \Pr^{\sigma,\delta}(\diamond T) \geq p$$

An implication does not hold in the opposite direction though. We call the expression $\sup_\sigma \inf_\delta \Pr^{\sigma,\delta}(\diamond T)$ the *value* of a distributed IMC with target T. In some cases there might not be any strategy that ensures the value of the distributed IMC no matter how the scheduler behaves. But, there are strategies that can ensure getting arbitrarily close to the value of the distributed IMC. This phenomenon also occurs in concurrent games. For further details see [AHK98].

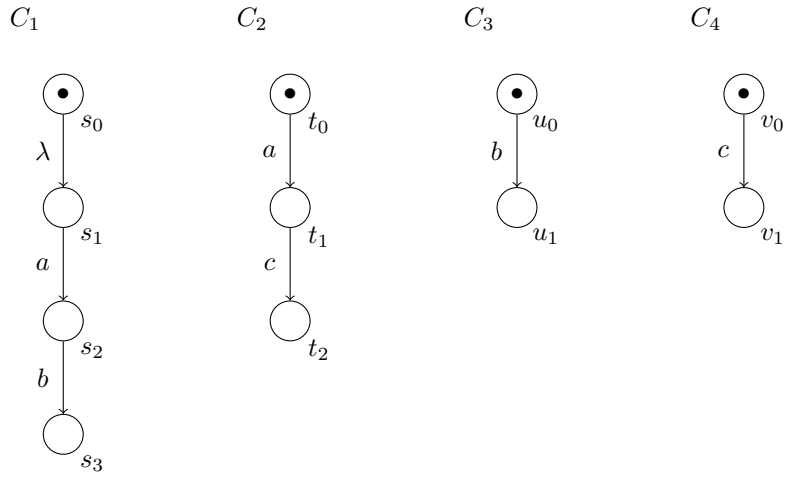


Figure 10.4: Distributed IMC where the scheduler can affect the order of transitions.

10.4 Schedulers are not that powerful

The task of a scheduler is to choose among concurrently enabled transitions, thereby resolving the non-determinism conceptually caused by interleaving. In this section, we address the impact of the decisions of the scheduler. We show that despite having the ability to affect the order in which transitions are taken in the *global* play, the scheduler cannot affect what every player observes *locally*. Thus, the scheduler affects neither the choices of any player nor what synchronization occurs. As a result, for winning objectives that are closed under *local observation equivalence*, the scheduler cannot affect the probability of winning.

Example 10.10 *Consider the distributed IMC in Figure 10.4. After the delay transition is taken in C_1 and there is synchronization on action a, the scheduler can choose whether there will be synchronization on b or c first. However, it can only affect the interleaving, not any of the local plays.*

For a play $\rho = s_0 c_0 \overset{a_1,t_1}{\to} s_1 c_1 \overset{a_2,t_2}{\to} s_2 c_2 \cdots$ we define the local play $\pi_j(\rho)$ of player j analogously to local histories. We define *local observation* equivalence $\sim$ over all plays ρ, ρ' by setting $\rho \sim \rho'$ if $\pi_j(\rho) = \pi_j(\rho')$ for all $j \in \Pi$. Let us stress that two local observation equivalent plays have exactly the same action and delay transitions happening at the same moments of time; only the order of action transitions happening at the same time can differ. Finally, we say that a set E of plays is *closed under local observation equivalence* if for any $\rho \in E$ and any ρ' such that $\rho \sim \rho'$ we have $\rho' \in E$.

It is now possible to show that the scheduler cannot affect the probability of events closed under local observation equivalence.

Theorem 10.11 *Let E be a measurable set of plays closed under local observation equivalence. For any strategy profile σ and schedulers δ and δ' we have*

$$\Pr^{\sigma,\delta}(E) = \Pr^{\sigma,\delta'}(E).$$

The proof of Theorem 10.11 is a bit technical and therefore moved to the appendix, it can be found in Appendix 11.

The proof can be divided into three main parts. The overall idea of the three parts are

1. showing that for pure strategy profiles, the schedulers cannot affect anything but the interleaving in a maximal 0-duration sequence of actions;

2. showing that 1. can be lifted to all events E closed under observation equivalence. This is done in two steps. First, it is shown to be the case for all interleaving abstract cylinders by induction using 1. Second, standard results from measure theory imply the result for all measurable sets of plays closed under local observation equivalence;

3. extending 2. from pure strategies to arbitrary strategies can be done reusing ideas from classical game theory on mixed and behavioural strategies in extensive-form games [Kuh53].

As a result, for the rest of the chapter we write $\Pr^{\sigma}(E)$ instead of $\Pr^{\sigma,\delta}(E)$ since the scheduler cannot affect the probability of the events we consider. Indeed, the reachability objectives defined in the previous section are closed under local observation equivalence.

10.5 Undecidability Results

In this section, we put distributed IMCs into context of other partial observation models. As a result, we show that reachability quickly gets undecidable:

Theorem 10.12 *For distributed IMCs we have that*

1. *the qualitative value, quantitative value, and quantitative existence problems are undecidable with $n \geq 2$ players; and*

2. *the qualitative existence problem is* ExpTime*-hard with $n = 2$ players and undecidable with $n \geq 3$ players.*

This theorem is obtained from the following two fundamental results.

	POMDPs	DEC-POMDPs	Distributed IMCs
Qual. Existence	Dec. [BGB12]	Und. ≥ 2 players	Und. ≥ 3 players
Value	Und. [GO10]	Und. ≥ 1 player [GO10]	Und. ≥ 2 players
Quant. Existence	Und. [Paz71]	Und. ≥ 1 player [Paz71]	Und. ≥ 2 players
Value	Und. [BMT77]	Und. ≥ 1 player [BMT77]	Und. ≥ 2 players

Table 10.5: Undecidability results for reachability. Unreferenced results are shown in this chapter.

- First, we show that distributed IMCs are not only more expressive (w.r.t. reachability) than POMDPs but also more expressive than DEC-POMDPs. We show it by reducing reachability in DEC-POMDPs with n players to reachability in distributed IMCs with $n+1$ players (Proposition 10.13).
- Second, we provide a novel (and somewhat surprising) result for DEC-POMDPs. We show that the qualitative existence problem for DEC-POMDPs is undecidable already for 2 players (Theorem 10.14).

Combining these two results and known results about POMDPs [Paz71; BMT77; GO10] the theorem follows. For an overview, see Table 10.5.

10.5.1 Reduction from DEC-POMDP

First we present the reduction from a DEC-POMDP $\mathcal{P}$ to a distributed IMC $\mathcal{G}$. In this subsection, we write $\Pr^{\sigma}_{\mathcal{P}}$ or $\Pr^{\sigma}_{\mathcal{G}}$ instead of $\Pr^{\sigma}$ to distinguish between the probability measure in the DEC-POMDP from the probability measure in the distributed IMC.

Proposition 10.13 *For a DEC-POMDP $\mathcal{P}$ with n players and a target set T of states of $\mathcal{P}$ we can construct in polynomial time a distributed IMC $\mathcal{G}$ with $n+1$ players and a target set T' of global states in $\mathcal{G}$ where:*

$$\exists \sigma : \Pr^{\sigma}_{\mathcal{G}}(\diamond T) = p \iff \exists \sigma' : \Pr^{\sigma'}_{\mathcal{P}}(\diamond T') = p.$$

Proof. Let us fix n and $\mathcal{P} = (S, \Pi, (\Sigma_i, \mathcal{O}_i)_{1 \leq i \leq n}, P, O, s^{in})$ where $\Pi = \{1, ..., n\}$. Further, let $\Sigma_i = \{a_{i1}, ..., a_{im_i}\}$ and $\mathcal{O}_i = \{o_{i1}, ..., o_{i\ell_i}\}$ for player $i \in \Pi$. The distributed IMC $\mathcal{G}$ has $n+1$ modules, one module for each player in $\mathcal{P}$ and the *main* module responsible for their synchronization. Intuitively,

- the module of every player i stores the last local observation in its state space. Every step of $\mathcal{P}$ is modelled as follows: The player *outputs* to the main module the action it chooses and then *inputs* from the main module the next observation.

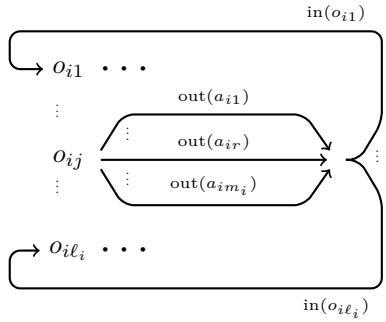


Figure 10.6: Module for player i.

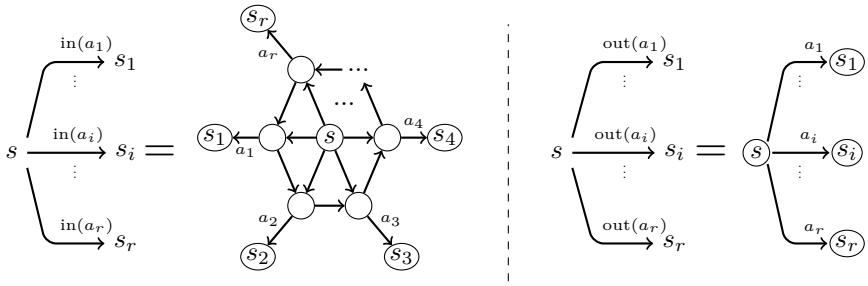


Figure 10.7: Input encoding to the left and output encoding to the right. In the input encoding, every transition not labelled with an action is a delay transition. All these delay transitions have the same rate λ.

- The main module stores the global state in its state space. Every step of $\mathcal{P}$ corresponds to the following: The main module *inputs* the actions of all players one by one, then it randomly picks the new state and new observations according to the rules of $\mathcal{P}$ based on the actions collected. The observations are lastly *output* to all players, again one by one.

We construct the distributed IMC so that only the outputting player chooses what action to output whereas the inputting player accepts whatever comes. The construction of modules for player i is illustrated in Figure 10.6 along with constructions for input and output in Figure 10.7.

Outputting an action $a \in \{a_1, \ldots, a_r\}$ in a state s is simple - this is modelled in s by standard action transitions for all these actions.

The interesting part is how an action from the set $\{a_1, \ldots, a_r\}$ is input in a state s. Instead of waiting in s, the player travels by delay transitions in a round-robin fashion through a cycle of r states, where in the i-th state, only the action a_i is available. Thus, the player has no influence and must input the action that comes. By this

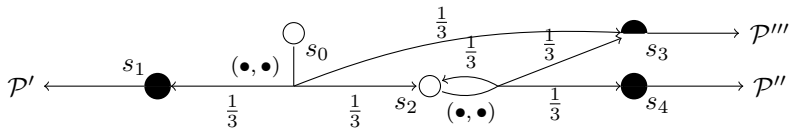


Figure 10.8: Overall structure of $\mathcal{P}$ without details of $\mathcal{P}', \mathcal{P}''$ and $\mathcal{P}'''$. Initially, with probability $\frac{1}{3}$ each the play either goes to s_1, s_2 or s_3. If the play goes to s_2 then with probability $\frac{1}{3}$ each the play either goes back to s_2, goes to s_3 or goes to s_4. From s_1 the play continues in $\mathcal{P}'$, from s_3 the play continues in $\mathcal{P}'''$ and from s_4 the play continues in $\mathcal{P}''$.

construction, the main module has at most one action transition in every state such that the player cannot influence anything; other modules get no insight by observing time and thus the players have the same power as in the DEC-POMDP. □

10.5.2 Undecidability of qualitative existence in DEC-POMDP

Next, we show that the qualitative existence problem for DEC-POMDPs even with $n \geq 2$ players is undecidable. The proof has similarities with ideas from [BK10] where it is shown that deciding existence of winning strategies in concurrent games with 3 players, partial observation and safety objectives is undecidable. Using the randomness of DEC-POMDPs we show undecidability of the qualitative existence problem for reachability in 2-player DEC-POMDPs.

