Random Games

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der Rheinisch-Westfälischen Technischen Hochschule Aachen zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von

D.E.A.

Florian Horn

aus Nancy, France

Berichter: Universitätsprofessor Dr. Wolfgang Thomas
Universitätsprofessor Dr. Anca Muscholl

Tag der mündlichen Prüfung: 28 Oktober 2008

Diese Dissertation ist auf den Internetseiten der Hochschulbibliothek online verfügbar.


RANDOM

Aux femmes de ma vie.

Acknowledgements

A thesis is more than a few score pages of theorems. It is a rite of passage in the life of a researcher and as such, an occasion to express gratitude for those who were by my side in this journey.

I want to thank first my grandfather, for the math problems he gave me to pass time during long car trips. They were simple linear equations (I was very young), but they taught me perhaps the most important lesson for my academic career: maths are fun. To this day, I can think, write, or talk about research only when I enjoy it. In this regard, I must also thank my whole family, with its strong scientific tradition, Damien and Romain, for their long-standing friendship, and Caro, the mother of my signature Tigger.

I met Claire in ÉNS Cachan, when we were both newcomers to the scientific world. She became one of my dearest friends, and I owe her more than I can ever say. The community of PhD students from LIAFA, RWTH, LABRI and PPS-on-the-other-side-of-the-elevators had a large part in making research enjoyable. I want to pay tribute, in particular, to Pierre's couch and impressive basket skills and Mathilde's seminal Friday cake$^{\text{TM}}$.

My advisors, Anca and Wolfgang, have been a constant source of inspiration and motivation. I am ever so grateful for their guidance, their help, their comments (even when they were less than pleased), and for the fact that they never denied me the freedom to choose my own research directions. I also have other researchers to thank for more specific skills: Hugo, for example, showed me that there is more to a proof than energetic hand gestures, while Olivier taught me how to ~~reject~~ review papers.

One of the best things in research is the capacity, nay, the necessity to travel far and often. I went to a lot of interesting places, I met a lot of interesting people, and sometimes, I even worked with them. Each of them deserve my thanks, but I will focus on two persons who helped me in my travels: Diana, for her part in the organisation of the GAMES project, and Noelle, for her tolerance to my lack of administrative savvy.

I cannot write without feedback, and I am indebted to each and every reviewer, official or otherwise, who took the time to read and comment on one of my drafts (with special thanks to the author of the review *Page 16: should be "Cthulhu"*). For this thesis, I found a lot of help in the comments of Christel and Marcin, my official reviewers, but also in those of Hugo, Julien, Mathilde, and Pierre who proof-read the drafts of the different chapters. Thanks also to France and Boulet, for the initial and final illustrations.

I had a million things to do and no time on the day of my defence, and I am grateful to the little helpers that Claire drafted at the last minute: Pierre, Julien, Boris, Medhi, Gim, Jade, Eude, Dominique, Mathilde, and Konrad. I also thank everyone who cooked for the after-talk food (the real criterion for a successful defence), and all the attendees, even those who came *only* for the food (more surprisingly, some came only for the talk).

A law of list-like acknowledgements is that there is always someone who should have been in, and is not. Forgive me if that is you: I will try and make it up to you the next time we meet.

Abstract

Games are a classical tool for the synthesis of controllers in reactive systems. In this setting, a game is defined by: an *arena*, which is a graph modelling the system and its evolution; and a *winning condition*, which models the specification that the controller must ensure. In each state, the outgoing transition is chosen either by the controller (Eve), an hostile environment (Adam), or a stochastic law (Random). This process is repeated for an infinite number of times, generating an infinite play whose winner depends on the winning condition.

Our first object of study is the fundamental case of reachability games. We present a new effective approach to the computation of the values, based on permutations of random states. In terms of complexity, the resulting "permutation algorithm" is orthogonal to the classical, strategy-based algorithms: it is exponential in the number of random states, but not in the number of controlled states. We also present an improvement heuristic for this algorithm, inspired by the "strategy improvement" algorithm.

We turn next to the very general class of prefix-independent games. We prove the existence of optimal strategies in these games. We also show that our permutation algorithm can be extended into a "meta-algorithm", turning any qualitative algorithm into a quantitative algorithm.

We study then the complexity of optimal strategies for Muller games, focusing on the amount of memory that can be saved through the use of randomised strategies. Using the Zielonka tree, we show tight bounds on the necessary and sufficient memory needed to define randomised optimal strategies for any given Muller condition. We also propose a polynomial algorithm for the winner problem in *explicit* Muller games. The results of the former chapter yield immediately `NP` and co-`NP` algorithms for the values problem.

Lastly, we consider the *finitary* versions of parity and Streett games, where the regular conditions are supplemented by *universal bounds* on delays. We propose a polynomial algorithm for the winner problem on finitary parity games. For finitary Streett games, a reduction to *Request-Response* games provides an `EXPTIME` algorithm for qualitative problems, and we show that the problem is `PSPACE`-hard.

Contents

Chapter 1

Prequel

"Pleased to meet you
Hope you guess my name
But what's puzzling you
Is the nature of my game"

Sympathy for the Devil
The Rolling Stones

After short introductory remarks on the development of Game Theory in Computer Science in Section 1.1, we describe in Section 1.2 the game model underlying the whole thesis, namely $2\frac{1}{2}$-player simple graph games on finite arenas with boolean winning condition, as well as the main problems on such games. Section 1.3 reviews our main contributions.

1.1 Background

Game Theory is a very versatile paradigm, whose applications range from biology [Smi82] to philosophy [Kav86] via economics [Cou38]. This matches the pervasiveness of games in general in human history: there has never been a society without games, at least since biblical times when the contemporaries of Abraham played the royal game of Ur [Fin07].

It is no surprise, then, that game theory found many applications in computer science: artificial intelligence [GMW87], logic [Bla92], semantics of programming languages [Chr03], etc. The model of graph games, which we use throughout this work, is quite straightforward: two players called

Eve and Adam move alternatively a token between the different positions of a board, with a set of rules which describe the legal moves and decide the winner. There is the fact that plays usually go on forever. Well, it could not be *that* simple, could it?

Automata. In the sixties, the problems of verification and synthesis of digital circuits [Chu62] led to the introduction of automata over infinite words [Büc62] and infinite graph games [McN65]. The first solution to Church's synthesis problem, by Büchi and Landweber, came from a game approach [BL69]. Rabin quickly followed suit, and provided a solution based on tree automata [Rab69].

Automata theory and graph games remain closely linked: two player games can be seen as alternating automata over a one-letter alphabet, while the emptiness of non-deterministic tree automata (on a finite alphabet) can be reduced to the problem of deciding the winner of a two-player game. Furthermore, the existence of strategies with finite memory can be used to complement automata: for example, Gurevich and Harrington used the latest appearance records structure for Muller games —already mentioned by Buchi and McNaughton in unpublished manuscripts— to get a simpler proof of Rabin's theorem [GH82].

The nature of the players strategies, especially with respect to memory, received a lot of attention in the following years, with notably the positional determinacy of parity games [Mos91, EJ91], the *index of appearance records* structure for Rabin/Streett games [BLV96], and the split tree [Zie98], whose analysis provided tight bounds in memory for all Muller conditions [DJW97].

To this day, graph games are one of the most popular and efficient approaches to automata problems: see for example [EWS01] on simulation relations, and [CL08] on the (restricted) star-height problem over trees.

Model checking. In the early eighties, the complexity of program verification outgrew the possibilities of hand-constructed proofs in Floyd-Hoare style logic [OG76]. This led to the introduction of model checking by Clarke and Emerson [CE81], and independently by Queille and Sifakis [QS82], in order to check whether a program meets a specification *without* having to build an explicit proof. The idea is to represent the evolution of a program as a finite Kripke structure, and the specification as a formula of propositional temporal logic. The resulting "model-checking problem" asks whether

a Kripke structure M is a model for a logical formula φ. A more complete exposition of the history of model checking can be found in [Cla08].

Model checking was quickly extended into a verification tool for any real systems, through a step of modelling: the system is represented as a Kripke structure, and the specification as a logical formula. The point of model checking is then to either guarantee the good behaviour of the system, or provide examples of faulty behaviours. The multiplicity of possible situations led to many variations of this problem: models may be finite [VW86, McM93], infinite [BJNT00, BFLP03, Mor08], stochastic [Var85, CY95, BCHG$^+$97], asynchronous [Maz75, GMSZ02] or timed [BKH99, BBBM08], while the specification can be written in several different logics: temporal [Pnu77, EL85], fixpoint [Mos91, EJ91], monadic second order [MP92, Kla97], data [BDM$^+$06].

Graph games provide very natural and robust classes of models for *open* systems, where the agent (represented by Eve) must interact with an uncontrollable environment (represented by Adam) [PR89]. A strategy for Eve is then a controller for the system, while a strategy for Adam is a counter-example for the satisfiability of the specification. Here also, the framework had to undergo a great deal of generalisation to account for all the possible situations: to cite but a few, let us mention stochastic transitions [dA97], concurrent moves [dAHK98, dAH00], timed [dAFH$^+$03] and hybrid [BBJ$^+$08] systems, pushdown arenas [Wal96, Ser05, CHM$^+$08], quantitative rewards [FGK08], and multiple players [MW03, GLZ04, MTY05, GU08].

Classical game theory. Another consequence of the stochastic and concurrent extensions was the reunification of games in computer science and games as studied in mathematical economics. These latter games evolved from one-step matrix games, where the outcome depends on a single and simultaneous choice of actions by the players. Borel introduced the notion of mixed strategy in these games [Bor21], and von Neumann proved the existence of optimal strategies —the well known "min-max theorem"— [vNM44], which was extended to the setting of multiplayer games with the notion of Nash equilibrium [Nas50].

In the early fifties, Shapley introduced stochastic graph games [Sha53] to account for situation where the evolution of the play, and not only the immediate payoffs, depend on the choices of the players. A stochastic game is a (finite) set of matrix games, and a play is a series of moves, instead of a single round. Furthermore, the outcome of a move determines whether and

where the play proceeds for the next move. The reward of a play is the sum of the elementary payoffs —which is finite with probability one, since there is a positive probability to stop in each state. An alternative and slightly different view of this reward is the discounted payoff: the game never stops, but the rewards at each step are affected by cumulative discount factors which guarantee that the reward of a play converges. Several other payoff functions were considered —*e.g.* mean payoff [Gil57], limsup [MS96], and reachability [Con93]— for other behaviours or situations.

The cross-breeding of the two traditions has been fruitful: the strategy improvement algorithm [HK66], in particular, has been extended to parity games in a discrete fashion [VJ00], while computer science techniques provided new insights on classical games problems [FPT04, Rou05, GZ07].

The study of graph games is a thriving topic in computer science, as witnesses the wealth of recent theses on the subject [Maj03, Ser05, Gim06, Cha07d]. The present work is to be my own tessella in this vast and ever-expanding mosaic.

1.2 Definitions

1.2.1 Playing

Our model of games is the graph games, introduced by Zermelo in [Zer13] and extended by Shapley in [Sha53]: an *arena* is a directed graph, where a token moves from state to state along the transitions. This model has known enough variants to prompt the authors of [CJH03] to propose a systematic classification: the variation we consider are infinite $2\frac{1}{2}$-player games on simple finite arenas. Before we proceed with the formal definitions, let us review the meaning of these terms, as well as the alternatives.

"Infinite": Our games never end: a full play is a sequence indexed by non-negative integers, and the winning conditions are defined on infinite plays. Notice that while a play *may* go on forever in a real game, *e.g.* in Go without the "superko" rule, this is usually not the *intended* form of a play. Infinite games subsume finite games, but there is an even more general model, in which the plays are indexed by ordinals [CH08b, CH08a, RS08].

"$2\frac{1}{2}$-player": There are three agents: Eve, Adam and Random. An example of "real" $2\frac{1}{2}$-player game is Backgammon: the three agents are White, Black, and the dices. This also provides natural names for games where one or more agents are absent:

- 2-player games—*deterministic games*— if there are no random moves, *e.g.* Go;
- $1\frac{1}{2}$-player games—*Markov decision processes*— if either Eve or Adam cannot move, *e.g.* Spider solitaire;
- 1-player games—*non-deterministic transition systems*— if only Eve or only Adam can move, *e.g.* Sokoban;
- $\frac{1}{2}$-player games—*Markov chains*— if there are only random moves *e.g.* Progress Quest.
- 0-player games—*deterministic transition systems*— if all the positions have only one successor, *e.g.* Conway's game of life.

It is also possible to consider games with three or more players, but their analysis depends on many assumptions about alliances, king-maker situations, and so on.

"Simple": Each state belongs either to Eve, Adam, or Random, and the owner of the current state decides on his own which transition is to be taken. Furthermore, both Eve and Adam know the exact position of the token at all times. This is in contrast with concurrent games — *e.g.* Janken — and partial-information games — *e.g.* Poker.

"on finite arenas" Throughout this work, we only consider games played on finite arenas. The alternative, of course, is to accept infinite —but finitely representable— arenas. Notice that there are real games with infinite arenas, *e.g.* Monopoly.

Arenas and Plays

Notation 1.1 *A probability distribution γ over a finite set X is a function from X to $[0,1]$ such that $\sum_{x\in X}\gamma(x)=1$. The set of probability distributions over X is denoted by $\mathcal{D}(X)$.*

Formally, we define a $2\frac{1}{2}$-*player arena* $\mathcal{A}$ *over a set of colours* $\mathcal{C}$ as a tuple $(\mathcal{Q}, \mathcal{Q}_E, \mathcal{Q}_A, \mathcal{Q}_R, \mathcal{T}, \delta, \chi)$, where:

- $\mathcal{Q}$ is a finite set whose elements are the *states* of $\mathcal{A}$;
- $\mathcal{Q}_E$, $\mathcal{Q}_A$, and $\mathcal{Q}_R$ partition $\mathcal{Q}$ between Eve's states (graphically represented as $\bigcirc$'s), Adam's states ($\square$'s), and random states ($\triangle$'s);
- $\mathcal{T} \subseteq \mathcal{Q} \times \mathcal{Q}$ is the set of *transitions* of $\mathcal{A}$, and there are no dead-ends: $\forall q \in \mathcal{Q}, \exists s \in \mathcal{Q}, (q, s) \in \mathcal{T}$;
- $\delta : \mathcal{Q} \to \mathcal{D}(\mathcal{Q})$ is the random law on the successors of a state of $\mathcal{Q}_R$, and $\delta(r)(q) > 0 \iff (r, q) \in \mathcal{T}$;
- χ is a partial *colouring function*, mapping the states to the colours $\mathcal{C}$.

Notation 1.2 *In the whole thesis, whenever we call an arena* $\mathcal{A}$*, we implicitly mean that* $\mathcal{A} = (\mathcal{Q}, \mathcal{Q}_E, \mathcal{Q}_A, \mathcal{Q}_R, \mathcal{T}, \delta, \chi)$*. Likewise, the arena* $\mathbb{A}$ *is equal to* $(\mathbb{Q}, \mathbb{Q}_E, \mathbb{Q}_A, \mathbb{Q}_R, \mathbb{T}, \mathbb{\delta}, \mathbb{\chi})$*, and* $\mathfrak{A}$ *to* $(\mathfrak{Q}, \mathfrak{Q}_E, \mathfrak{Q}_A, \mathfrak{Q}_R, \mathfrak{T}, \mathfrak{d}, \mathfrak{X})$.

A *play* ρ *of* $\mathcal{A}$ is a —finite or infinite— path in the graph $(\mathcal{Q}, \mathcal{T})$: a sequence of states such that $\forall i < |\rho| - 1, (\rho_i, \rho_{i+1}) \in \mathcal{T}$. The set of plays starting in a state q is denoted Ω_q. The functions Occ (on finite or infinite plays) and Inf (on infinite plays) denote, respectively, the sets of *occurring* and *limit* states:

$$\mathrm{Occ}(\rho) = \{q \in \mathcal{Q} \mid \exists i, \rho_i = q\} \ ;$$
$$\mathrm{Inf}(\rho) = \{q \in \mathcal{Q} \mid \exists^\infty i, \rho_i = q\} \ .$$

Strategies and measures

Strategies are the "recipes" Adam and Eve use when it is their turn to play. We define most of the concepts from Eve's point of view. Similar notions always exist for Adam, and their definition is straightforward. A (randomised) strategy σ for Eve is a function from the finite prefixes ending in a state of Eve to distributions of probabilities over the legal states:

$$\sigma : \mathcal{Q}^* \mathcal{Q}_E \to \mathcal{D}(\mathcal{Q}) \ ;$$
$$\forall w \in \mathcal{Q}^*, \forall q \in \mathcal{Q}_E, \forall s \in \mathcal{Q}, \sigma(wq)(s) > 0 \Rightarrow (q, s) \in \mathcal{T} \ .$$

A strategy is pure if it does not use randomisation:

$$\forall w, \forall q, \sigma(w)(q) = 0 \vee \sigma(w)(q) = 1 \ .$$

A pure strategy can thus be seen as a function from the prefixes to the states, and we often write $\sigma(w)$ for "the unique state q such that $\sigma(w)(q) = 1$".

A play ρ is *consistent with* σ if and only if $\forall i < |\rho|, \rho_{i-1} \in \mathcal{Q}_E \Rightarrow \sigma(\rho_{0\ldots i-1})(\rho_i) > 0$. The set of plays consistent with σ (resp. τ; σ and τ) is denoted by Ω^σ (resp. Ω^τ; $\Omega^{\sigma,\tau}$). Once an initial state q and two strategies σ and τ have been fixed, $\Omega_q^{\sigma,\tau}$ can naturally be made into a measurable space $(\Omega_q^{\sigma,\tau}, \mathcal{O})$, where $\mathcal{O}$ is the σ-field generated by the cones $\{\mathcal{O}_w \mid w \in \mathcal{Q}^*\}$: $\rho \in \mathcal{O}_w$ if and only if w is a prefix of ρ. The probability measure $\mathbb{P}_q^{\sigma,\tau}$ is recursively defined by:

$$\forall r \in \mathcal{Q}, \mathbb{P}_q^{\sigma,\tau}(\mathcal{O}_r) = \begin{cases} 1 & \text{if } r = q \ , \\ 0 & \text{if } r \neq q \ ; \end{cases}$$

$$\forall w \in \mathcal{Q}^*, (r,s) \in \mathcal{Q}^2, \mathbb{P}_q^{\sigma,\tau}(\mathcal{O}_{wrs}) = \begin{cases} \mathbb{P}_q^{\sigma,\tau}(\mathcal{O}_{wr}) \cdot \sigma(wr)(s) & \text{if } r \in \mathcal{Q}_E \ , \\ \mathbb{P}_q^{\sigma,\tau}(\mathcal{O}_{wr}) \cdot \tau(wr)(s) & \text{if } r \in \mathcal{Q}_A \ , \\ \mathbb{P}_q^{\sigma,\tau}(\mathcal{O}_{wr}) \cdot \delta(r)(s) & \text{if } r \in \mathcal{Q}_R \ . \end{cases}$$

Carathéodory's extension theorem allows us to extend $\mathbb{P}_q^{\sigma,\tau}$ to the Borel sets of $(\Omega_q^{\sigma,\tau}, \mathcal{O})$. When we deal with events, we indifferently use $\rho \in \Gamma$ and $\rho \models \Gamma$, $\Gamma \cup \Delta$ and $\Gamma \vee \Delta$, *et cetera*.

Sub-arenas and end-components

The restriction of an arena $\mathcal{A}$ to a subset X of $\mathcal{Q}$, denoted by $\mathcal{A}_{|X}$ is a *sub-arena of* $\mathcal{A}$ if and only if:

- $\forall q \in X \cap (\mathcal{Q}_E \cup \mathcal{Q}_A), \exists s \in X, (q,s) \in \mathcal{T}$;
- $\forall q \in X \cap \mathcal{Q}_R, \forall s \in \mathcal{Q}, (q,s) \in \mathcal{T} \Rightarrow s \in X$.

The *end-components of* $\mathcal{A}$ [CY95, dA97] are the supports of the strongly connected subarenas of $\mathcal{A}$. Lemma 1.3 is central in many a proof about stochastic games:

Lemma 1.3 ([dA97]) *For any initial state q and strategies σ, τ for Eve and Adam, the limit of the ensuing play is an end-component with probability one.*

Strategies with memory

Strategies can also be defined as *strategies with memory*, for a given set of memory states M. A strategy σ with memory M is then a function from $\mathcal{Q} \times M$ to $\mathcal{D}(\mathcal{Q} \times M)$. Alternatively, a pure strategy with memory M can be described as two separate functions: a "next-move function" $\sigma^{\mathtt{n}} : (\mathcal{Q}_E \times M) \to \mathcal{Q}$ and a "memory-update" function $\sigma^{\mathtt{u}} : (\mathcal{Q} \times M) \to M$. Randomised functions $\sigma^{\mathtt{u}}$ and $\sigma^{\mathtt{n}}$ can also be used to define randomised strategies, but it is not possible to represent all the randomised strategies with memory M in this way: there may be a correlation between the moves and the updates. There is a last, intermediate, model of strategies with memory: a semi-randomised strategy σ with memory M is defined by a randomised "next-move function" $\sigma^{\mathtt{n}} : (\mathcal{Q}_E \times M) \to \mathcal{D}(\mathcal{Q})$ and a pure "memory-update" function $\sigma^{\mathtt{u}} : (\mathcal{Q} \times M) \to M$. However, these strategies are less compact than general randomised strategies with memory.