Theorem 10.14 *It is undecidable whether for a DEC-POMDP $\mathcal{P}$ with $n \geq 2$ players and a set T of target states in $\mathcal{P}$ if there exists a strategy profile σ such that* $\Pr^{\sigma}_{\mathcal{P}}(\diamond T) = 1$.

Proof sketch. We do a reduction from the non-halting problem of a deterministic Turing machine M that starts with a blank input tape. From M we construct a DEC-POMDP $\mathcal{P}$ with two players $\Pi = \{1, 2\}$ such that M does not halt if and only if players 1 and 2 have strategies $\sigma = (\sigma_1, \sigma_2)$ which ensure that the probability of reaching a target set T is 1. Figure 10.8 shows the overall structure of $\mathcal{P}$ without details of sub-modules $\mathcal{P}', \mathcal{P}''$ and $\mathcal{P}'''$.

Both players have two possible observations, black and white. We depict the observation of player 1 in the top-half and of player 2 in the bottom-half of every state. The play starts in s_0 and with probability 1, every player receives the black observation exactly once during the play. If the play goes to s_1 or s_4 the players will receive the observation at the same time and if the play goes to s_3 then player 2 will receive the observation in the step after player 1 does. The modules $\mathcal{P}', \mathcal{P}''$ and $\mathcal{P}'''$ are designed so that:

- In $\mathcal{P}'$, a target state is reached if and only if the sequence of actions played by both players encodes the initial configuration of M.

- In $\mathcal{P}''$, a target state is reached with probability 1 if and only if both players play the same infinite sequence of actions. Note that randomness is essential to build such a module.
- In $\mathcal{P}'''$, the target set is reached if and only if the sequences of actions played by player 1 and 2 encode two configurations C_1 and C_2 of M, respectively, such that C_1 is not an accepting configuration and C_2 is a successor configuration of C_1. This can be done since a finite automaton can be constructed that recognizes if one configuration is a successor of the other when reading both configurations at the same time [BK10]. Note that it is possible because such configurations can only differ by a constant amount (the control state, the tape head position and in symbols in cells near the tape head).

It can be shown by induction that if there are strategies σ_1, σ_2 that ensure reaching T with probability 1 then every σ_i has to play the encoding of the jth configuration of M when it receives the black observation in the jth step. Further, it can be shown that these strategies do ensure reaching T with probability 1 if M does not halt on the empty input tape and do not ensure reaching T with probability 1 if M halts. □

10.6 Decidability for non-urgent models

In this section, we turn our attention to a subclass of distributed IMCs, called *non-urgent*. In this subclass the players have the possibility to keep redeciding after they have committed to actions. They also have the possibility to wait and not commit to any action and just let time pass. This is done without timeouts, so while waiting to choose which action to commit to or while redeciding they can be sure that their current local state does not change. We show how such situations can be modelled in our framework.

In this subclass we obtain decidability for both the qualitative and quantitative value problems for 2 players.

Definition 10.15 *A distributed IMC* $\mathcal{G} = ((S_i, \Sigma_i, \hookrightarrow_i, \rightsquigarrow_i, s_{0i}))_{1 \leq i \leq n}$ *is* non-urgent *if for every* $1 \leq j \leq n$*:*

1. *Every* $s \in S_j$ *is of one of the following forms:*
   a) Synchronisation state *with at least one outgoing synchronization action transition and exactly one outgoing delay transition which is a self-loop.*
   b) Private state *with arbitrary outgoing delay transitions and private action transitions.*
2. *Player* j *has an action* $\varnothing_j \in \Sigma_j$ *enabled in every synchronization state from* S_j *that allows to "do nothing" and thus postpone the synchronization. To this end,* $\varnothing_j$ *is also in* Σ_k *for every other player* $k \neq j$ *but* $\varnothing_j$ *does not appear in* $\hookrightarrow_k$*. As a result,* j *does not take part in any synchronization while choosing* $\varnothing_j$*.*

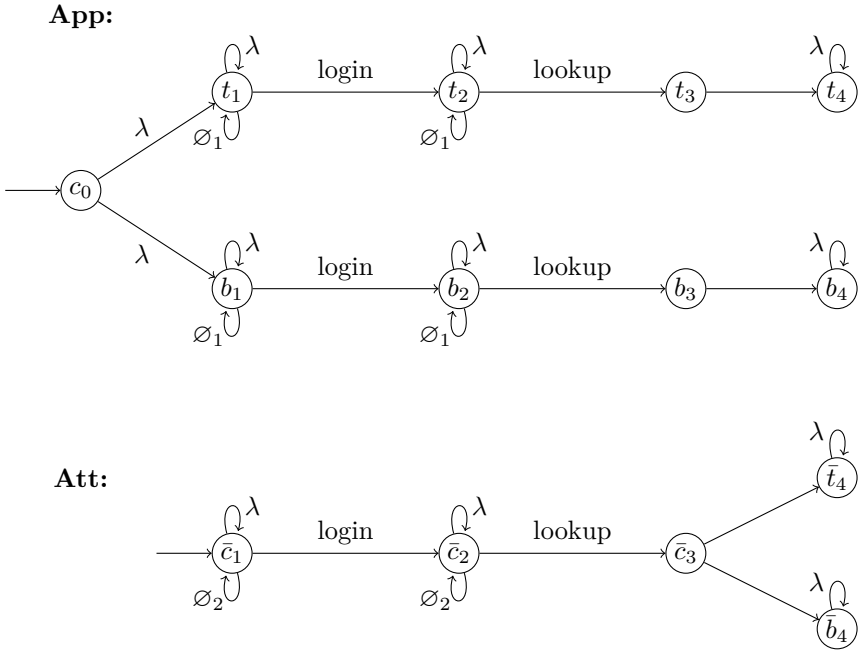


Figure 10.9: A non-urgent distributed IMC.

In a non-urgent distributed IMC, $s \in S$ is called a (global) *synchronization* state if it is the initial state or all $s(j)$ are synchronization states. We denote global synchronization states by S'. All other global states $S \setminus S'$ are called *private*.

Example 10.16 *Consider the non-urgent variant of Example 10.3 on the right. The "do nothing" actions are a natural concept; the only real modelling restriction is that one cannot model a communication time-out any more, the delay transitions from synchronization states need to be self-loops.*

Surprisingly, in this model, the secret can be leaked with probability 1 *as follows. As before, the players reach the states* $(t_2, \bar{c}_2)$ *or* $(b_2, \bar{c}_2)$ *with equal probability. Now, the* App *player can arbitrarily postpone the* lookup *by committing to action* $\varnothing_1$*. Whenever the delay self-loop is taken, the player can re-decide to perform* lookup*. Since the self-loop is taken repetitively, the* App *player is flexible in choosing the timing of* lookup*. Thus, leaking the secret is simple, e.g. by performing* lookup *in an odd second when in* t_2 *and in an even second when in* b_2*.*

For two players, we construct a general synchronization scheme that (highly probably) shows the players the current global state after each communication.

Theorem 10.17 *The quantitative value problem for 2-player non-urgent distributed IMCs where the target set consists only of synchronization states is in* ExpTime.

Being a special case, also the qualitative value problem is decidable. In essence, the problem becomes decidable because in the synchronization states, the players can effectively exchange arbitrary information. This resembles the setting of [Gen+13]. The insight that observing global time provides an additional synchronization mechanism is not novel in itself, but it is obviously burdensome to formally capture in time-abstract models of asynchronous communication, and thus usually not considered. For distributed IMC, it still is non-trivial to develop; here it hinges on the non-urgency assumption. The results of [Gen+13] also indicate that for three or more players, additional constraints on the topology may be needed to obtain decidability.

In the rest of the section we prove Theorem 10.17, fixing a 2-player non-urgent distributed IMC $\mathcal{G} = ((S_i, \Sigma_i, \hookrightarrow_i, \rightsquigarrow_i, s_{0i}))_{1 \leq i \leq 2}$, a probability $p \in [0, 1]$, and a set $T \subseteq S'$ of target states. We present the algorithm based on a reduction to a discrete-time Markov decision process and then discuss its correctness.

10.6.1 The algorithm

We provide an algorithm based on a reduction to the value problem for a discrete-time MDP. The idea is that by using a certain communication scheme explained below, the players can use the knowledge of the global clock and the non-urgent nature of the synchronization states to approximate having full observation.

The algorithm takes a distributed IMC $\mathcal{G}$ as input and from this creates a discrete-time MDP $\mathcal{M}_\mathcal{G}$ which, informally, assumes full observability of the players in the game. As such, they can be treated as the one player in the MDP.

Formally, we construct $\mathcal{M}_\mathcal{G} = (S', \Sigma, P, s_0)$ where

- $S' \subseteq S$ is the set of global synchronization states;
- $\Sigma = (\mathcal{C} \times \chi_1 \times \chi_2) \cup \{\bot\}$ where χ_j is the set of pure memoryless strategies of player j in $\mathcal{G}$ that choose $\varnothing_j$ in every synchronization state;
- For an arbitrary state (s_1, s_2), we define the transition function as follows:
  - For any $(c, \sigma_1, \sigma_2) \in \Sigma$, the transition $P((s_1, s_2), (c, \sigma_1, \sigma_2))$ is defined if c is available in (s_1, s_2) and the players agree in c on some action a, i.e. $\mathrm{En}(c) = \{a\}$. If defined, the distribution $P((s_1, s_2), (c, \sigma_1, \sigma_2))$ assigns to any successor state (s_1', s_2') the probability that in $\mathcal{G}$ the state (s_1', s_2') is reached from (s_1, s_2) via states in $S \setminus S'$ by choosing c and then using the pure memoryless strategy profile (σ_1, σ_2).
  - To avoid deadlocks, the transition $P((s_1, s_2), \bot)$ is defined iff no other transition is defined in (s_1, s_2) and it is a self-loop, i.e. it assigns probability 1 to (s_1, s_2).

The MDP $\mathcal{M}_\mathcal{G}$ has size exponential in $|\mathcal{G}|$ due to the number of pure memoryless strategies in χ_1 and χ_2. Note that all target states T are included in S'. Slightly abusing notation, let $\diamond T$ denote the set of plays in $\mathcal{M}_\mathcal{G}$ that reach the set T. From standard results on MDPs [Put09], there exists an optimal pure memoryless strategy π^* in $\mathcal{M}_\mathcal{G}$, i.e. satisfying $\Pr^{\pi^*}(\diamond T) = \sup_\pi \Pr^\pi(\diamond T)$. Furthermore, such a strategy π^* and the value $v = \Pr^{\pi^*}(\diamond T)$ can be computed in time polynomial in $|\mathcal{M}_\mathcal{G}|$. Finally, the algorithm returns TRUE if $v \geq p$ and FALSE, otherwise.

10.6.2 Correctness of the algorithm

Below we give a proof sketch for the correctness of the algorithm above. This boils down to showing that the value of $\mathcal{G}$ is equal to the value of the MDP.

Proposition 10.18 *The value of $\mathcal{G}$ is equal to the value of $\mathcal{M}_\mathcal{G}$, i.e.*

$$\sup_\sigma \Pr^\sigma(\diamond T) \quad = \quad \sup_\pi \Pr^\pi(\diamond T).$$

Proof sketch. As regards the $\leq$ inequality, it suffices to show that any strategy profile σ can be mimicked by some strategy π. This is simple as the player in $\mathcal{M}_\mathcal{G}$ has always knowledge of the global state. Much more interesting is the $\geq$ inequality. We argue that for any strategy π there is a sequence of strategy profiles $\sigma^1, \sigma^2, \ldots$ in $\mathcal{G}$ such that

$$\lim_{i\to\infty} \Pr^{\sigma^i}(\diamond T) = \Pr^\pi(\diamond T).$$

The crucial idea is that a strategy profile communicates correctly (with high probability) the current global state in a synchronization state by *delaying* as follows. The time is divided into phases, each of $|S_1'| \cdot 2|S_2'|$ slots (where S_i' are the synchronization states of player i). We depict a phase by the table in Figure 10.10 where the time flows from top to bottom and from left to right (as reading a book).

Players 1 and 2 try to synchronize in the row and column, respectively, corresponding to their current states (3 and 1 respectively as marked by the circles) and in each slot take the optimal choice $c_i(s_1, s_2)$ given the current global state is (s_1, s_2); in the remaining slots they choose to do nothing. Since the players can change their choice only at random moments of time when the self-loop is triggered, their synchronizing choice can always stretches a bit into the successive silent slot (in a $\neg$sync column).

The more we increase the duration of each slot, the lower is the chance that a synchronization choice of a player stretches to the next *synchronization* slot. Thus, the lower is the chance of an erroneous synchronization.

We define the duration of the slot to increase with i and also along the play so that for any fixed i the probability of at least one erroneous synchronization is bounded by $\kappa_i < 1$ and for $i \to \infty$, we have $\kappa_i \to 0$. □

Note that in Example 10.16 the App and Att players can use a communication scheme just like suggested in the proof sketch above. Doing this the players can ensure

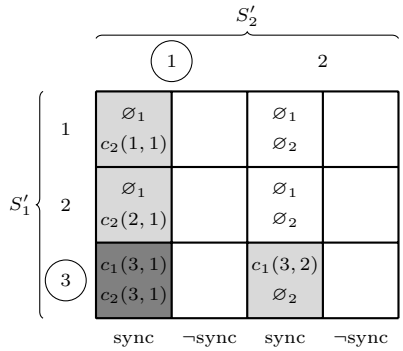


Figure 10.10: A communication phase with 12 time slots is illustrated in the figure. Player 1 is in state 3 and player 2 is in state 1 which is illustrated by the circles. This means that player 1 tries to synchronize in sync time slots in the bottom row, whereas player 2 tries to synchronize in time slots in the left-most column. Since a player does not know the state of the other player, the choice made in each time slot depends on the particular time slot, not only on the state of the player himself. For instance, player 2 is ready to synchronize on three different actions in the three different time slots in the left-most column, each corresponding to the optimal choice if player 1 is in state 1, 2 or 3 respectively. As the players do not synchronize with probability 1 during a phase, the players continue a sequence of phases like this until synchronization occurs. This is because the players can only redecide when self-loops are triggered and this is done in a stochastic fashion. Thus, a player might not get a chance to redecide within a given time slot.

winning with probability arbitrarily close to 1 by choosing appropriate durations of the time slots. Thus, they can leak the secret by using the global time.

10.7 Summary

This chapter has introduced a foundational framework for modelling and synthesizing distributed controllers interacting in continuous time via handshake communication. The continuous time nature of the model induces that the interleaving scheduler has in fact very little power.

Distributed IMCs can be considered as an attractive base model especially in the context of information flow and other security-related studies. This is because in contrast to the discrete time setting, the power of the interleaving scheduler is no matter of debate, it can leak no essential information.

We studied cooperative reachability problems in distributed IMCs for which we presented a number of undecidability results. In addition to undecidability results for distributed IMCs, we have sharpened the undecidability border in decentralized POMDPs by showing undecidability of the qualitative existence problem already for two players. That is, undecidability of deciding whether there exist strategies for the two players that ensure reaching the set of target states with probability 1.

For non-urgent models we have established decidability of both quantitative and qualitative problems for the two-player case by using a particular synchronization scheme between the players. Whether this technique can be extended to more players and whether the exponential-time complexity is optimal are left open.

CHAPTER 11

Conclusion

We now give a summary of the main results presented in this thesis, evaluate on the goals and give some pointers to future work.

Highlights

In Chapter 4 we introduced the novel concept of winning cores in parity games and showed interesting properties of these. They provide new knowledge about the structure of winning regions and strategies in such games. Further, we presented a polynomial-time under-approximation algorithm based on winning cores. Experimental results on benchmark games and random games are very promising with respect to quality of approximation and efficiency compared to existing approaches. In most cases the algorithm performed better than the current state of the art.

In Chapter 5 we obtained precise computational complexities of model-checking for quantitative extensions of CTL*, ATL and ATL* in one-counter games. Model-checking games were used to obtain optimal complexity in each of the cases considered. Tight lower bounds were found for all cases considered. We also found the precise complexity of solving one-counter games with LTL objecties. The complexity results ranged from PSPACE-completeness to 2EXPSPACE-completeness.

In Chapter 6 we extended the well-known technique of symmetry reduction to be able to handle turn-based games with finite branching. It was shown how this technique is applicable for reducing the state space both when solving parity games and when model-checking ATL*.

In Chapter 7 the satisfiability problem for various flat fragments of ATL* was considered. We showed that for all the logics considered, satisfiability of formulas with a nesting depth of 2 of strategic quantifiers is already as hard as satisfiability for the full logic. Precise complexity results were shown for all fragments considered. The complexity results ranged from NP-complete to PSPACE-complete with several problems being complete for the class Σ_3^P in the polynomial hierarchy.

In Chapter 9 we introduced the new game-based formalism *symmetric game networks* for modelling distributed and reactive systems in settings with several identical players. We studied the computational complexity of the existence of symmetric Nash equilibria in these models and showed several undecidability results. Decidability results were obtained when restricting to finite-memory Nash equilibria.

In Chapter 10 we introduced the formalism of *distributed interactive Markov chains* to model distributed and reactive systems using continuous time and prob-

ability distributions over the length of delays. We first showed that the interleaving scheduler has no real power in this model. Next, we showed undecidability of the existence of strategies to ensure reaching a target with probability 1 already for 3 players. As part of the proof we showed undecidability of the same question already for 2 players in the well-established model of decentralized partially-observable Markov decision processes (DEC-POMDPs). We finally obtained decidability of the value problem in the restricted class of non-urgent models with 2 players.

Evaluation

In Section 1.7 we stated three main categories of goals. Briefly, these categories are:

1. Improving the efficiency of existing techniques for game-based verification and synthesis.
2. Deciding the computational complexity of foundational problems in model-checking, satisfiability and synthesis.
3. Developing new formalisms for game-based modelling of distributed and reactive systems.

Our results on parity games in Chapter 4 fall in the first category in two different ways. First, we have provided a new polynomial-time underapproximation algorithm with very promising performance compared to the state of the art algorithms. Second, with winning cores we have provided new structural insights into the workings of parity games. This has opened up a new direction for attacking the problem of deciding whether solving parity games (and equivalently, model-checking μ-calculus) can be done in polynomial time. This problem has currently been unsolved for more than 30 years [Koz83; EL86].

The extension of the symmetry reduction technique in Chapter 6 makes it possible to solve various questions about games more efficiently by reducing the state space, thus falling in the first category as well. It has yet to be seen how large an impact this will have in practice, but similar techniques in model-checking of transition systems [ID96; Cla+96; Cla+98], real-time systems [Hen+03] and probabilistic systems [KNP06] are promising.

The complexity results obtained in Chapter 5 and Chapter 7 falls in the second category of goals. Here, we determined the precise computational complexity for several model-checking and satisfiability problems in ATL* and fragments. In particular, the results in Chapter 5 are, together with results from [MP15; CSW16], some of the first known decidability results for ATL and ATL* model-checking in infinite-state games. Another important result was the 2ExpSpace-completeness shown for one-counter games with LTL-objectives.

In Chapter 9 and 10 we also presented computational complexity results on basic questions in the new models introduced. However, there are still many open questions to consider for these models. We also showed in Chapter 10 that almost-sure

reachability in DEC-POMDPs is already undecidable for 2 players. This is currently the strongest undecidability result for this formalism.

Regarding the third goal we have introduced two new modelling formalisms for distributed settings in Chapter 9 and 10. The first extends existing formalisms by making it possible to explicitly model several players that are identical. It also makes it possible to reason about several players applying the same strategy. The second extends the continuous-time formalism of interactive Markov chains [Her02] to a game-based setting appropriate for synthesis of strategies. It is the first game-based formalism for modelling distributed settings with continuous time and probability distributions over length of delays.

Future directions and open problems

Some of the more interesting directions for future work that originates from our results are the following:

- To investigate whether the new concept of winning cores in parity games can provide a succesful approach for resolving the question of whether parity games can be solved in polynomial time.
- To apply the symmetry reduction technique in practical applications of game-based verification and synthesis. Recall that there are no guarantees on systems without symmetry, but that the technique has had a large impact in other areas of verification.
- To analyze the models introduced in Chapters 9 and 10 more in-depth. In particular, to search for applicable subclasses with decidable problems and a lower computational complexity than the problems considered in this thesis. Hopefully, these can be used in practice to synthesize real distributed systems that are correct-by-construction.

Bibliography

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009. ISBN: 978-0-521-42426-4.

[ACD93] Rajeev Alur, Costas Courcoubetis, and David L. Dill. "Model-Checking in Dense Real-time". In: *Inf. Comput.* 104.1 (1993), pages 2–34.

[ÅGJ07] Thomas Ågotnes, Valentin Goranko, and Wojciech Jamroga. "Alternating-time temporal logics with irrevocable strategies". In: *TARK*. 2007, pages 15–24.

[AHK02] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. "Alternating-time temporal logic". In: *J. ACM* 49.5 (2002), pages 672–713.

[AHK98] Luca de Alfaro, Thomas A. Henzinger, and Orna Kupferman. "Concurrent Reachability Games". In: *39th Annual Symposium on Foundations of Computer Science*. 1998, pages 564–575.

[Alu+98] Rajeev Alur, Thomas A. Henzinger, Orna Kupferman, and Moshe Y. Vardi. "Alternating Refinement Relations". In: *CONCUR*. 1998, pages 163–178.

[Ber+02] Daniel S. Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. "The Complexity of Decentralized Control of Markov Decision Processes". In: *Math. Oper. Res.* 27.4 (2002), pages 819–840.

[Ber+06] Dietmar Berwanger, Anuj Dawar, Paul Hunter, and Stephan Kreutzer. "DAG-Width and Parity Games". In: *23rd Annual Symposium on Theoretical Aspects of Computer*. 2006, pages 524–536.

[BG13] Nils Bulling and Valentin Goranko. "How to Be Both Rich and Happy: Combining Quantitative and Qualitative Strategic Reasoning about Multi-Player Games (Extended Abstract)". In: *SR*. 2013, pages 33–41.