Notice that any pure (resp. randomised) strategy can be represented as a pure (resp. semi-randomised) strategy with memory $\mathcal{Q}^*$. However, the point is often to get strategies with *finite* memory, or *positional strategies*, where the memory is reduced to a singleton.

In particular, a strategy σ with finite memory M can be used to describe the *restriction of $\mathcal{A}$ to σ*, denoted by $\mathcal{A}^\sigma$. If σ is pure or semi-randomised, we get the following $1\frac{1}{2}$-player arena:

- $\mathcal{Q}^\sigma = \mathcal{Q} \times M$;
- $\mathcal{Q}^\sigma_A = \mathcal{Q}_A \times M$;
- $\mathcal{Q}^\sigma_R = (\mathcal{Q}_R \cup \mathcal{Q}_E) \times M$;
- $\mathcal{T}^\sigma = \begin{array}{l} \{((q,m),(r,n) \mid q \in \mathcal{Q}_A \text{ and } n = \sigma^{\mathtt{u}}(q,m)\} \\ \quad \cup \{((q,m),(r,n) \mid (q,m) \in \mathcal{Q}^\sigma_R \text{ and } \delta(q,m)(r,n) > 0\} \end{array}$
- $\delta^\sigma(q,m)(s,n) = \begin{cases} \delta(q)(s) & \text{if } q \in \mathcal{Q}_R \text{ and } n = \sigma^{\mathtt{u}}(q,m) \\ \sigma^{\mathtt{n}}(q,m)(s) & \text{if } q \in \mathcal{Q}_E \text{ and } n = \sigma^{\mathtt{u}}(q,m) \end{cases}$

The problem with general randomised strategies is that Adam gets too much information: he is not supposed to know the current memory state of Eve. The good notion for a game restricted to a general randomised strategy would be a $1\frac{1}{2}$-player game with partial information, but its analysis is outside of the scope of this thesis.

Attractors and traps

For any subset X of $\mathcal{Q}$, we define the events $\text{Reach}(X) = \{\rho \mid \exists i, \rho_i \in X\}$ and $\text{Reach}^\infty X = \{\rho \mid \exists^\infty i, \rho_i \in X\}$, and the *attractor of Eve to X in $\mathcal{A}$* $\text{Attr}_E(X, \mathcal{A})$ as $\cup_{i>0} X^i$:

$$\begin{aligned} X^0 &= X \\ X^{i+1} &= X^i \cup \{q \in (\mathcal{Q}_E \cup \mathcal{Q}_R) \mid \exists s \in X^i, (q,s) \in \mathcal{T}\} \\ &\quad \cup \{q \in \mathcal{Q}_A \mid \forall s \in \mathcal{Q}, (q,s) \in \mathcal{T} \Rightarrow s \in X^i\} \end{aligned}$$

An *attractor strategy of Eve to X* is a pure and positional strategy $\overrightarrow{a_E}(X)$ such that $\forall q \in \cup_{i>1} X^i, q \in X^i \Rightarrow \overrightarrow{a_E}(X)(q) \in X^{i-1}$. Propositions 1.4 and 1.5 follow directly from the definition of an attractor and Lemma 1.3:

Proposition 1.4 *For any state q in $\text{Attr}_E(X, \mathcal{A})$, there is a real number $\eta > 0$ such that for any strategy τ of Adam, we have:*

$$\mathbb{P}_q^{\overrightarrow{a_E}(X),\tau}(\text{Reach}(X)) > \eta$$

Proposition 1.5 *For any state q in $\mathcal{Q}$, for any strategy τ of Adam, we have:*

$$\mathbb{P}_q^{\overrightarrow{a_E}(X),\tau}(\text{Reach}^\infty(X) \mid \text{Reach}^\infty(\text{Attr}_E(X, \mathcal{A}))) = 1$$

An interesting remark is that the positional randomised strategy $\texttt{uni}_\mathcal{A}$, which chooses any legal successor in $\mathcal{A}$ with a uniform distribution, acts as an universal attractor strategy for any subset X of $\mathcal{A}$ [CdAH04]: Propositions 1.4 and 1.5 still hold if we replace a_X by $\texttt{uni}_\mathcal{A}$.

The dual notion of a *trap X for Eve* denotes a region from which Eve cannot escape:

- $\forall q \in X \cap (\mathcal{Q}_E \cup \mathcal{Q}_R), \forall s \in \mathcal{Q}, (q,s) \in \mathcal{T} \Rightarrow s \in X$;
- $\forall q \in X \cap \mathcal{Q}_A, \exists s \in X, (q,s) \in \mathcal{T}$.

A strategy τ such that $\forall w \in \mathcal{Q}^*, \forall q \in X \cap \mathcal{Q}_A, \tau(wq)(X) = 1$ is a *trapping strategy of Adam in X*.

Proposition 1.6 *For any subset X of $\mathcal{Q}$, $\mathcal{Q} \setminus \text{Attr}_E(X, \mathcal{A})$ is a trap for Eve in $\mathcal{A}$.*

Proposition 1.7 *If X is a trap for Eve in $\mathcal{A}$, $\mathcal{A}_{|X}$ is a subarena of $\mathcal{A}$.*

Proposition 1.8 *Let X be a trap for Eve in $\mathcal{A}$, and σ be a strategy for Eve in $\mathcal{A}_{|X}$. For any state q in X and strategy τ for Adam in $\mathcal{A}$, the probability measure $\mathbb{P}_q^{\sigma,\tau}$ is the same in $\mathcal{A}_{|X}$ and in $\mathcal{A}$.*

1.2.2 Winning

Conditions

A *winning condition* $\mathcal{W}$ *on* $\mathcal{C}$ is a Borel subset of $\mathcal{C}^\infty$. A play ρ in an arena $\mathcal{A}$ on $\mathcal{C}$ is *winning for Eve* in the game $(\mathcal{A}, \mathcal{C})$ if $\chi(\rho) \in \mathcal{W}$, and *winning for Adam* otherwise.

A *regular condition* is a ω-regular language of $\mathcal{C}^\infty$. However, there is a tradition of using the classical acceptance conditions of regular automata directly on the play, as if $\mathcal{A}$ itself was an alternating automaton on a singleton alphabet. The resulting parity, Rabin, Streett, and Muller games are used in verification, logic, and automata [GTW02]:

- a *parity arena* $\mathcal{A}$ (resp. *parity arena of rank* k) is an arena on $\mathbb{N}$ (resp. $\{0, \ldots, k-1\}$), and the winner of a play ρ in the corresponding parity game depends on the smallest colour in the limit of ρ:

$$\rho \in \text{Parity} \iff \min\ \chi(\text{Inf}(\rho)) \text{ is even}$$

- a *Rabin arena* $\mathcal{A}$ *of rank* k is an arena on $\{-k, \ldots, -1, 1, \ldots, k\}$. An intuitive interpretation of the *Rabin condition of rank* k is to consider the negative integers as activators and the positive ones as inhibitors: a play ρ is winning for Eve if at least one activator $-i$ in $\text{Inf}(\rho)$ is not matched by the corresponding inhibitor i:

$$\rho \in \text{Rabin}(k) \iff \exists\ 1 \leq i \leq k, -i \in \chi(\text{Inf}(\rho)) \wedge i \notin \chi(\text{Inf}(\rho))$$

- a *Streett arena* $\mathcal{A}$ *of rank* k is an arena on $\{-k, \ldots, -1, 1, \ldots, k\}$. An intuitive interpretation of the *Streett condition of rank* k is to consider the negative integers as requests and the positive ones as responses: a play ρ is winning for Eve if each request $-i$ in $\text{Inf}(\rho)$ is matched by the corresponding response i:

$$\rho \in \text{Streett}(k) \iff \forall\ 1 \leq i \leq k, -i \in \chi(\text{Inf}(\rho)) \Rightarrow i \in \chi(\text{Inf}(\rho))$$

- a *Muller condition* $\mathcal{F}$ *on* $\mathcal{C}$ is a subset of $\mathcal{P}(\mathcal{C})$. The winner of a play ρ in the corresponding *Muller game on* $\mathcal{C}$ depends directly on its limit:

$$\rho \in \text{Muller}(\mathcal{F}, \mathcal{C}) \iff \chi(\text{Inf}(\rho)) \in \mathcal{F}\ .$$

In a game $\mathcal{G} = (\mathcal{A}, \mathcal{W})$, the *value of a state q under the strategies σ and τ*, denoted $v_{\sigma,\tau}(q)$, is the measure of $\mathcal{W}$ under $\mathbb{P}_q^{\sigma,\tau}$. The *value of a strategy σ for Eve* is the infimum of the $\{\sigma, \tau\}$-values:

$$v_\sigma(q) = \inf_\tau v_{\sigma,\tau}(q) \ .$$

Likewise, the *value of a strategy τ for Adam* is defined as a supremum:

$$v_\tau(q) = \sup_\sigma v_{\sigma,\tau}(q) \ .$$

Regions

De Alfaro and Henzinger define in [dAH00] several qualitative notions of *winning strategies* and *winning regions*, depending on the chances Eve gets to win:

Sure / Heroic: A strategy σ for Eve is *surely winning* (or sure) from a state q if and only if for any strategy τ for Adam, any play starting in q and consistent with σ and τ is winning for Eve. Dually, a strategy σ for Eve is *heroically winning* (or heroic) from a state q if and only if for any strategy τ for Adam, there is a play ρ starting in q, consistent with σ and τ, and winning for Eve. The corresponding *sure* and *heroic* regions are defined as follows:

$$\begin{aligned} \mathrm{Win}_E^{\mathcal{W},\forall}(\mathcal{A}) &= \{q \mid \exists\sigma, \forall\rho \in \Omega_q^\sigma, \rho \vdash \mathcal{W}\} \ ; \\ \mathrm{Win}_E^{\mathcal{W},\exists}(\mathcal{A}) &= \{q \mid \exists\sigma, \forall\tau, \exists\rho \in \Omega_q^{\sigma,\tau} \text{ such that } \rho \vdash \mathcal{W}\} \ . \end{aligned}$$