[BGB12] Christel Baier, Marcus Größer, and Nathalie Bertrand. "Probabilistic ω-automata". In: *J. ACM* 59.1 (2012), pages 1–52.

[BGM14] Patricia Bouyer, Patrick Gardy, and Nicolas Markey. "Quantitative Verification of Weighted Kripke Structures". In: *ATVA*. 2014, pages 64–80.

[BHR84] Stephen D. Brookes, C. A. R. Hoare, and A. W. Roscoe. "A Theory of Communicating Sequential Processes". In: *J. ACM* 31.3 (1984), pages 560–599.

[BJ13] Nils Bulling and Wojciech Jamroga. "Comparing variants of strategic ability: how uncertainty and memory influence general properties of games". English. In: *J. of AAMAS* (2013), pages 1–45. ISSN: 1387-2532.

[BJK10] Tomás Brázdil, Petr Jancar, and Antonín Kucera. "Reachability Games on Extended Vector Addition Systems with States". In: *ICALP (2)*. 2010, pages 478–489.

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008. ISBN: 978-0-262-02649-9.

[BK10] Dietmar Berwanger and Lukasz Kaiser. "Information Tracking in Games on Graphs". In: *Journal of Logic, Language and Information* 19.4 (2010), pages 395–412.

[Blo+09] Roderick Bloem, Krishnendu Chatterjee, Thomas A. Henzinger, and Barbara Jobstmann. "Better Quality in Synthesis through Quantitative Objectives". In: *Computer Aided Verification, 21st International Conference, CAV 2009, Grenoble, France, June 26 - July 2, 2009. Proceedings*. 2009, pages 140–156.

[Blo+12] Roderick Bloem, Barbara Jobstmann, Nir Piterman, Amir Pnueli, and Yaniv Sa'ar. "Synthesis of Reactive(1) designs". In: *J. Comput. Syst. Sci.* 78.3 (2012), pages 911–938.

[Blo+14] Roderick Bloem, Krishnendu Chatterjee, Karin Greimel, Thomas A. Henzinger, Georg Hofferek, Barbara Jobstmann, Bettina Könighofer, and Robert Könighofer. "Synthesizing robust systems". In: *Acta Inf.* 51.3-4 (2014), pages 193–220.

[BMP10] Laura Bozzelli, Aniello Murano, and Adriano Peron. "Pushdown module checking". In: *Formal Methods in System Design* 36.1 (2010), pages 65–95.

[BMS14] Patricia Bouyer, Nicolas Markey, and Daniel Stan. "Mixed Nash Equilibria in Concurrent Games". In: *Proceedings of the 34th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'14)*. December 2014, pages 351–363.

[BMT77] Alberto Bertoni, Giancarlo Mauri, and Mauro Torelli. "Some Recursive Unsolvable Problems Relating to Isolated Cutpoints in Probabilistic Automata". In: *ICALP*. 1977, pages 87–94.

[BMV14] Patricia Bouyer, Nicolas Markey, and Steen Vester. "Nash Equilibria in Symmetric Games with Partial Observation". In: *Proceedings 2nd International Workshop on Strategic Reasoning (SR)*. Electronic Proceedings in Theoretical Computer Science. Grenoble, France, 2014, pages 49–55.

[BMV16] Patricia Bouyer, Nicolas Markey, and Steen Vester. "Nash Equilibria in Symmetric Graph Games with Partial Observation". To appear in a special issue of *Information & Computation*. 2016.

[Bou+08] Patricia Bouyer, Ulrich Fahrenberg, Kim Guldstrand Larsen, Nicolas Markey, and Jirí Srba. "Infinite Runs in Weighted Timed Automata with Energy Constraints". In: *FORMATS*. 2008, pages 33–47.

[Bou+12] Patricia Bouyer, Romain Brenguier, Nicolas Markey, and Michael Ummels. "Concurrent games with ordered objectives". In: *Proc. 15th International Conference on Foundations of Software Science and Computation Structure (FoSSaCS'12)*. March 2012, pages 301–315.

[Bou+15] Patricia Bouyer, Romain Brenguier, Nicolas Markey, and Michael Ummels. "Pure Nash Equilibria in Concurrent Games". In: *Logical Methods in Computer Science* 11.2:9 (June 2015).

[Boz07] Laura Bozzelli. "Complexity results on branching-time pushdown model checking". In: *Theor. Comput. Sci.* 379.1-2 (2007), pages 286–297.

[Bri+09] Thomas Brihaye, Arnaud Da Costa, François Laroussinie, and Nicolas Markey. "ATL with Strategy Contexts and Bounded Memory". In: *Proc. of LFCS'2009, Springer LNCS 5407*. 2009, pages 92–106.

[Can+08] Ran Canetti, Ling Cheung, Dilsun Kirli Kaynar, Moses Liskov, Nancy A. Lynch, Olivier Pereira, and Roberto Segala. "Analyzing Security Protocols Using Time-Bounded Task-PIOAs". In: *Discrete Event Dynamic Systems* 18.1 (2008), pages 111–159.

[CE81] Edmund M. Clarke and E. Allen Emerson. "Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic". In: *Logic of Programs*. 1981, pages 52–71.

[CGP01] Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model checking*. MIT Press, 2001. ISBN: 978-0-262-03270-4.

[Cha+06] Krishnendu Chatterjee, Laurent Doyen, Thomas A. Henzinger, and Jean-François Raskin. "Algorithms for Omega-Regular Games with Imperfect Information". In: *CSL*. 2006, pages 287–302.

[Cha08] Krishnendu Chatterjee. "Linear Time Algorithm for Weak Parity Games". In: *CoRR* abs/0805.1391 (2008).

[Che+06] Ling Cheung, Nancy A. Lynch, Roberto Segala, and Frits W. Vaandrager. "Switched PIOA: Parallel composition via distributed scheduling". In: *Theor. Comput. Sci.* 365.1-2 (2006), pages 83–108.

[CHJ06] Krishnendu Chatterjee, Thomas A. Henzinger, and Marcin Jurdzinski. "Games with secure equilibria". In: *Theor. Comput. Sci.* 365.1-2 (2006), pages 67–82.

[CHP07] Krishnendu Chatterjee, Thomas A. Henzinger, and Nir Piterman. "Strategy Logic". In: *Proceedings of the 18th International Conference on Concurrency Theory (CONCUR'07)*. Edited by Luís Caires and Vasco T. Vasconcelos. Volume 4703. Lecture Notes in Computer Science. Springer-Verlag, September 2007, pages 59–73.

[CHP10] Krishnendu Chatterjee, Thomas A. Henzinger, and Nir Piterman. "Strategy Logic". In: *Information and Computation* 208.6 (June 2010), pages 677–693.

[Chu40] Alonzo Church. "A Formulation of the Simple Theory of Types". In: *J. Symb. Log.* 5.2 (1940), pages 56–68.

[Cla+96] Edmund M. Clarke, Somesh Jha, Reinhard Enders, and Thomas Filkorn. "Exploiting Symmetry in Temporal Logic Model Checking". In: *Formal Methods in System Design* 9.1/2 (1996), pages 77–104.

[Cla+98] Edmund M. Clarke, E. Allen Emerson, Somesh Jha, and A. Prasad Sistla. "Symmetry Reductions inModel Checking". In: *CAV*. 1998, pages 147–158.

[CMJ04] Krishnendu Chatterjee, Rupak Majumdar, and Marcin Jurdziński. "On Nash equilibria in stochastic games". In: *Proc. 18th International Workshop on Computer Science Logic (CSL'04)*. September 2004, pages 26–40.

[Coo71] Stephen A. Cook. "The Complexity of Theorem-Proving Procedures". In: *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*. 1971, pages 151–158.

[CSW16] Taolue Chen, Fu Song, and Zhilin Wu. "Global Model Checking on Pushdown Multi-Agent Systems". In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*. 2016, pages 2459–2465.

[DG09] Stéphane Demri and Régis Gascon. "The Effects of Bounding Syntactic Resources on Presburger LTL". In: *J. Log. Comput.* 19.6 (2009), pages 1541–1575.

[DKT12] Christoph Dittmann, Stephan Kreutzer, and Alexandru I. Tomescu. "Graph Operations on Parity Games and Polynomial-Time Algorithms". In: *CoRR* abs/1208.1640 (2012).

[DLM10] Arnaud Da Costa, François Laroussinie, and Nicolas Markey. "ATL with strategy contexts: Expressiveness and Model Checking". In: *Proceedings of the 30th Conferentce on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'10)*. Edited by Kamal Lodaya and Meena Mahajan. Volume 8. Leibniz International Proceedings in Informatics. Leibniz-Zentrum für Informatik, December 2010, pages 120–132.

[DM86] Partha Dasgupta and Eric Maskin. "The Existence of Equilibrium in Discontinuous Economic Games, 1: Theory". In: *The Review of Economic Studies* 53.1 (January 1986), pages 1–26.

[Dri03] Govert van Drimmelen. "Satisfiability in Alternating-Time Temporal Logic". In: *Proceedings of 18th IEEE Symposium on Logic in Computer Science (LICS)*. 2003, pages 208–217.

[DS02] Stéphane Demri and Ph. Schnoebelen. "The Complexity of Propositional Linear Temporal Logics in Simple Cases". In: *Inf. Comput.* 174.1 (2002), pages 84–103.

[EH86] E. Allen Emerson and Joseph Y. Halpern. ""Sometimes" and "Not Never" revisited: on branching versus linear time temporal logic". In: *J. ACM* 33.1 (1986), pages 151–178.

[EJ91] E. Allen Emerson and Charanjit S. Jutla. "Tree Automata, Mu-Calculus and Determinacy (Extended Abstract)". In: *FOCS*. 1991, pages 368–377.

[EJS01] E. Allen Emerson, Charanjit S. Jutla, and A. Prasad Sistla. "On model checking for the μ-calculus and its fragments". In: *Theor. Comput. Sci.* 258.1-2 (2001), pages 491–522.

[EL86] E. Allen Emerson and Chin-Laung Lei. "Efficient Model Checking in Fragments of the Propositional Mu-Calculus (Extended Abstract)". In: *Proceedings of the First Annual IEEE Symp. on Logic in Computer Science*. 1986, pages 267–278.

[EM79] A. Ehrenfeucht and J. Mycielski. "Positional Strategies for Mean Payoff Games". In: *International Journal of Game Theory* 8.2 (1979), pages 109–113.

[Eme08] E. Allen Emerson. "The Beginning of Model Checking: A Personal Perspective". In: *25 Years of Model Checking - History, Achievements, Perspectives*. 2008, pages 27–45.

[Eme90] E. Allen Emerson. "Temporal and Modal Logic". In: *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*. 1990, pages 995–1072.

[ES96] E. Allen Emerson and A. Prasad Sistla. "Symmetry and Model Checking". In: *Formal Methods in System Design* 9.1/2 (1996), pages 105–131.

[FL09] Oliver Friedmann and Martin Lange. "Solving Parity Games in Practice". In: *Proceedings of Automated Technology for Verification and Analysis, 7th International Symposium.* 2009, pages 182–196.

[FO14] Bernd Finkbeiner and Ernst-Rüdiger Olderog. "Petri Games: Synthesis of Distributed Systems with Causal Memory". In: *Proceedings Fifth International Symposium on Games, Automata, Logics and Formal Verification, GandALF 2014, Verona, Italy, September 10-12, 2014.* 2014, pages 217–230.

[Fri09] Oliver Friedmann. "An Exponential Lower Bound for the Parity Game Strategy Improvement Algorithm as We Know it". In: *Proceedings of the 24th Annual IEEE Symposium on Logic in Computer Science*. 2009, pages 145–156.

[Fri11] Oliver Friedmann. "Recursive algorithm for parity games requires exponential time". In: *RAIRO - Theor. Inf. and Applic.* 45.4 (2011), pages 449–457.

[FS05] Bernd Finkbeiner and Sven Schewe. “Uniform Distributed Synthesis”. In: *20th IEEE Symposium on Logic in Computer Science (LICS)*. 2005, pages 321–330.

[Gaj+15] Jakub Gajarský, Michael Lampis, Kazuhisa Makino, Valia Mitsou, and Sebastian Ordyniak. “Parameterized Algorithms for Parity Games”. In: *Mathematical Foundations of Computer Science 2015 - 40th International Symposium*. 2015, pages 336–347.

[GD06] V. Goranko and G. van Drimmelen. “Complete Axiomatization and Decidablity of Alternating-time temporal logic”. In: *Theor. Comp. Sci.* 353 (2006), pages 93–117.

[GDF14] Sergio Giro, Pedro R. D’Argenio, and Luis María Ferrer Fioriti. “Distributed probabilistic input/output automata: Expressiveness, (un)decidability and algorithms”. In: *Theor. Comput. Sci.* 538 (2014), pages 84–102.

[Gen+13] Blaise Genest, Hugo Gimbert, Anca Muscholl, and Igor Walukiewicz. “Asynchronous Games over Tree Architectures”. In: *ICALP*. Volume 7966. LNCS. Springer, 2013, pages 275–286.

[GL13] Stefan Göller and Markus Lohrey. “Branching-Time Model Checking of One-Counter Processes and Timed Automata”. In: *SIAM J. Comput.* 42.3 (2013), pages 884–923.

[GO10] Hugo Gimbert and Youssouf Oualhadj. “Probabilistic Automata on Finite Words: Decidable and Undecidable Problems”. In: *ICALP*. 2010, pages 527–538.

[Göl+10] Stefan Göller, Christoph Haase, Joël Ouaknine, and James Worrell. “Model Checking Succinct and Parametric One-Counter Automata”. In: *ICALP (2)*. 2010, pages 575–586.

[GS09] Valentin Goranko and Dmitry Shkatov. “Tableau-based decision procedures for logics of strategic ability in multiagent systems”. In: *ACM Trans. Comput. Log.* 11.1 (2009).

[GTW02] Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*. Volume 2500. Lecture Notes in Computer Science. Springer, 2002. ISBN: 3-540-00388-6.

[Guc+12] D. Guck, T. Han, J.-P. Katoen, and M.R. Neuhäußer. “Quantitative Timed Analysis of Interactive Markov Chains”. In: *NFM*. Volume 7226. LNCS. Springer, 2012, pages 8–23.

[GV14] Valentin Goranko and Steen Vester. “Optimal Decision Procedures for Satisfiability in Fragments of Alternating-time Temporal Logics”. In: *Advances in Modal Logic 10, invited and contributed papers from the tenth conference on Advances in Modal Logic (AiML)*. 2014, pages 234–253.

[GZ05] Hugo Gimbert and Wieslaw Zielonka. “Games Where You Can Play Optimally Without Any Memory”. In: *Concurrency Theory (CONCUR), 16th International Conference, Proceedings*. 2005, pages 428–442.

[Hal95] Joseph Y. Halpern. “The Effect of Bounding the Number of Primitive Propositions and the Depth of Nesting on the Complexity of Modal Logic”. In: *Artif. Intell.* 75.2 (1995), pages 361–372.

[Hen+03] Martijn Hendriks, Gerd Behrmann, Kim Guldstrand Larsen, Peter Niebert, and Frits W. Vaandrager. “Adding Symmetry Reduction to Uppaal”. In: *FORMATS*. 2003, pages 46–59.

[Hen12] Thomas A. Henzinger. “Quantitative Reactive Models”. In: *Model Driven Engineering Languages and Systems - 15th International Conference, MODELS 2012, Innsbruck, Austria, September 30-October 5, 2012. Proceedings*. 2012, pages 1–2.

[Her02] Holger Hermanns. *Interactive Markov Chains: The Quest for Quantified Quality*. Volume 2428. Lecture Notes in Computer Science. Springer, 2002. ISBN: 3-540-44261-8.

[HJ08] H. Hermanns and S. Johr. “May we reach it? Or must we? In what time? With what probability?” In: *MMB*. VDE Verlag, 2008, pages 125–140.

[HK09] Holger Hermanns and Joost-Pieter Katoen. “The How and Why of Interactive Markov Chains”. In: *Formal Methods for Components and Objects*. Volume 6286. LNCS. Springer, 2009, pages 311–337.

[HKK13] Holger Hermanns, Jan Krcál, and Jan Kretínský. “Compositional Verification and Optimization of Interactive Markov Chains”. In: *CONCUR*. 2013, pages 364–379.

[HKP13] Michael Huth, Jim Huan-Pu Kuo, and Nir Piterman. “Fatal Attractors in Parity Games”. In: *Foundations of Software Science and Computation Structures - 16th International Conference*. 2013, pages 34–49.

[HKP14] Michael Huth, Jim Huan-Pu Kuo, and Nir Piterman. “Fatal Attractors in Parity Games: Building Blocks for Partial Solvers”. In: *CoRR* abs/1405.0386 (2014).

[HKV16] Holger Hermanns, Jan Krcál, and Steen Vester. “Distributed Synthesis in Continuous Time”. In: *Foundations of Software Science and Computation Structures (FOSSACS) - 19th International Conference*. 2016, pages 353–369.

[HMU03] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to automata theory, languages, and computation - international edition (2. ed)*. Addison-Wesley, 2003. ISBN: 978-0-321-21029-6.

[Hol04] Gerard J. Holzmann. *The SPIN Model Checker - primer and reference manual*. Addison-Wesley, 2004. ISBN: 978-0-321-22862-8.

[Hol95] Markus Holzer. “On Emptiness and Counting for Alternating Finite Automata”. In: *Developments in Language Theory*. 1995, pages 88–97.

[Hun15] Paul Hunter. “Reachability in Succinct One-Counter Games”. In: *Reachability Problems - 9th International Workshop, RP 2015, Warsaw, Poland, September 21-23, 2015, Proceedings*. 2015, pages 37–49.

[ID96] C. Norris Ip and David L. Dill. “Better Verification Through Symmetry”. In: *Formal Methods in System Design* 9.1/2 (1996), pages 41–75.

[JL03] J. Johannsen and M. Lange. “CTL+ Is Complete for Double Exponential Time”. In: *Proc. ICALP'03*. Springer, LNCS 2719, 2003, pages 767–775.

[JPZ06] Marcin Jurdzinski, Mike Paterson, and Uri Zwick. “A deterministic subexponential algorithm for solving parity games”. In: *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms*. 2006, pages 117–123.

[JS07] Petr Jancar and Zdenek Sawa. “A note on emptiness for alternating finite automata with a one-letter alphabet”. In: *Inf. Process. Lett.* 104.5 (2007), pages 164–167.

[Jur00] Marcin Jurdzinski. “Small Progress Measures for Solving Parity Games”. In: *17th Annual Symposium on Theoretical Aspects of Computer Science*. 2000, pages 290–301.

[Jur98] Marcin Jurdzinski. “Deciding the Winner in Parity Games is in UP cap co-Up”. In: *Inf. Process. Lett.* 68.3 (1998), pages 119–124.

[Kat+11] J.-P. Katoen, I.S. Zapreev, E.M. Hahn, H. Hermanns, and D.N. Jansen. “The ins and outs of the probabilistic model checker MRMC”. In: *Performance Evaluation* 68.2 (2011), pages 90–104.

[Kei14] Jeroen J. A. Keiren. “Benchmarks for Parity Games”. In: *CoRR* abs/1407.3121 (2014).

[KKN09] J.-P. Katoen, D. Klink, and M. R. Neuhäußer. “Compositional Abstraction for Stochastic Systems”. In: *FORMATS*. Volume 5813. LNCS. Springer, 2009, pages 195–211.

[KNP02] Marta Z. Kwiatkowska, Gethin Norman, and David Parker. “PRISM: Probabilistic Symbolic Model Checker”. In: *Computer Performance Evaluation, Modelling Techniques and Tools 12th International Conference, TOOLS 2002, London, UK, April 14-17, 2002, Proceedings*. 2002, pages 200–204.

[KNP06] Marta Z. Kwiatkowska, Gethin Norman, and David Parker. “Symmetry Reduction for Probabilistic Model Checking”. In: *CAV*. 2006, pages 234–248.

[Kön36] Dénes König. *Theorie der endlichen und unendlichen Graphen: kombinatorische Topologie der Streckenkomplexe*. Leipzig: Akad. Verlag, 1936.

[Koz83] Dexter Kozen. “Results on the Propositional mu-Calculus”. In: *Theor. Comput. Sci.* 27 (1983), pages 333–354.

[Kuh53] Harold W. Kuhn. “Extensive games and the problem of information”. In: *Annals of Mathematics Studies* 28 (1953).

[Kup+00] Orna Kupferman, P. Madhusudan, P. S. Thiagarajan, and Moshe Y. Vardi. “Open Systems in Reactive Environments: Control and Synthesis”. In: *CONCUR*. 2000, pages 92–107.

[LM15] François Laroussinie and Nicolas Markey. “Augmenting ATL with strategy contexts”. In: *Information and Computation* (2015). (To appear).

[LMO08] François Laroussinie, Nicolas Markey, and Ghassan Oreiby. “On the Expressiveness and Complexity of ATL”. In: *Logical Methods in Computer Science* 4.2 (2008).

[LMS04] Christof Löding, P. Madhusudan, and Olivier Serre. “Visibly Pushdown Games”. In: *FSTTCS*. 2004, pages 408–420.

[LPY97] Kim Guldstrand Larsen, Paul Pettersson, and Wang Yi. “UPPAAL in a Nutshell”. In: *STTT* 1.1-2 (1997), pages 134–152.

[Mad97] Angelika Mader. *Verification of Modal Properties Using Boolean Equation Systems*. Edition versal 8. Berlin, Germany: Bertz Verlag, 1997.

[Mil83] Robin Milner. “Calculi for Synchrony and Asynchrony”. In: *Theor. Comput. Sci.* 25 (1983), pages 267–310.

[Min61] Marvin L. Minsky. “Recursive Unsolvability of Post’s Problem of “Tag” and other Topics in Theory of Turing Machines”. In: *The Annals of Mathematics* 74.3 (November 1961), pages 437–455.

[MMV10] Fabio Mogavero, Aniello Murano, and Moshe Y. Vardi. “Reasoning about strategies”. In: *Proceedings of the 30th Conferentce on Foundations of Software Technology and Theoretical Computer Science (FSTTCS’10)*. 2010, pages 133–144.

[Mog+14] Fabio Mogavero, Aniello Murano, Giuseppe Perelli, and Moshe Y. Vardi. “Reasoning About Strategies: On the Model-Checking Problem”. In: *ACM Transactions on Computational Logic* 15.4 (August 2014), 34:1–34:47.