Almost-sure / Positive: A strategy σ for Eve is *almost-surely winning* (or almost-sure) from a state q if and only if for any strategy τ for Adam, the probability that the ensuing play is winning for Eve is one. Dually, a strategy σ for Eve is *positively winning* (or positive) from a state q if and only if for any strategy τ for Adam, the probability that the ensuing play is winning for Eve is positive. The corresponding *almost-sure* region and *positive* regions are defined as follows:

$$\begin{aligned} \mathrm{Win}_E^{\mathcal{W},1}(\mathcal{A}) &= \{q \mid \exists\sigma, \forall\tau, \mathbb{P}_q^{\sigma,\tau}(\mathcal{W}) = 1\} \ ; \\ \mathrm{Win}_E^{\mathcal{W},>0}(\mathcal{A}) &= \{q \mid \exists\sigma, \forall\tau, \mathbb{P}_q^{\sigma,\tau}(\mathcal{W}) > 0\} \ . \end{aligned}$$

Limit-one / Bounded: The *bounded region of Eve* is the set of states with positive value, and dually, the *limit-one region of Eve* is the set of states with value one:

$$\begin{aligned}\mathrm{Win}_E^{\mathcal{W},\sim 1}(\mathcal{A}) &= \{q \mid \forall \eta < 1, \exists \sigma, \forall \tau, v_{\sigma,\tau}(q) \geq \eta\} \ ; \\ \mathrm{Win}_E^{\mathcal{W},\gg 0}(\mathcal{A}) &= \{q \mid \exists \eta > 0, \exists \sigma, \forall \tau, v_{\sigma,\tau}(q) \geq \eta\} \ .\end{aligned}$$

These six notions of winning can also be defined for Adam in a straightforward way. By convention, we want the first superscript to correspond to the winning condition of the *game*, in which Adam is the opponent. For example, the almost-sure region of Adam in the game $\mathcal{G} = (\mathcal{A}, \mathcal{W})$ is denoted $\mathrm{Win}_A^{\mathcal{W},1}(\mathcal{A})$ and refers to the region where Adam can guarantee $\neg\mathcal{W}$ with probability one.

The two following propositions are direct consequences of the definitions:

Proposition 1.9 *Let $\mathcal{G} = (\mathcal{A}, \mathcal{W})$ be a $2\frac{1}{2}$-player games. We have:*

$$\begin{array}{ccccc} \mathrm{Win}_E^{\mathcal{W},\forall}(\mathcal{A}) & \subseteq & \mathrm{Win}_E^{\mathcal{W},1}(\mathcal{A}) & \subseteq & \mathrm{Win}_E^{\mathcal{W},\sim 1}(\mathcal{A}) \\ & & & & \cap| \\ \mathrm{Win}_E^{\mathcal{W},\exists}(\mathcal{A}) & \supseteq & \mathrm{Win}_E^{\mathcal{W},>0}(\mathcal{A}) & \supseteq & \mathrm{Win}_E^{\mathcal{W},\gg 0}(\mathcal{A}) \ . \end{array}$$

Proposition 1.10 *Let $\mathcal{G} = (\mathcal{A}, \mathcal{W})$ be a $2\frac{1}{2}$-player games. We have:*

$$\begin{aligned}\mathrm{Win}_E^{\mathcal{W},\forall}(\mathcal{A}) \ \cap \ \mathrm{Win}_A^{\mathcal{W},\exists}(\mathcal{A}) = \emptyset \\ \mathrm{Win}_E^{\mathcal{W},1}(\mathcal{A}) \ \cap \ \mathrm{Win}_A^{\mathcal{W},>0}(\mathcal{A}) = \emptyset \\ \mathrm{Win}_E^{\mathcal{W},\sim 1}(\mathcal{A}) \ \cap \ \mathrm{Win}_A^{\mathcal{W},\gg 0}(\mathcal{A}) = \emptyset\end{aligned}$$

1.3 Usual problems and contributions

1.3.1 Problems

Determinacy and existence of values

A natural question is whether the disjunctions of Proposition 1.10 partition $\mathcal{Q}$ or not. The *sure determinacy of 2-player Borel*[1] *games* [Mar75], transposed to $2\frac{1}{2}$-player games by replacing the random states with states of Adam, yields $\mathrm{Win}_E^{\mathcal{W},\forall}(\mathcal{A}) \cup \mathrm{Win}_A^{\mathcal{W},\exists}(\mathcal{A}) = \mathcal{Q}$ for any game $\mathcal{G} = (\mathcal{A}, \mathcal{W})$. In the case

[1]The existence of non-determined 2-player games relies on the axiom of choice.

of $2\frac{1}{2}$-player games, the *quantitative determinacy of Blackwell games* [Mar98] states that the *value of a state* q can indifferently be defined as the supremum of the σ-values, or the infimum of the τ-values:

$$\mathbf{v}(q) = \sup_{\sigma} \inf_{\tau} v_{\sigma,\tau}(q) = \inf_{\tau} \sup_{\sigma} v_{\sigma,\tau}(q) \ .$$

It follows immediately that $\mathrm{Win}_E^{\mathcal{W},\sim 1}(\mathcal{A}) \cup \mathrm{Win}_A^{\mathcal{W},\gg 0}(\mathcal{A}) = \mathcal{Q}$. However, there is no such general answer to the problem of "qualitative determinacy":

$$\mathrm{Win}_E^{\mathcal{W},1}(\mathcal{A}) \cup \mathrm{Win}_A^{\mathcal{W},>0}(\mathcal{A}) \stackrel{?}{=} \mathcal{Q} \ .$$

Qualitative and quantitative problems

In [CJH03], the authors also classify the different problems on games:

- *Qualitative problems* depend on the winning regions of the players — for all six notions of "winning". A "qualitative-complete" problem on a game $\mathcal{G} = (\mathcal{A}, \mathcal{W})$ consists in deciding, for any given state q, player P, and notion of winning ?, whether q belongs to $\mathrm{Win}_P^{\mathcal{W},?}(\mathcal{A})$.
- *Quantitative problems* on the other hand, depend on the values of the states —and thus are interesting only in $2\frac{1}{2}$-player games. A "quantitative-complete" problem consists in computing the value $\mathbf{v}(q)$ of any given state q.

The decidability and complexity of qualitative and quantitative problems generate a major part of the articles in graph games theory.

Complexity of the winning strategies

Another question ponders the nature of the winning strategies, in terms of randomisation and memory. This is especially useful from a verification point of view, as the strategies represent possible implementations of controllers, whose cost is often more critical than the specification costs. In automata theory, the existence of positional strategies for specific winning condition has been an invaluable tool for several problems.

1.3.2 Contributions

Solving reachability games

We present two new algorithms computing optimal strategies in $2\frac{1}{2}$-player reachability games. They are based on the existence of optimal *permutation strategies*, a sub-class of positional strategies derived from permutations of the random states. As our algorithms never consider the same permutation twice, their worst-case complexity mostly depends on the number of such permutations, making the solution of $2\frac{1}{2}$-player reachability games fixed-parameter tractable, when the parameter is the number of random states: this is orthogonal to the complexity of the strategy-based algorithms, which rather depends on the number of player's states.

The first algorithm, the *permutation-enumeration algorithm* is a simple exhaustive search. Its complexity is thus exponential, but it avoids the use of linear programming. The second one, the *permutation-improvement algorithm*, emulates the heuristic of the classical *strategy improvement algorithm* [HK66] in order to avoid an exhaustive search.

Another asset of our algorithms is that they do not rely on the expensive *stopping hypothesis* [Con92]: this allows us, in the next chapter, to extend them to the much broader case of prefix-independent games.

Prefix-independent winning conditions

In prefix-independent games, the winner of a play depends only on its limit, and not on finite prefixes. We show that in these games, the positive and bounded regions, as well as the limit-one and almost-sure regions, are equal. We prove then their optimal determinacy, and provide an algorithm computing the values of any prefix-independent game with $|\mathcal{Q}_R|!$ calls to a qualitative algorithm. Alternatively, a single non-deterministic guess can replace the multiple iterations. It follows from our proof of correctness that optimal strategies are no more complex than almost-sure strategies.

This generalises and extends several results on the winning regions of regular [dAHK98, dAH00] and prefix-independent games [Cha07a]. The complexity of our general algorithm is better or on a par with the complexity of several known algorithms for special cases [CJH04, CdAH05, CHH08].

Muller games

We present a polynomial algorithm for the qualitative problems of *explicit Muller games*. It follows then from our results of the former chapter that the quantitative problems of explicit Muller games belongs in NP and co-NP. The only algorithm previously known for these games was the all-purpose PSPACE algorithm for Muller games [McN93, NRY96].

Our next achievement is the computation of tights bounds in memory for optimal randomised strategies in Muller games. The comparison with similar results for pure strategies [DJW97] allows us to ascertain the differences between the two models of strategies in this aspect.

Former results on randomised strategies provided upper bounds for randomised games [CdAH04, Cha07b], but to the best of the author's knowledge, no lower bounds.

Finitary games

Finitary conditions [AH98] supplement regular conditions with bounds on the time spent between a "bad" event and a subsequent "good" event which compensate for it. Chatterjee and Henzinger studied 2-player games with finitary parity and Streett conditions and proposed algorithms computing the winning regions of the players [CH06a].

We extend this study to the case of $2\frac{1}{2}$-player games and provide faster algorithms for both kinds of games. In particular, we have shown that the qualitative finitary parity games can be solved in polynomial time —here also, the results of Chapter 3 yield directly NP and co-NP algorithms for quantitative problems. We also show that finitary Streett games can be reduced in polynomial time to Request-Response games [WHT03].

Chapter 2

Reachability games

"Consistency is the last refuge of the unimaginative."

Oscar Wilde

One of the simplest, and yet most useful, winning conditions is the *reachability condition*: there is a distinguished *target state* in the arena, denoted by $\circledcirc$, and Eve's objective is to ensure that the token reaches it at some point during the play.

In this chapter, we consider the problems of computing the values and optimal strategies in such games. Figure 2.1 presents an example of $2\frac{1}{2}$-player reachability game, that we use throughout the chapter to demonstrate notions and intuitions.

Section 2.1 introduces some general concepts on $2\frac{1}{2}$-player reachability games, as well as the *strategy improvement algorithm*. We present in Section 2.2 a new approach to the computation of values and optimal strategies, based on permutations over the random states. Section 2.3 exposes then an improvement heuristic for this "permutation algorithm".

2.1 First notions

This section is devoted to the fundamental notions that we use throughtout the chapter in order to deal with the values of $2\frac{1}{2}$-player reachability games. It includes a large part of the state of the art in the domain, which can also be found in a more detailled way in the survey of Condon [Con93].

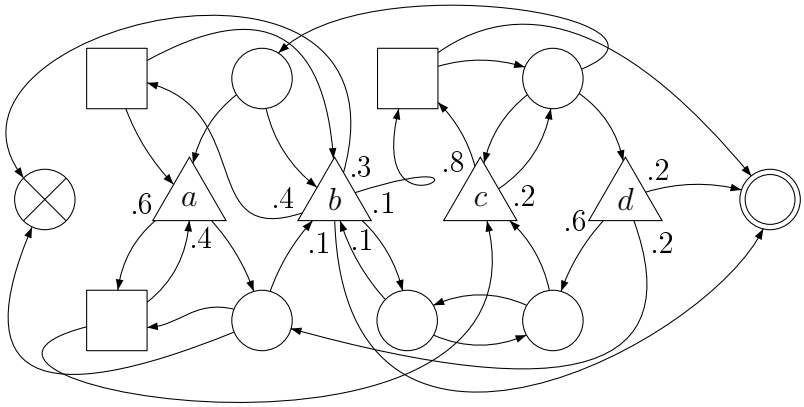


Figure 2.1: A $2\frac{1}{2}$-player reachability game

However, there is a lot more work on $2\frac{1}{2}$-player reachability games (see, for example, [Hal07] on the use of randomised algorithms) that we *don't* describe here, because it bears too few relations with our own results.

We first describe two special class of reachability games (2.1.1), and then present some fundamental results about the values (2.1.2). A description of the *strategy improvement algorithm* concludes the section (2.1.3).

2.1.1 Normalised and stopping games

Qualitative problems are easy to solve on $2\frac{1}{2}$-player reachability games, and derive directly from the notion of attractor:

$$\mathrm{Win}_E^{\mathrm{Reach}(\circledcirc),>0}(\mathcal{A}) = \mathrm{Win}_E^{\mathrm{Reach}(\circledcirc),\gg 0}(\mathcal{A}) = \mathrm{Attr}_E(\{\circledcirc\},\mathcal{A})$$

$$\mathrm{Win}_A^{\mathrm{Reach}(\circledcirc),\sim 1}(\mathcal{A}) = \mathrm{Win}_A^{\mathrm{Reach}(\circledcirc),1}(\mathcal{A}) = \mathcal{Q} \setminus \mathrm{Win}_E^{\mathrm{Reach}(\circledcirc),>0}(\mathcal{A})$$

$$\mathrm{Win}_E^{\mathrm{Reach}(\circledcirc),\sim 1}(\mathcal{A}) = \mathrm{Win}_E^{\mathrm{Reach}(\circledcirc),1}(\mathcal{A}) = \mathcal{Q} \setminus \mathrm{Attr}_A(\mathrm{Win}_A^{\mathrm{Reach}(\circledcirc),1}(\mathcal{A}),(\mathcal{A}))$$

$$\mathrm{Win}_A^{\mathrm{Reach}(\circledcirc),>0}(\mathcal{A}) = \mathrm{Win}_A^{\mathrm{Reach}(\circledcirc),\gg 0}(\mathcal{A}) = \mathcal{Q} \setminus \mathrm{Win}_E^{\mathrm{Reach}(\circledcirc),1}(\mathcal{A})$$

The class of *normalised games* is the class of games where the these qualitative questions have trivial answer: apart from the sink $\otimes$ and the target $\circledcirc$, no state has value zero or one. This class is mainly of aesthetic significance, as it simplifies the proofs of algorithms and theorems which would still be correct, *mutatis mutandis*, on general $2\frac{1}{2}$-player reachability

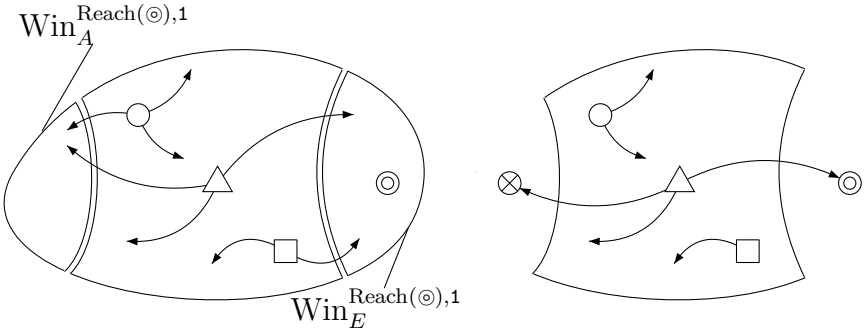
games. However, there are some cases where normalised games are a much cheaper alternative to stopping games.

Definition 2.1 *A $2\frac{1}{2}$-player reachability game $\mathcal{G} = (\mathcal{A}, \mathrm{Reach}(\circledcirc))$ is normalised if and only if the only state with value one is the target $\circledcirc$, and there is only one state with value zero, which we denote by $\otimes$.*

It is easy to transform any $2\frac{1}{2}$-player reachability game $\mathcal{G} = (\mathcal{A}, \mathrm{Reach}(\circledcirc))$ into a normalised game $\mathfrak{G} = (\mathfrak{A}, \mathrm{Reach}(\circledcirc))$:

- the region $\mathrm{Win}_E^{\mathrm{Reach}(\circledcirc),1}(\mathcal{A})$ is merged into a single state, which is the target of $\mathfrak{G}$;
- the region $\mathrm{Win}_A^{\mathrm{Reach}(\circledcirc),1}(\mathcal{A})$ is merged into a single state, which is the sink of $\mathfrak{G}$.

This transformation is represented on Figure 2.2.



(a) Original game $\mathcal{G}$ (b) Normalised game $\mathfrak{G}$

Figure 2.2: Reachability game normalisation

There is another incentive to use normalised games: as the reduction is very cheap (linear), and the resulting game is smaller in general than the original one, it is a good idea in practice to normalise a game before running any quantitative algorithm on it.

2.1.1.1 Stopping games

The stopping hypothesis is less benign, as stopping games really have stronger properties, inherited from the original model of Shapley: in the games of [Sha53], the token has a positive probability to stop in each visited state. As a result, the plays are finite with probability one. In our model, we call *stopping games* the games which share this property:

Definition 2.2 *A $2\frac{1}{2}$-player reachability game is stopping if and only if, for any strategies σ of Eve and τ of Adam, the probability that the token eventually reaches $\otimes$ or $\circledcirc$ is one:*

$$\forall \sigma, \tau, q, \mathbb{P}_q^{\sigma,\tau}(\mathrm{Reach}(\otimes) \vee \mathrm{Reach}(\circledcirc)) = 1$$

The point of these games is that they are symmetric: avoiding $\otimes$ and reaching $\circledcirc$ amount to the same thing, which is not the case in general. As a consequence, the following intuitive properties of each player's strategies hold for both in stopping games.

Proposition 2.3 *Let $\mathcal{G} = (\mathcal{A}, \mathrm{Reach}(\circledcirc))$ be a $2\frac{1}{2}$-player reachability game and τ be a positional strategy for Adam such that:*

$$\forall q \in \mathcal{Q}_A, \mathbf{v}(\tau(q)) = \mathbf{v}(q) \ .$$

Then τ is optimal. This is not true in general for Eve's strategies.

Proposition 2.4 *Let $\mathcal{G} = (\mathcal{A}, \mathrm{Reach}(\circledcirc))$ be a $2\frac{1}{2}$-player reachability game and σ and τ be positional strategies for Eve and Adam such that:*

$$\forall q \in \mathcal{Q}_E, v_{\sigma,\tau}(\sigma(q)) = \max\{v_{\sigma,\tau}(s) \mid (q,s) \in \mathcal{T}\} \ .$$

Then σ is an optimal counter-strategy to τ. This is not true in general for Adam's strategies.

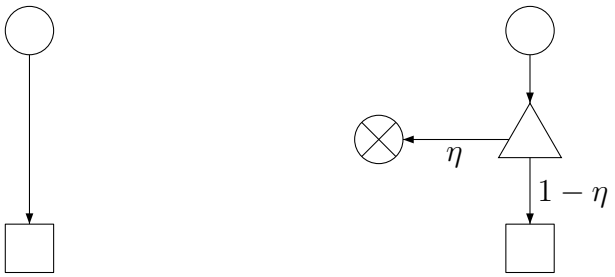
Condon showed in [Con92] the existence of a polynomial reduction which preserves optimal strategies and threshold regions:

Proposition 2.5 ([Con92]) *Let $\mathcal{G} = (\mathcal{A}, \mathrm{Reach}(\circledcirc))$ be a $2\frac{1}{2}$-player reachability game. There is a stopping reachability game $\mathfrak{G} = (\mathfrak{A}, \mathrm{Reach}(\circledcirc))$ such that:*

- *$\mathfrak{Q}_E = \mathcal{Q}_E$, $\mathfrak{Q}_A = \mathcal{Q}_A$, and $\mathfrak{Q}_R \supset \mathcal{Q}_R$;*

- *to each transition in* $\mathcal{T}(q)$ *corresponds a transition in* $\mathfrak{T}(q)$*;*
- *the size of* $\mathfrak{G}$ *is quadratic in the size of* $\mathcal{G}$*;*
- $\forall q \in \mathcal{Q}, \mathbf{v}(q) > \frac{1}{2} \iff \mathfrak{v}(q) > \frac{1}{2}$*;*
- *if* σ *(resp.* τ*) is an optimal strategy for Eve (resp. Adam) in* $\mathfrak{G}$*, then it is also optimal in* $\mathcal{G}$*.*

Figure 2.3 shows the idea of the reduction: in each transition, there is a small probability η that the token goes directly to the sink $\otimes$ instead of its intended destination. For an small enough η, the optimal strategies of the reduced game are also optimal in the original game (the converse is not true in general). However, the binary representation of a suitable η is linear in the size of $\mathcal{G}$, so the reduction involves a quadratic blow-up in size.



(a) Original game $\mathcal{G}$ (b) Stopping game $\mathfrak{G}$

Figure 2.3: Reduction to stopping games

Remark 2.6 *In [Con92], Condon considers games where the random states have only two successors, with equal probabilities. Thus, her reduction involves several successive random states instead of one, but it still involves a quadratic blow-up.*

Using Proposition 2.5, it is possible to consider only stopping games, and derive general theorems about $2\frac{1}{2}$-player reachability games. However, in this chapter, we try to minimise the use of this option. Our reasons are twofold. First, altough polynomial, the reduction of Figure 2.3 is quite expensive in

practice as the precision grows; it is even not clear that it can be adapted if the expected precision is not known beforehand, or the probabilities are not rational. Second, the reduction to stopping games is not very intuitive, especially when it comes to a generalisation for infinite games, as we do in Chapter 3.