[Mos84] Andrzej Wlodzimierz Mostowski. “Regular expressions for infinite trees and a standard form of automata”. In: *Computation Theory - Fifth Symposium,* 1984, pages 157–168.

[MP15] Aniello Murano and Giuseppe Perelli. “Pushdown Multi-Agent System Verification”. In: *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*. 2015, pages 1090–1097.

[MS03] Nicolas Markey and Ph. Schnoebelen. “Model Checking a Path”. In: *CONCUR*. 2003, pages 248–262.

[Mur89] T. Murata. “Petri Nets: Properties, Analysis and Applications.” In: *Proceedings of the IEEE* 77.4 (April 1989), pages 541–580.

[Mus15] Anca Muscholl. "Automated Synthesis of Distributed Controllers". In: *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part II*. 2015, pages 11–27.

[MV14] Nicolas Markey and Steen Vester. "Symmetry Reduction in Infinite Games with Finite Branching". In: *Proceedings of the 12th International Symposium on Automated Technology for Verification and Analysis (ATVA)*. Volume 8837. Lecture Notes in Computer Science. Springer, November 2014, pages 281–296.

[Nas50] John F. Nash Jr. "Equilibrium Points in n-Person Games". In: *Proc. National Academy of Sciences* 36.1 (January 1950), pages 48–49.

[Nas51] John F. Nash Jr. "Non-cooperative Games". In: *Annals of Mathematics* 54.2 (September 1951), pages 286–295.

[Neu81] Marcel F Neuts. *Matrix-geometric solutions in stochastic models: an algorithmic approach*. Courier Corporation, 1981.

[Nis+07] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V. Vazirani. *Algorithmic Game Theory*. New York, NY, USA: Cambridge University Press, 2007. ISBN: 0521872820.

[Obd03] Jan Obdrzálek. "Fast Mu-Calculus Model Checking when Tree-Width Is Bounded". In: *Computer Aided Verification, 15th International Conference, CAV, Proceedings*. 2003, pages 80–92.

[OR94] Martin J. Osborne and Ariel Rubinstein. *A Course in Game Theory*. MIT Press, 1994.

[Osb04] Martin J. Osborne. *An Introduction to Game Theory*. Oxford University Press, 2004.

[Pap94] Christos H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994. ISBN: 978-0-201-53082-7.

[Paz71] Azaria Paz. *Introduction to Probabilistic Automata (Computer Science and Applied Mathematics)*. Orlando, FL, USA: Academic Press, Inc., 1971. ISBN: 0125476507.

[PD12] Silvia S. Pelozo and Pedro R. D'Argenio. "Security Analysis in Probabilistic Distributed Protocols via Bounded Reachability". In: *TGC*. 2012, pages 182–197.

[Pet62] Carl Adam Petri. "Kommunikation mit Automaten". ger. PhD thesis. Universität Hamburg, 1962.

[Pit07] Nir Piterman. "From Nondeterministic Büchi and Streett Automata to Deterministic Parity Automata". In: *Logical Methods in Computer Science* 3.3 (2007).

[Pnu77] Amir Pnueli. "The Temporal Logic of Programs". In: *FOCS*. 1977, pages 46–57.

[PPS06] Nir Piterman, Amir Pnueli, and Yaniv Sa'ar. "Synthesis of Reactive(1) Designs". In: *Verification, Model Checking, and Abstract Interpretation, 7th International Conference, VMCAI 2006, Charleston, SC, USA, January 8-10, 2006, Proceedings*. 2006, pages 364–380.

[PR89a] Amir Pnueli and Roni Rosner. "On the Synthesis of a Reactive Module". In: *POPL*. 1989, pages 179–190.

[PR89b] Amir Pnueli and Roni Rosner. "On the Synthesis of an Asynchronous Reactive Module". In: *ICALP*. 1989, pages 652–671.

[PR90] Amir Pnueli and Roni Rosner. "Distributed Reactive Systems Are Hard to Synthesize". In: *FOCS*. 1990, pages 746–757.

[Put09] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. Volume 414. John Wiley & Sons, 2009.

[Rei84] John H. Reif. "The Complexity of Two-Player Games of Incomplete Information". In: *J. Comput. Syst. Sci.* 29.2 (1984), pages 274–301.

[San10] Tuomas Sandholm. "The State of Solving Large Incomplete-Information Games, and Application to Poker". In: *AI Magazine* 31.4 (2010), pages 13–32.

[SC85] A. Prasad Sistla and Edmund M. Clarke. "The Complexity of Propositional Linear Temporal Logics". In: *J. ACM* 32.3 (1985), pages 733–749.

[Sch07] Sven Schewe. "Solving Parity Games in Big Steps". In: *FSTTCS 2007: Foundations of Software Technology and Theoretical Computer Science, 27th International Conference, Proceedings*. 2007, pages 449–460.

[Sch08] Sven Schewe. "ATL* Satisfiability Is 2EXPTIME-Complete". In: *Proc. of ICALP (2)*. 2008, pages 373–385.

[Seg95] Roberto Segala. "Modeling and Verification of Randomized Distributed Real-Time Systems". PhD thesis. Massachusetts Institute of Technology, 1995.

[Ser06] Olivier Serre. "Parity Games Played on Transition Graphs of One-Counter Processes". In: *FoSSaCS*. 2006, pages 337–351.

[Sha53] L.S. Shapley. "Stochastic Games". In: *Proceedings of the National Academy of Sciences* 39.10 (1953), pages 1095–1100.

[Sti95] Colin Stirling. "Local Model Checking Games". In: *CONCUR '95: Concurrency Theory, 6th International Conference, Proceedings*. 1995, pages 1–11.

[SV10] Lutz Schröder and Yde Venema. "Flat Coalgebraic Fixed Point Logics". In: *Proc. of CONCUR'2010*. Springer, LNCS 6269, 2010, pages 524–538.

[Tho08] Wolfgang Thomas. "Church's Problem and a Tour through Automata Theory". In: *Pillars of Computer Science, Essays Dedicated to Boris (Boaz) Trakhtenbrot on the Occasion of His 85th Birthday*. 2008, pages 635–655.

[Tho95] Wolfgang Thomas. “On the Synthesis of Strategies in Infinite Games”. In: *STACS*. 1995, pages 1–13.

[Tur36] Alan M. Turing. “On Computable Numbers, with an Application to the Entscheidungsproblem”. In: *Proceedings of the London Mathematical Society* 2.42 (1936), pages 230–265.

[UW11] Michael Ummels and Dominik Wojtczak. “The Complexity of Nash Equilibria in Stochastic Multiplayer Games”. In: *Logical Methods in Computer Science* 7.3:20 (September 2011).

[Var98] Moshe Y. Vardi. “Reasoning about The Past with Two-Way Automata”. In: *ICALP*. 1998, pages 628–641.

[Ves12] Steen Vester. *Symmetric Nash equilibria*. Research Report LSV-12-23. Laboratoire Spécification et Vérification, ENS Cachan, France, December 2012.

[Ves15] Steen Vester. “On the Complexity of Model-Checking Branching and Alternating-Time Temporal Logics in One-Counter Systems”. In: *Proceedings of the 13th International Symposium on Automated Technology for Verification and Analysis (ATVA)*. 2015, pages 361–377.

[Ves16] Steen Vester. “Winning Cores in Parity Games”. To appear in *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. 2016.

[VJ00] Jens Vöge and Marcin Jurdzinski. “A Discrete Strategy Improvement Algorithm for Solving Parity Games”. In: *Computer Aided Verification, 12th International Conference, Proceedings*. 2000, pages 202–215.

[VW86] Moshe Y. Vardi and Pierre Wolper. “An Automata-Theoretic Approach to Automatic Program Verification (Preliminary Report)”. In: *Proceedings of the Symposium on Logic in Computer Science (LICS ’86), Cambridge, Massachusetts, USA, June 16-18, 1986*. 1986, pages 332–344.

[Wal+06] Dirk Walther, Carsten Lutz, Frank Wolter, and Michael Wooldridge. “ATL satisfiability is indeed ExpTime-complete”. In: *Journal of Logic and Computation* 16.6 (2006), pages 765–787.

[WBE08] Thomas Wahl, Nicolas Blanc, and E. Allen Emerson. “SVISS: Symbolic Verification of Symmetric Systems”. In: *TACAS*. 2008, pages 459–462.

[WS11] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011. ISBN: 978-0-521-19527-0.

[WVS83] Pierre Wolper, Moshe Y. Vardi, and A. Prasad Sistla. “Reasoning about Infinite Computation Paths (Extended Abstract)”. In: *FOCS*. 1983, pages 185–194.

[Zie98] Wieslaw Zielonka. “Infinite Games on Finitely Coloured Graphs with Applications to Automata on Infinite Trees”. In: *Theor. Comput. Sci.* 200.1-2 (1998), pages 135–183.

Appendix for Chapter 10

Definition of the probability measure

First, we pose an assumption on the module of each player j to avoid that a play stops because no further steps are possible. More precisely, we assume that every state has at least one outgoing delay transition or it only has outgoing private action transitions (that cannot be blocked by other modules).

One can see that this is no real restriction as follows. For states without any transition, we can add a delay self-loop; states with synchronisation transitions are transformed as depicted in Figure 1. Note that in the second case, simply adding delay self-loops would change the behaviour because taking a delay self-loop allows the player to change the choice.

As common for continuous-time systems, the definition is based on *cylinder* sets generated from interval-timed histories. An *interval-timed history* is a sequence

$$H = s_0 c_0 \overset{a_1, I_1}{\to} s_1 c_1 \cdots \overset{a_k, I_k}{\to} s_k$$

where each I_{i+1} is a real interval bounding the time spent *waiting in* s_i. We further require that $I_i = [0, 0]$ whenever $a_i \in \Sigma$ and that the set of histories that *conform* to H is non-empty. We say that a history $s_0 c_0 \overset{a_1, t_1}{\to} s_1 c_1 \cdots \overset{a_k, t_k}{\to} s_k$ *conforms to* H if $t_i - t_{i-1} \in I_i$ for each $1 \leq i \leq k$ (where $t_0 = 0$).

Slightly abusing notation, we interpret an interval-timed history H also as the set of histories that conform to H. We define the cylinder $\mathrm{Cyl}(H) = \{\rho \in \mathrm{Play} \mid \exists i. \rho_{\leq i} \in$

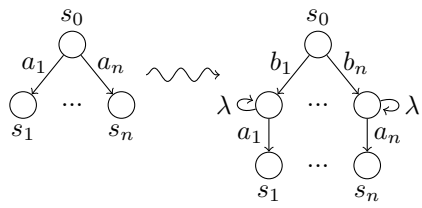


Figure 1: Transformation for states with synchronisation transitions that adds new private actions $b_1, \ldots, b_n$.

$H\}$ as the set of plays which have a prefix in H. We define the measurable spaces $(\text{Play}, \mathcal{F})$ and $(\text{Hist}, \mathcal{H})$ where $\mathcal{F}$ and $\mathcal{H}$ are the σ-algebra generated from all cylinders and interval-timed histories, respectively:

$$\mathcal{F} = \sigma\left(\{\text{Cyl}(H) \mid H \text{ is an interval-timed history}\}\right)$$
$$\mathcal{H} = \sigma\left(\{H \mid H \text{ is an interval-timed history}\}\right)$$

Analogously, we can define measurable spaces $(\text{Hist}_j, \mathcal{H}_j)$ over local histories (by allowing also I_i for action transitions to have non-zero length).