2.1.2 Equations and Positional strategies

A simple technique of *strategy translation* yields the following system of equations on the values of a game $\mathcal{G} = (\mathcal{A}, \mathrm{Reach}(\circledcirc))$:

$$\begin{aligned} \forall q \in \mathcal{Q}_E, \mathbf{v}(q) &= \max_{s \in E(q)} (\mathbf{v}(s)) \\ \forall q \in \mathcal{Q}_A, \mathbf{v}(q) &= \min_{s \in E(q)} (\mathbf{v}(s)) \\ \forall q \in \mathcal{Q}_R, \mathbf{v}(q) &= \sum_{s \in E(q)} \delta(q)(s) \cdot \mathbf{v}(s) \\ \mathbf{v}(\circledcirc) &= 1 \\ \mathbf{v}(\otimes) &= 0 \end{aligned} \tag{2.1}$$

In the case of stopping games (but *not* in general games), there is only one solution to this system:

Proposition 2.7 *In a stopping $2\frac{1}{2}$-player reachability game, the values are the only solution to (2.1). Furthermore, if the whole transition function is described by rationals on n bits, the values are rationals which can be written on $2n$ bits.*

This proposition suggests immediately an algorithm computing the values of a stopping $2\frac{1}{2}$-player reachability game: check exhaustively (or guess non-deterministically) the values of the game, and check that they are a solution to (2.1). The two following theorems, about the complexity of the value problems, are a direct consequence:

Theorem 2.8 *Quantitative decision problems about $2\frac{1}{2}$-player reachability games belong to* NP *and co-*NP.

Theorem 2.9 *Let $\mathcal{G} = (\mathcal{A}, \mathrm{Reach}(\circledcirc))$ be a $2\frac{1}{2}$-player reachability game. The values of $\mathcal{G}$ can be computed in time $O(4^{2 \cdot (|\mathcal{T}| + |\delta|)})$.*

Another fundamental result which follows immediately from (2.1) is the existence of positional values in $2\frac{1}{2}$-player reachability games:

Theorem 2.10 *In a $2\frac{1}{2}$-player reachability game, both players have positional optimal strategies.*

Proof. In a stopping $2\frac{1}{2}$-player reachability game, a positional strategy such that:

- $\forall q \in \mathcal{Q}_E, \mathbf{v}(\sigma(q)) = \mathbf{v}(q)$ if σ is a strategy for Eve, or
- $\forall q \in \mathcal{Q}_A, \mathbf{v}(\tau(q)) = \mathbf{v}(q)$ if τ is a strategy for Adam.

is optimal. By Proposition 2.5, there are also positional strategies in general $2\frac{1}{2}$-player reachability games. □

This allow us to consider only positional strategies, which are much easier to handle. So, in the remainder of this chapter, whenever we mention a "strategy", we mean a positional strategy.

The values of a pair of strategies σ and τ are solutions to the following system of equations:

$$\begin{aligned} \forall q \in V_E, v_{\sigma,\tau}(q) &= v_{\sigma,\tau}(\sigma(q)) \\ \forall q \in V_A, v_{\sigma,\tau}(q) &= v_{\sigma,\tau}(\tau(q)) \\ \forall q \in V_R, v_{\sigma,\tau}(q) &= \sum_{s \in E(q)} \delta(q)(s) \cdot v_{\sigma,\tau}(s) \\ v_{\sigma,\tau}(\circledcirc) &= 1 \\ v_{\sigma,\tau}(\otimes) &= 0 \end{aligned} \tag{2.2}$$

Once again, the solution to (2.2) is not necessarily unique, unless the game is stopping. A useful property of positional strategies is that optimal strategies can be characterised by a notion of *stability*:

Definition 2.11 *Two strategies σ and τ are co-stable if and only if:*

- $\forall q \in \mathcal{Q}_E, v_{\sigma,\tau}(\sigma(q)) = \max\{v_{\sigma,\tau}(s) \mid (q,s) \in \mathcal{T}\}$
- $\forall q \in \mathcal{Q}_A, v_{\sigma,\tau}(\tau(q)) = \min\{v_{\sigma,\tau}(s) \mid (q,s) \in \mathcal{T}\}$

Proposition 2.12 *Let $\mathcal{G} = (\mathcal{A}, \mathrm{Reach}(\circledcirc))$ be a $2\frac{1}{2}$-player stopping reachability game, and σ and τ be two strategies for Eve and Adam. Then, (i) and (ii) are equivalent:*

(i) σ and τ are co-stable

(ii) σ is an optimal strategy for Eve, and τ is an optimal strategy for Adam.

Proof. Proposition 2.12 follows directly from Proposition 2.7, as two strategies σ and τ are co-stable if and only if $v_{\sigma,\tau}$ is a solution to (2.1). □

We can thus search exhaustively for optimal strategies, instead of searching directly the optimal values. The complexity of the resulting algorithm — Algorithm 2.1 — is much better: $O(|\mathcal{Q}|^{|\mathcal{Q}|})$.

```
   Input: a game G = (A, Reach(◎))
   Output: optimal strategies for both players
1  forall σ ∈ Σ do
2  |  forall τ ∈ T do
3  |  |  if σ and τ are co-stable then
4  |  |  |  return σ, τ
5  |  |  end
6  |  end
7  end
```

Algorithm 2.1: Strategy enumeration for reachability games

Remark 2.13 *Proposition 2.12 does not hold when the game is not stopping games, so Algorithm 2.1 can return incorrect results in this case . For this same reason, the strategy algorithms of the next sections usually suppose that the input games are stopping. We show, however, that a careful adaptation allows us to cancel this hypothesis.*

2.1.3 Strategy improvement

In practice, one never uses Algorithm 2.1. The two static **forall** loops can be replaced by more efficient dynamic strategy improvement schemes [HK66].

The idea is to use the values of a strategy in order to compute a *better* one, unless the current strategy is already optimal.

We consider first the case of $1\frac{1}{2}$-player games where $\mathcal{Q}_E = \emptyset$[1]. Strategy improvement algorithms for $1\frac{1}{2}$-player games are first mentioned in [How60]. Notice that, in $1\frac{1}{2}$-player games, normalised implies stopping:

Proposition 2.14 *Let $\mathcal{G}$ be a normalised $1\frac{1}{2}$-player game. Then $\mathcal{G}$ is stopping.*

Proof. As $\mathcal{G}$ is normalised, the attractor of Eve to the target $\circledcirc$ is $\mathcal{Q} \setminus \{\otimes\}$. Eve has only one strategy which is thus the attractor strategy to $\circledcirc$. So, by Proposition 1.5, $\forall \tau, q, \mathbb{P}_q^\tau(\mathrm{Reach}(\otimes) \vee \mathrm{Reach}(\circledcirc)) = 1$. Proposition 2.14 follows. □

At the core of the strategy improvement algorithm is the concept of switching an unstable strategy:

Definition 2.15 *Let $\mathcal{G} = (\mathcal{A}, \mathrm{Reach}(\circledcirc))$ be a $1\frac{1}{2}$-player game such that $\mathcal{Q}_E = \emptyset$, and τ be a strategy for Adam. The switched strategy of τ is the strategy θ defined as:*

- *if $\forall s \in \mathcal{T}(q), v_\tau(s) \geq v_\tau(\tau(q))$, then $\theta(q) = \tau(q)$;*
- *otherwise, $\theta(q)$ is chosen such that $\forall s \in \mathcal{T}(q), v_\tau(s) \geq v_\tau(\theta(q))$.*

The algorithm, computing Adam's optimal strategy and described as Algorithm 2.2, consists in repeatedly switching the current strategy, until it is stable.

Input: A $1\frac{1}{2}$-player safety game $\mathcal{G}$
Output: Optimal strategy for Adam
1 **repeat**
2 | `switch` τ
3 **until** τ `is stable`
4 **return** $\tau, \mathbf{v}$

Algorithm 2.2: Strategy Improvement for $1\frac{1}{2}$-player safety games

Correctness is ensured by Proposition 2.12, and termination by Proposition 2.16:

[1]The concepts work mostly in the same way when $\mathcal{Q}_A = \emptyset$, albeit with different proofs.

Proposition 2.16 *Let $\mathcal{G}$ be a normalised $1\frac{1}{2}$-player safety game, and τ be a strategy for Adam. Then, either τ is stable, or the strategy θ obtained by switching τ is such that $v_\theta < v_\tau$.*

The notion of switching strategies described in Definition 2.15 needs to be adapted in order to be used in $2\frac{1}{2}$-player games. In this context,one switches a strategy *with respect to another*:

Definition 2.17 *Let $\mathcal{G} = (\mathcal{A}, \mathrm{Reach}(\circledcirc))$ be a $2\frac{1}{2}$-player reachability game, and σ and τ be strategies for Eve and Adam. The switched strategy of σ with respect to τ is the strategy ς is defined as:*

- *if $\forall s \in \mathcal{T}(q), v_{\sigma,\tau}(s) \leq v_{\sigma,\tau}(\sigma(q))$, then $\varsigma(q) = \sigma(q)$;*
- *otherwise, $\varsigma(q)$ is chosen such that $\forall s \in \mathcal{T}(q), v_{\sigma,\tau}(s) \leq v_{\sigma,\tau}(\varsigma(q))$.*

The switched strategy of τ with respect to σ is defined symmetrically. It corresponds to the switched strategy of τ in the $1\frac{1}{2}$-player safety game $\mathcal{G}^\sigma$.

In Algorithm 2.3, improving a strategy σ, consists in computing an optimal counter-strategy τ, and then switching σ with respect to τ. The run stops only when the strategies are co-stable.

Input: The game $\mathcal{G}$
Output: Optimal strategies and values

```
1 choose σ as an attractor strategy of Eve to ⊚
2 repeat
3 |  compute an optimal counter-strategy τ to σ
4 |  switch σ with respect to τ
5 until σ and τ are co-stable
6 return σ, τ
```

Algorithm 2.3: Strategy Improvement for $2\frac{1}{2}$-player reachability games

Again, the impossibility of an infinite run is proved through a notion of progress:

Proposition 2.18 *Let $\mathcal{G} = (\mathcal{A}, \mathrm{Reach}(\circledcirc))$ be a $2\frac{1}{2}$-player reachability game, σ be a positional attractor strategy to $\circledcirc$ for Eve, τ be an optimal counter-strategy to σ, and ς be the switched strategy of σ with respect to τ. Then, either σ and τ are optimal, or, for any strategy θ, $v_{\varsigma,\theta} > v_{\sigma,\tau}$.*

There are two remarks to be made about Algorithm 2.3. The first one is that we can remove the stopping hypothesis of [Con93] by using normalised games. There is no stopping hypothesis in Proposition 2.18. By Proposition 2.14, normalisation is enough for the improvement of Adam's strategy. However, it is not true in general that if $\mathcal{G}$ is normalised, then $\mathcal{G}^\sigma$ is also normalised —while such a property holds for stopping games. If σ is an attractor strategy to $\circledcirc$, though, then $\mathcal{G}^\sigma$ is normalised. Proposition 2.18 guarantees that σ remains an attractor strategy to $\circledcirc$ for the whole run. The second remark consists in precisions about the improvement steps of both players:

- The improvement of Eve's strategy in line 4 consists in a single switching. It is not the computation of an optimal counter-strategy to Adam's current strategy, nor should it be, as it leads to infinite loops.
- Symmetrically, it is not enough to switch Adam's strategy only once instead of computing an optimal counter-strategy in line 3. This also leads to infinite runs.

These two examples, as well as several other unsound variations [Mar07] of Algorithm 2.3, are presented in [Con93].

In terms of theoretical complexity, Algorithms 2.2 and 2.3 do not fare much better than Algorithm 2.1. Progress ensures that any given pair of strategy cannot be considered more than once, and no more. However, in practice, both algorithms run very fast, to the point that they are widely conjectured to be polynomial:

Conjecture 2.19 *Algorithm 2.2 runs in polynomial time on any normalised* $1\frac{1}{2}$*-player safety game.*

Conjecture 2.20 *Algorithm 2.3 runs in polynomial time on any normalised* $2\frac{1}{2}$*-player reachability game game.*

Note that the strategy improvement algorithm for 2-player parity games described in [VJ00], which is derived from Algorithm 2.3, runs in polynomial time on 1-player games [Jur07]. However, even if Conjecture 2.19 does not hold, one can get a better complexity for Algorithm 2.3 by using linear programming in line 3 instead of Algorithm 2.2. As Derman showed in [Der62] the optimal values of a $1\frac{1}{2}$-player safety game are the solution of the linear program presented in Algorithm 2.4.

Input: a game $\mathcal{G} = (\mathcal{A}, \text{Reach}(\circledcirc))$ such that $\mathcal{Q}_E = \emptyset$
Output: Values
1 `minimize` $\sum_{q \in \mathcal{Q}} v(q)$ `subject to the constraints:`
2 $v(q) \leq v(s)$ `if` $q \in \mathcal{Q}_A$ `and` $s \in \mathcal{T}(q)$
3 $v(q) = \sum_{s \in \mathcal{T}(q)} \delta(q)(s) \cdot v(q)$ `if` $q \in \mathcal{Q}_R$
4 $v(q) \geq 0$ `if` $q \in \mathcal{Q}$
5 $v(\otimes) = 0$
6 $v(\circledcirc) = 1$
7 **return** v

Algorithm 2.4: Linear programming for $1\frac{1}{2}$-player games

Linear programs can be solved in polynomial time [Kha79, Ren88], resulting in an overall complexity for Algorithm 2.3 that is exponential only in $\mathcal{Q}_E$ or $\mathcal{Q}_A$ instead of both.

2.2 Permutation Algorithm

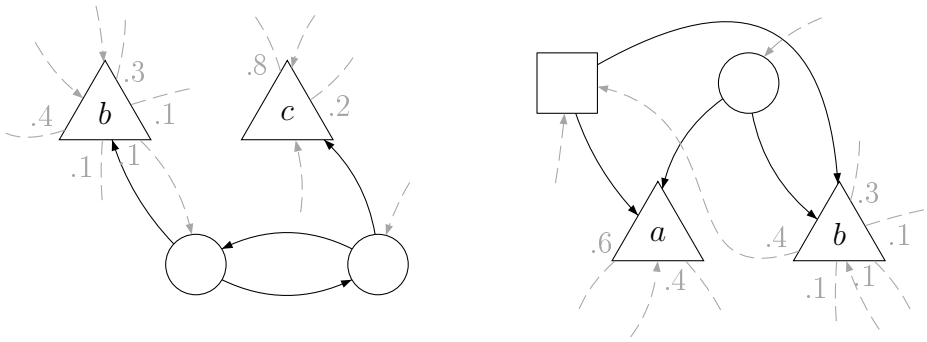
In a joint work with Hugo Gimbert [GH08, GH09], we propose a new algorithm computing the values of $2\frac{1}{2}$-player reachability games. Its principle is to check exhaustively a special set of pairs of strategies, among which there is at least one pair of optimal strategies.

The underlying intuition is that the only meaningful events in a play are the visits to random states. Between two visits, the players strive to impose which state will be visited next, and the result of their interaction can easily be predicted. In particular:

- only the next random state matters, *not* the current one;
- there is no reason that Eve and Adam should *ever* agree on a choice.

Two occurences of such situations, excerpted from the game of Figure 2.1, are illustrated on Figure 2.4.

In Figure 2.4(a), Eve can choose between the two random states (refusing to choose is not consistent with the reachability objective). Why should she choose b in one state and c in the other ? The two strategies "always go to b" and "always go to c" are the only relevant ones.



(a) Self consistency (b) Consistency between Eve and Adam

Figure 2.4: Case for Consistency

In Figure 2.4(b), we consider relationships between the two players' strategies. From their respective states ○ and □, Eve and Adam can send the token to either a or b. Why should they choose the same state ? Here, only the cases where Eve prefers one and Adam the other are relevant.

These intuitions —including, but not limited to, the two cases of Figure 2.4— are realized by pairs of strategies corresponding to a permutation of the random states. We define permutation-based strategies and regions (2.2.1), as well as the notions of liveness and self-consistency (2.2.2). Our algorithm is an exhausitve search for a live and self-consistent permuation: there is always such a permutation, and the corresponding strategies are optimal (2.2.3). We study then its complexity, and present a class of reachability games where the values can be computed in polynomial time (2.2.4).

2.2.1 Strategies and regions

In order to effect our intuitions [Mun07], we introduce several permutation-based concepts. First, whenever we mention a *permutation* π, we mean a permutation over the k random states, such that $\{\pi_1, \ldots, \pi_k\} = \mathcal{Q}_R$. Such a permutation represents a "preference order" over the random states: if Eve is given a choice between two random states π_i and π_j with $i > j$, then her "π-strategy" sends the token to π_i. Symmetricaly, in the same situation, Adam's π-strategy sends the token to π_j. For this reason, the target and sink states can often be considered as random states in permutation-based

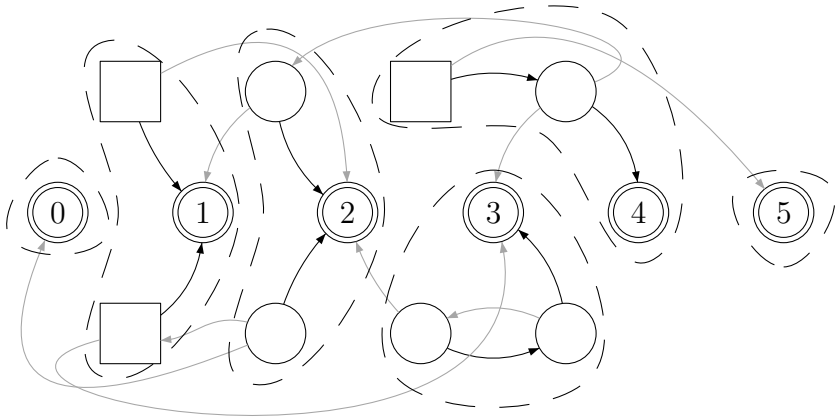


Figure 2.5: The single-reward game derived from Figure 2.1

concepts, with the implicit assumption that they are respectively the greatest and lowest states: $\pi_{k+1} = \circledcirc$ and $\pi_0 = \otimes$.

A intuitive way to understand the permutation-based concepts is to consider a 2-player game, where the game stops after a finite number of steps and Adam pays a reward to Eve at the end:

1. if the token reaches a state π_i, Adam pays i coins;

2. if there is a loop in the path, Adam has nothing to pay.

The π-regions are the value regions of this game, and the π-strategies are the corresponding optimal strategies. For example, if we use the permutation $\pi = abcd$ in the game of Figure 2.1, we get the game, regions, and strategies represented in Figure 2.5.

In order to formalise these concepts, we define an "attractor-like" deterministic construction: the *deterministic attractor for Eve to a region X in the arena $\mathcal{A}$*, denoted $\mathrm{Det}_E(X, \mathcal{A})$ is the set of states from where Eve can ensure that the token will (1) reach X (2) not cross a random state before it reaches X:

Definition 2.21 *The deterministic attractor of Eve to the set X, denoted*

$\mathrm{Det}_E(X, \mathcal{A})$ *is computed recursively:*

$$\begin{aligned} X^0 &= X \\ X^{i+1} &= X^i \cup \{q \in \mathcal{Q}_E \mid \exists s \in X^i, (q,s) \in \mathcal{T}\} \\ &\quad \cup \{q \in \mathcal{Q}_A \mid \forall s \in \mathcal{Q}, (q,s) \in \mathcal{T} \Rightarrow s \in X^i\} \end{aligned}$$

A random state belongs to $\mathrm{Det}_E(X, \mathcal{A})$ *if and only if it belongs to* X*. The dual notion of deterministic trap for Eve is a region from which Eve cannot escape, except through a random state.*

The π*-regions* are defined as embedded deterministic attractors to the random states, ordered by π: $W_\pi[k+1] = \{\circledcirc\}$, $\forall 1 \leq i \leq k, W_\pi[i] = \mathrm{Det}_E(\{\pi_i, \pi_{i+1}, \ldots, \pi_k\}, \mathcal{A}) \setminus W_\pi[i+1]$, and $W_\pi[0] = \{\otimes\}$. The π-regions constitutes a partition of the states, so we denote by $\pi(q)$ the unique integer i such that $q \in W_\pi[i]$ —in particular, $\pi(\pi_i) = i$. The computation of the π-regions of the game $\mathcal{G}$) is described as Procedure `Regions`$(\mathcal{G}, \pi)$.