For a given *strategy profile* σ, i.e. a tuple of strategies $\sigma = (\sigma_1, \ldots, \sigma_n)$ of individual players, a given scheduler δ and initial state s_0, we obtain a purely stochastic process and we can define a probability measure $\Pr_{s_0}^{\sigma,\delta}$ over Play. The probability measure is uniquely determined by fixing probabilities for cylinder sets $\text{Cyl}(H)$ for any interval-timed history

$$H = s_0 c_0 \overset{a_1, I_1}{\to} s_1 c_1 \cdots \overset{a_k, I_k}{\to} s_k.$$

Let $\alpha_1, ..., \alpha_v$ be the indices of delay transitions, i.e. for each α_j we have $a_{\alpha_j} \in \Pi$; and $\beta_1, ..., \beta_u$ be the indices of action transitions, i.e. for each β_j we have $a_{\beta_j} \in \Sigma$. For $1 \leq i \leq v$, let $\ell_i = \inf(I_{\alpha_i})$ and $u_i = \sup(I_{\alpha_i})$. Then $\Pr_{s_0}^{\sigma,\delta}(\text{Cyl}(H))$ is defined as

$$\int_{\ell_1}^{u_1} \ldots \int_{\ell_v}^{u_v} \prod_{1 \leq i \leq v} De^{\alpha_i}(d_i) \cdot \prod_{0 \leq i < k} St^i \cdot \prod_{1 \leq i \leq u} Sc^{\beta_i} \, \mathrm{d}d_v \cdots \mathrm{d}d_1$$

where the terms $De^i(d)$, Sc^i, and St^i express the contribution of the ith transition to the overall probability caused by the delays, decisions of the scheduler, and decision on the strategies, respectively (see below). The variables d_i denote the delay at ith delay transitions and induce a history

$$h = s_0 c_0 \overset{a_1, t_1}{\to} s_1 c_1 \cdots \overset{a_k, t_k}{\to} s_k$$

such that $t_i = \sum_{\ell, \alpha_\ell \leq i} d_{\alpha_\ell}$. Finally, we set

$$De^i(d) = Q(s_i, s_{i+1}) \cdot e^{-E(s_i)d},$$

where $Q(s, s') = \sum_{s \overset{\lambda, j}{\rightsquigarrow} s'} \lambda$ and $E(s) = \sum_{s' \neq s} Q(s, s')$, and

$$St^i = \prod_{j \in \text{Move}(a_i)} \sigma_j(\pi_j(h_{\leq i}))(c_i(j)),$$
$$Sc^i = \delta(h_{\leq i-1}, c_{i-1})(a_i).$$

When $s_0' \neq s_0$ then $\Pr_{s_0'}^{\sigma,\delta}(\text{Cyl}_{\mathcal{G}}(H)) = 0$.

Proof of Theorem 10.11

Before we prove Theorem 10.11 we need a few definitions and lemmas.

For an interval-timed history let $\pi_j(H) = \{\pi_j(\rho) \in \text{Play} \mid \rho \text{ conforms to } H\}$. Further, Let $\sim$ be an equivalence relation on interval-timed histories defined such that $H \sim H'$ for two interval-timed histories H and H' if and only if $\pi_j(H) = \pi_j(H')$ for every $j \in \Pi$. We write $[H]$ for the set of histories h such that there exists $H' \sim H$ with $h \in H'$.

Now, for an interval-timed history $H = s_0c_0 \overset{a_1,I_1}{\to} s_1c_1 \cdots \overset{a_k,I_k}{\to} s_k$ such that the last action $a_k \in \Pi$ if $k > 0$ we define the interleaving-abstract cylinder as the set of plays with a prefix conforming to interval-timed histories H' assuring $H \sim H'$:

$$\text{Cyl}^{\text{ia}}(H) = \{\rho \in \text{Play} \mid \rho_{\leq k} \in [H]\}$$

Note that the interleaving-abstract cylinders are contained in the σ-algebra $\mathcal{F}$ generated by the cylinder sets. We can therefore define a sub-σ-algebra $\mathcal{I}$ of $\mathcal{F}$ generated by interleaving-abstract cylinders

$$\mathcal{I} = \sigma(\{\text{Cyl}^{\text{ia}}(H) \mid H \text{ is an interval-timed history s.t. the last action is not in } \Sigma\})$$

Since this is a sub-σ-algebra of $\mathcal{F}$ it inherits the probability measures defined earlier restricted to $\mathcal{I}$. Note that interleaving-abstract events are also 0-time abstract. That is, they are events which are invariant under reordering of 0-time interactions. Indeed, if no player can distinguish two histories h and h', then the delay transitions must be the same in the two histories. Further, since all players have access to global time it must be the same actions that are performed in h and h' in every 0-duration sub-history Now, the only way in which h and h' can differ is in the interleaving of these 0-duration subhistories.

Lemma .1 *Let $\sigma = (\sigma_1, ..., \sigma_n)$ be a pure strategy profile and let δ, δ' be two schedulers. Further, let $H = s_0c_0 \overset{a_1,I_1}{\to} s_1c_1 \cdots \overset{a_k,I_k}{\to} s_k$ be an interval-timed history such that $a_i \in \Sigma$ for all $1 \leq i < k$ and $a_k \in \Pi$. Then*

$$\Pr_{s_0}^{\sigma,\delta}(\text{Cyl}^{\text{ia}}(H)) = \Pr_{s_0}^{\sigma,\delta'}(\text{Cyl}^{\text{ia}}(H))$$

Further, this probability is either 0 or 1.

Proof. First note that before a delay transition happens there can be no loops in any of the local histories $\pi_j(H)$. When we restrict to histories up to the first delay transition, a pure strategy for player j can be considered to be simply a maximal sequence $(a_1^j, s_1^j)...(a_{\ell_j}^j, s_{\ell_j}^j)$ of choices of player j. This is because for any strategy σ_j of player j, the j-play moves along the unique path in the module of player j that is chosen by player j at each step. The only thing player j can observe is whether the transition (that he chose) is taken or not.

Assume now that we have a pure strategy profile $\sigma = (\sigma_1, ..., \sigma_n)$. This induces such a sequence of choices for each player. The longest possible j-play that can occur under σ before the first delay transition is $\rho^j = s_0(j)(a_1^j, s_1^j), \overset{a_1^j,0}{\to} s_1^j ... \overset{a_{\ell_j}^j,0}{\to} s_{\ell_j}^j$. Further, all possible j-plays are prefixes of ρ^j. We now suppose for contradiction that there are two histories h, h' consistent with σ ending after the first delay transition such that there exists a player j with $|\pi_j(h)| \neq |\pi_j(h')|$. Since $\pi_j(h)$ and $\pi_j(h')$ are both prefixes of ρ^j either $\pi_j(h)$ is a proper prefix of $\pi_j(h')$ or the other way around. Without loss of generality let $u = |\pi_j(h)| < |\pi_j(h')|$.

Suppose $\sigma_j(\pi_j(h)) = (b_0, s)$. Now, player j has not been able to synchronize on $b_0 \in \Sigma$ in the last state of h. However, he has been able to synchronize on it along h' due to different choices of the scheduler. Note that at earlier points on h he might have synchronized on b_0 a number of times already. Let this number of times be $c_0 \in \mathbb{N}$. This means that all players capable of synchronizing on b_0 must have done so exactly c_0 times along h. At least one of these players, let us call him j_1, must have stopped before committing to synchronize on b_0 the (c_0+1)th time, because otherwise the play would have progressed since no action can be enabled when a delay transition takes place. Note that each of these players are willing to perform b_0 at least $c_0 + 1$ times at some point since this happens in h'. Let j_1 be committed to synchronize on action $b_1 \neq b_0$ when the play stops in h and suppose he has already synchronized c_1 times on b_1 before this point. We can now perform the same reasoning again to find a player j_2 that has stopped in h before reaching the point where he is ready to synchronize on b_1 for the (c_1+1)th time. At this point he is committed to performing the action b_2 for the $(c_2 + 1)$th time. This reasoning gives us an infinite sequence j_i of players committed to actions b_i in the last state after having performed the action b_i exactly c_i times before.

We now introduce a partial order $\preceq \subseteq (\Sigma \times \mathbb{N})^2$ on elements $(a, d) \in (\Sigma \times \mathbb{N})$ such that there exists j so action a occurs at least d times in h'. The relation is defined such that $(a, d) \preceq (a', d')$ if there is a player j such that a occurs d times on h' before a' occurs d' times on h' (and a' actually does occur d' times at some point on h'). It is reflexive and transitive because all players that has an action in their alphabet must commit in order to synchronize on it. Anti-symmetry follows from this and the fact that ρ^j is linear for every j.

Now, we have that $(b_{i+1}, c_{i+1}) \prec (b_i, c_i)$ for all $i \geq 0$. This is the case firstly because $\pi_{j_i}(h)$ is a prefix of $\pi_{j_i}(h')$ for the players giving rise to the sequence of b_i's. Secondly, because $b_{i+1} \neq b_i$. This means that

$$(b_0, c_0) \succ (b_1, c_1) \succ (b_2, c_2)...$$

is an infinite strictly decreasing sequence. Since $\succ$ is only defined on a finite number of elements this gives a contradiction. Thus, $|\pi_j(h)| = |\pi_j(h')|$. Further, $\pi_j(h) = \pi_j(h')$ since one is a prefix of the other. Since j was chosen arbitrarily, this means that the local history $\pi_j(h)$ before the delay transition cannot be changed

by any scheduler and is uniquely determined by the pure strategies. From this, the lemma follows. □

We now extend to arbitrary events in $\mathcal{I}$ by applying the result above. However, first we need some notation. If $H = s_0 c_0 \overset{a_1,I_1}{\to} s_1 c_1 \cdots \overset{a_k,I_k}{\to} s_k$ is an interval-timed history then the set of histories $h \in [H]$ with specific delays $d_1, ..., d_v$ on the delay transitions is denoted $[H]^{d_1,...,d_v}$. Note that this set is finite.

Lemma .2 *Let $E \in \mathcal{I}$, $s_0 \in S$, $\sigma = (\sigma_1, ..., \sigma_n)$ be a pure strategy profile and δ, δ' be two schedulers. Then*

$$\Pr_{s_0}^{\sigma,\delta}(E) = \Pr_{s_0}^{\sigma,\delta'}(E)$$

Proof. We show this by showing that for every time-abstract cylinder $\mathrm{Cyl}^{\mathrm{ia}}(H)$ for an interval-timed history $H = s_0 c_0 \overset{a_1,I_1}{\to} s_1 c_1 \cdots \overset{a_k,I_k}{\to} s_k$ such that $a_k \in \Pi$ if $k > 0$, every pure strategy profile σ and every pair δ, δ' of schedulers we have

$$\Pr_{s_0}^{\sigma,\delta}(\mathrm{Cyl}^{\mathrm{ia}}(H)) = \Pr_{s_0}^{\sigma,\delta'}(\mathrm{Cyl}^{\mathrm{ia}}(H))$$

First, for any scheduler δ we have

$$\Pr_{s_0}^{\sigma,\delta}(\mathrm{Cyl}^{\mathrm{ia}}(H)) = \sum_{H' \sim H} \Pr_{s_0}^{\sigma,\delta}(\mathrm{Cyl}(H'))$$

since these cylinder sets are disjoint.

Suppose that $a_i \in \Pi$ for the indices $\alpha_1, ..., \alpha_v$ and $a_i \in \Sigma$ for the indices $\beta_1, ..., \beta_u$. For $1 \le i \le v$, let $\ell_i = \inf(I_{\alpha_i})$ and $u_i = \sup(I_{\alpha_i})$. Next, consider fixed delays $d_1, ..., d_v$ and let $[H]^{d_1,...,d_v} = \{h^1, ..., h^r\}$. For $1 \le m \le r$ we denote

$$h^m = s_0^m c_0^m \overset{a_1^m,t_1}{\to} s_1^m c_1^m \cdots \overset{a_k^m,t_k}{\to} s_k^m$$

where the timestamps t_i are induced by the delays $d_1, ..., d_v$ on the delay transitions. Now, since every interval-timed history $H' \sim H$ contains the same intervals and same delay transitions we have

$$\begin{aligned}
&\Pr_{s_0}^{\sigma,\delta}(\mathrm{Cyl}^{\mathrm{ia}}(H)) \\
&= \sum_{H' \sim H} \Pr_{s_0}^{\sigma,\delta}(\mathrm{Cyl}(H')) \\
&= \int_{\ell_1}^{u_1} ... \int_{\ell_v}^{u_v} \prod_{1 \le i \le v} Q(s^m_{\alpha_i}, s^m_{\alpha_i+1}) \cdot e^{-E(d_i)} \\
&\cdot \sum_{h^m \in [H]^{d_1,...,d_v}} \Big(\prod_{1 \le i \le u} \delta(h^m_{\le \beta_i - 1}, c^m_{\beta_i - 1})(a^m_{\beta_i})
\end{aligned}$$

$$\cdot \prod_{0\le i<k} \prod_{j\in \mathrm{Move}(a_i^m)} \sigma_j(\pi_j(h^m_{\le i}))(c_i^m(j))\Bigg)\, \mathrm{d}d_v \cdots \mathrm{d}d_1$$

We will now show by induction on v that for any fixed delays $d_1, ..., d_v$

$$p^{d_1,\ldots,d_v} = \sum_{h^m\in[H]^{d_1,\ldots,d_v}} \Bigg(\prod_{1\le i\le u} \delta(h^m_{\le\beta_i-1}, c^m_{\beta_i-1})(a^m_{\beta_i})$$

$$\prod_{0\le i<k} \prod_{j\in \mathrm{Move}(a_i^m)} \sigma_j(\pi_j(h^m_{\le i}))(c_i^m(j))\Bigg)$$

either equals 0 or 1 independently of the scheduler δ. This implies that $\Pr^{\sigma,\delta}_{s_0}(\mathrm{Cyl}^{\mathrm{ia}}(H))$ is independent of the scheduler and thus proves the lemma.

For the base case suppose that $v = 0$. Then the only possibility is $H = s_0$ since H cannot end with an action transition. In this case it is immediate that $p^{d_1,\ldots,d_v} = 1$.

For the inductive case suppose that $v > 0$ and that it holds for all H' with less than v delay transitions. Note that $s_{\alpha_{v-1}}$ is the same state for every $h \in [H]^{d_1,\ldots,d_v}$. Thus, we have $[H]^{d_1,\ldots,d_v} = \{h \cdot h' \mid h \in [H_{\le\alpha_{v-1}}]^{d_1,\ldots,d_{v-1}} \text{ and } h' \in [H_{\ge\alpha_{v-1}}]^{d_v}\}$. That is, the set of histories in $[H]^{d_1,\ldots,d_v}$ is obtained by gluing together every prefix up to $s_{\alpha_{v-1}}$ with every suffix starting in $s_{\alpha_{v-1}}$. For two histories h^m and h^n in these two sets we denote their concatenation (where final state of h^m is merged with initial state of h^n) by h^{mn}. This gives us

$$p^{d_1,\ldots,d_v} = \sum_{h^m\in[H_{\le\alpha_{v-1}}]^{d_1,\ldots,d_{v-1}}} \sum_{h^n\in[H_{\ge\alpha_{v-1}}]^{d_v}} \Bigg($$

$$\prod_{i:\beta_i<\alpha_{v-1}} \delta(h^m_{\le\beta_i-1}, c^m_{\beta_i-1})(a^m_{\beta_i})$$

$$\prod_{0\le i<\alpha_{v-1}} \prod_{j\in \mathrm{Move}(a_i^m)} \sigma_j(\pi_j(h^m_{\le i}))(c_i^m(j))$$

$$\prod_{i:\beta_i>\alpha_{v-1}} \delta(h^{mn}_{\le\beta_i-1}, c^{mn}_{\beta_i-1})(a^{mn}_{\beta_i})$$

$$\prod_{\alpha_{v-1}\le i<\alpha_v} \prod_{j\in \mathrm{Move}(a_i^{mn})} \sigma_j(\pi_j(h^{mn}_{\le i}))(c_i^{mn}(j))\Bigg)$$

$$= \sum_{h^m\in[H_{\le\alpha_{v-1}}]^{d_1,\ldots,d_{v-1}}}$$

$$\cdot \prod_{i:\beta_i<\alpha_{v-1}} \delta(h^m_{\le\beta_i-1}, c^m_{\beta_i-1})(a^m_{\beta_i})$$

$$\cdot \prod_{0\le i<\alpha_{v-1}} \prod_{j\in \mathrm{Move}(a_i^m)} \sigma_j(\pi_j(h^m_{\le i}))(c_i^m(j))$$

$$\cdot \sum_{h^n \in [H_{\geq \alpha_{v-1}}]^{d_v}} \Bigg(\prod_{i:\beta_i > \alpha_{v-1}} \delta(h^{mn}_{\leq \beta_i - 1}, c^{mn}_{\beta_i - 1})(a^{mn}_{\beta_i}) \prod_{\alpha_{v-1} \leq i < \alpha_v} \prod_{j \in \mathrm{Move}(a^{mn}_i)} \sigma_j(\pi_j(h^{mn}_{\leq i}))(c^{mn}_i(j)) \Bigg)$$

If we can show that the second sum is either 0 or 1 independently of the scheduler then we can apply the induction hypothesis on the remaining part. Note that no player can distinguish between the prefixes h^m since they are in the same equivalence class. Thus, using pure strategies the players can only base their decision on what happens after reaching $s^{mn}_{\alpha_{v-1}}$. Now, using the same technique as in the proof of Lemma .1 the result follows. □

From Lemma .2 we know that the probabilities of events in $\mathcal{I}$ are independent of the scheduler when the strategy profile is pure. Using this we can show that this is also the case for non-pure strategy profiles. The idea of the proof is similar to the proof of Kuhn's Theorem [Kuh53] behavioural strategies are shown equivalent in perfect recall extensive-form games. This intuition is applied in the 0-duration subhistories.

Theorem 10.11. Let $E \in \mathcal{I}$, let $s_0 \in S$ be a state, let $\sigma = (\sigma_1, ..., \sigma_n)$ be a strategy profile and δ, δ' be two schedulers. Then

$$\Pr^{\sigma,\delta}_{s_0}(E) = \Pr^{\sigma,\delta'}_{s_0}(E)$$

Proof. We show this by showing that for every time-abstract cylinder $\mathrm{Cyl}^{\mathrm{ia}}(H)$ for an interval-timed history $H = s_0 c_0 \overset{a_1,I_1}{\to} s_1 c_1 \cdots \overset{a_k,I_k}{\to} s_k$ such that $a_k \in \Pi$ if $k > 0$, every strategy profile σ and every pair δ, δ' of schedulers we have

$$\Pr^{\sigma,\delta}_{s_0}(\mathrm{Cyl}^{\mathrm{ia}}(H)) = \Pr^{\sigma,\delta'}_{s_0}(\mathrm{Cyl}^{\mathrm{ia}}(H))$$

We use the same notation as in the proof of Lemma .2. Again we have

$$\begin{aligned}
&\Pr^{\sigma,\delta}_{s_0}(\mathrm{Cyl}^{\mathrm{ia}}(H)) \\
&= \int_{\ell_1}^{u_1} ... \int_{\ell_v}^{u_v} \prod_{1 \leq i \leq v} Q(s^m_{\alpha_i}, s^m_{\alpha_i + 1}) \cdot e^{-E(d_i)} \\
&\cdot \sum_{h^m \in [H]^{d_1, \ldots, d_v}} \Bigg(\prod_{1 \leq i \leq u} \delta(h^m_{\leq \beta_i - 1}, c^m_{\beta_i - 1})(a^m_{\beta_i}) \\
&\cdot \prod_{0 \leq i < k} \prod_{j \in \mathrm{Move}(a^m_i)} \sigma_j(\pi_j(h^m_{\leq i}))(c^m_i(j)) \Bigg) \, \mathrm{d}d_v \cdots \mathrm{d}d_1
\end{aligned}$$

For fixed delays $d_1, ..., d_v$ we will show that a discrete probability distribution over a finite set of pure strategies gives rise to the same probability as above. As

this is independent of the scheduler by Lemma .2 so is the probability for the mixed strategies. As this holds for all delays, the Theorem follows.

Now, for each history $h^m \in [H]^{d_1,\ldots,d_v}$ we define a pure strategy profile σ^{h^m} that plays according to h^m as well as a probability p^{h^m} defined by

$$p^{h^m} = \prod_{0 \le i < k} \prod_{j \in \mathrm{Move}(a_i^m)} \sigma_j(\pi_j(h^m))(c_i^m(j))$$

If $p = \sum_{h^m \in [H]^{d_1,\ldots,d_v}} p^{h^m} < 1$ then define a pure strategy σ'' that plays such that a history $h \in [H]^{d_1,\ldots,d_v}$ is not possible. Further, define $p^{\sigma''} = 1 - p$. Now, consider the experiment of using scheduler δ and picking either one of the strategy profiles σ^{h^m} with probability p^{h^m} or σ'' with probability $p^{\sigma''}$ and applying these strategies. The probability that a prefix of the play is in $[H]^{d_1,\ldots,d_v}$ in this experiment is independent of the scheduler because of Lemma .2. We will show that it is in fact equal to $\Pr_{s_0}^{\sigma,\delta}(\mathrm{Cyl}^{\mathrm{ia}}(H))$. Indeed, the probability is

$$\begin{aligned}
&\sum_{\sigma^{h^m}} p^{h^m} \sum_{h^m \in [H]^{d_1,\ldots,d_v}} \cdot \Bigg(\prod_{1 \le i \le v} Q(s_{\alpha_i}^m, s_{\alpha_i+1}^m) \cdot e^{-E(d_i)} \\
&\cdot \prod_{1 \le i \le u} \delta(h_{\le \beta_i - 1}^m, c_{\beta_i - 1}^m)(a_{\beta_i}^m) \\
&\cdot \prod_{0 \le i < k} \prod_{j \in \mathrm{Move}(a_i^m)} \sigma_j^{h^m}(\pi_j(h_{\le i}^m))(c_i^m(j)) \Bigg) \\
&= \prod_{1 \le i \le v} Q(s_{\alpha_i}^m, s_{\alpha_i+1}^m) \cdot e^{-E(d_i)} \\
&\cdot \sum_{h^m \in [H]^{d_1,\ldots,d_v}} \left(p^{h^m} \cdot \prod_{1 \le i \le u} \delta(h_{\le \beta_i - 1}^m, c_{\beta_i - 1}^m)(a_{\beta_i}^m) \right)
\end{aligned}$$

By inserting the expression for p^{h^m} the result follows. □

Index