Input: A $2\frac{1}{2}$-player reachability game $\mathcal{G}$ and a permutation π
Output: The π-regions of $\mathcal{G}$
1 $W[k+1] \leftarrow \{\circledcirc\}$
2 $W[0] \leftarrow \{\otimes\}$
3 **for** $(i = 1, i \leq k, i++)$ **do**
4 $\quad W[i] \leftarrow \mathrm{Det}_E(\{\pi_i, \ldots, \pi_k\} \cup \{\circledcirc\}, \mathcal{A}) \setminus W[i+1]$
5 **end**
6 **return** W

Procedure `Regions`$(\mathcal{G}, \pi)$

The π-strategies are the natural attracting and trapping strategies following from Definition 2.21, which enforce Propositions 2.22 and 2.23:

Proposition 2.22 *If the token starts in a state of* $W_\pi[i]$ *and Eve plays* σ_π*, then the token surely reaches a random state, and the first random state* π_j *that the token reaches is such that* $j \geq i$*.*

Proposition 2.23 *If the token starts in a state of* $W_\pi[i]$*, Adam plays* τ_π*, and the token reaches a random state, then the first random state* π_j *that the token reaches is such that* $j \leq i$*.*

A consequence of Propositions 2.22 and 2.23 is Proposition 2.24:

Proposition 2.24 *Let $\mathcal{G} = (\mathcal{A}, \mathrm{Reach}(\circledcirc))$ be a $2\frac{1}{2}$-player reachability game and π be a permutation. For any state $q \in W_\pi[i]$, we have:*

$$\begin{aligned} \mathbf{v}(q) &\geq \min\{\mathbf{v}(\pi_j) \mid j \geq i\} \\ \mathbf{v}(q) &\leq \max\{\mathbf{v}(\pi_j) \mid j \leq i\} \end{aligned}$$

Proof. Proposition 2.24 follows from a technique of "strategy translation" similar to the one used in the proof of (2.1): both Eve and Adam can play their π-strategy until the token reaches a random state, and then revert to an (ε-)optimal strategy. Propositions 2.22 and 2.23 yield then the desired inequations. □

2.2.2 Evaluating a Permutation

Our first step in order to evaluate a permutation π is to compute its values from the π-strategies σ_π and τ_π: $v_\pi = v_{\sigma_\pi,\tau_\pi}$. We denote by $v_\pi[i]$ the value of the i-th random state in π: $v_\pi[i] = v_\pi(\pi_i)$. It follows immediately from Propositions 2.22 and 2.23 that all the states in the same region share the same π-value: $\pi(q) = i \Rightarrow v_\pi(q) = v_\pi[i]$. We can also interpret these values using a "compacted" $\frac{1}{2}$-player reachability game $\mathfrak{G}^\pi$ with $k+2$ states:

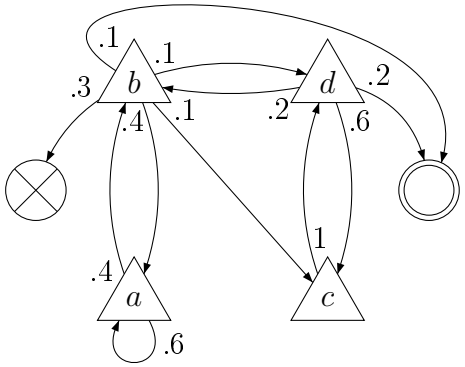
- $\mathfrak{Q} = \{0, \ldots, k+1\}$
- $\mathfrak{d}(i)(\otimes) = \delta(\pi_i)(\otimes)$
- $\mathfrak{d}(i)(\circledcirc) = \delta(\pi_i)(\circledcirc)$
- $\mathfrak{d}(i)(j) = \delta(\pi_i)(W_\pi[j])$

This amounts to merging each region $W_\pi[i]$ into a single state i. Figure 2.6 shows the game resulting from our running example, as a graph (2.6(a)) and as a matrix (2.6(b)).

This interpretation is used in Procedure `Values(`$\mathcal{G}, \pi, W$`)` to compute the π-values, using a primitive `MarkovChainSolver`.

In the game of Figure 2.6 with the permutation $\pi = abcd$, we get $v_\pi(a) = v_\pi(b) = .4$ and $v_\pi(c) = v_\pi(d) = .7$. As we will see, the permutation π is *self-consistent* (Definition 2.26) and *live* (Definition 2.27), thus for each i, $v_\pi[i]$ is the value of all the states in W_π in the original game (Lemma 2.33).

The notion of *self-consistency* is our equivalent to the notion of stability: in strategy-based algorithms, a "good" strategy for Eve sends the token to



(a) Graph Representation

	⊗	a	b	c	d	◎
⊗	1	0	0	0	0	0
a	0	.6	.4	0	0	0
b	.3	.4	0	.1	.1	.1
c	0	0	0	0	1	0
d	0	0	.2	.6	.2	0
◎	0	0	0	0	0	1

(b) Matrix Representation

Figure 2.6: The "compacted" game $\mathfrak{G}^\pi$

Input: A reachability game $\mathcal{G}$, a permutation π, and a partition W
Output: The π-values
1 **for** $(i = 1, i \leq k, i++)$ **do**
2 **for** $(j = 0, j \leq k+1, j++)$ **do**
3 $\texttt{mc}[i][j] \leftarrow \delta(\pi_i)(W[j])$
4 **end**
5 **end**
6 $v \leftarrow$`MarkovChainSolver(mc)`
7 **return** v

Procedure `Values`$(\mathcal{G}, \pi, W)$

the successor with the highest value *computed from the candidate strategy*; in permutation-based algorithms, a "good" permutation is consistent with the preorder of the values *computed from the candidate permutation*. We first define consistency in the general case of independent permutation and values for the random states.

Definition 2.25 *A permutation π is consistent with a set of values v if and only if for any two states π_i and π_j in $\mathcal{Q}_R$, $i < j \Rightarrow v(\pi_i) \leq v(\pi_j)$.*

Definition 2.26 *A permutation π is self-consistent if and only if it is consistent with v_π: for any two states π_i and π_j in $\mathcal{Q}_R$, $i < j \Rightarrow v_\pi[i] \leq v_\pi[j]$.*

Input: A permutation π and a vector of values v
Output: The consistency of π and v
1 `consistent` $\leftarrow$ `true`
2 **for** $(i = 1, i \leq k, i++)$ **do**
3 | `consistent` $\leftarrow$ `consistent` $\wedge (v[i] \leq v[i+1])$
4 **end**
5 **return** `consistent`

Procedure `Consistent`(π, v)

It can be shown that the values of a self-consistent permutation are solution to (2.1). This would be enough to get an algorithm for stopping games, as we will show that there is always a self-consistent permutation. However, the stopping reduction comes with a price, and we can avoid it with a cheaper logical condition: the notion of *liveness* captures the intuitive fact that a random state π_i with a positive value always has a positive probability to immediately go to a better region (from Eve's point of view).

Definition 2.27 *A permutation π over the set $\mathcal{Q}_R$ is live if and only if for any state $\pi_i \in \mathcal{Q}_R, \delta(\pi_i)(\cup_{j>i}^{k+1} W_\pi[j]) > 0$.*

Input: A reachability game $\mathcal{G}$, a permutation π, and a partition W
Output: The liveness of π in $\mathcal{G}$
1 `live` $\leftarrow$ `true`
2 **for** $(i = 1, i \leq k, i++)$ **do**
3 | `live` $\leftarrow$ `live` $\wedge (\delta(\pi_i)(\cup_{j>i} W[j]) > 0)$
4 **end**
5 **return** `live`

Procedure `Live`$(\mathcal{G}, \pi, W)$

One could think that this notion is already captured by self-consistency, as it is a "bad idea" for Eve to send the token to a random state that does not verify the internal property. However, the choice of the permutation also effects Adam's strategy: if he wrongly chooses to avoid a state, all the values may grow, with the possible side-effect to hide the initial mistake. We give an example of this process in Figure 2.7, which zooms on a detail of Figure 2.1.

Eve's strategy in $\bigcirc$ should be to send the token to b, as Adam could otherwise trap the play in $\{a, \bigcirc, \square\}$. However, let us consider the unlive

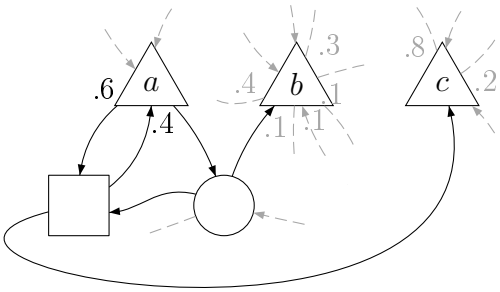


Figure 2.7: Liveness does not follow from self-consistency

permutation $\mu = bcad$: Adam sends the token from $\Box$ to c to avoid a; Eve sends the token from $\bigcirc$ to $\Box$ to reach either a or c. We have thus $v_\mu(a) = v_\mu(c)$. Actually, $v_\mu(b) \leq v_\mu(a) = v_\mu(c) \leq v_\mu(d)$, so μ is self-consistent, but the μ-values are not the correct ones. Formally, the point of liveness is expressed by Proposition 2.28.

Proposition 2.28 *Let $\mathcal{G} = (\mathcal{A}, \mathrm{Reach}(\circledcirc))$ be a $2\frac{1}{2}$-player reachability game and π be a live permutation. Then, for any strategy τ for Adam,*

$$\mathbb{P}_q^{\sigma_\pi,\tau}(\mathrm{Reach}(\circledcirc) \vee \mathrm{Reach}(\otimes)) = 1 \ .$$

Proof. By Lemma 1.3, the limit of a play ρ is an end-component with probability one. Let X be an end-component of $\mathcal{G}^{\sigma_\pi,\tau}$. We denote the integer $i = \max\{j | X \cap W_\pi[j] \neq \emptyset\}$ by i. There are three cases:

$i = 0$: As $\mathcal{G}$ is normalised, $X = \{\otimes\}$.

$1 \leq i \leq k$: By Proposition 2.22 and by definition of i, π_i belongs to X. By liveness of π, $\delta(\pi_i)(\cup_{j=i+1}^{k+1} W_\pi[j]) > 0$. As X is an end-component, there is a $j > i$ such that $W_\pi[j] \cap X \neq \emptyset$, in contradiction with the definition of i.

$i = k + 1$: As X is strongly connected, $X = \{\circledcirc\}$.

Proposition 2.28 follows. □

In a sense, liveness is a counterpart for the stopping property, with Proposition 2.28 used in the proofs in lieu of the characteristics of stopping games. Notice that liveness is not a "weaker" property: there are stopping games with unlive permutations (see for example Figure 2.8 on page 42).

2.2.3 Algorithm and Correctness

Our algorithm, described as Algorithm 2.9, consists then in an exhaustive search for a live and self-consistent permutation.

Input: A reachability game G
Output: A partition of $\mathcal{Q}$ and the corresponding values
forall $\pi \in \mathcal{S}_k$ **do**
  $W \leftarrow \texttt{Regions}(\mathcal{G}, \pi)$
  $v \leftarrow \texttt{Values}(\mathcal{G}, \pi, W)$
  $\texttt{self} \leftarrow \texttt{Consistent}(\pi, v)$
  $\texttt{live} \leftarrow \texttt{Live}(\mathcal{G}, \pi, W)$
  if $(\texttt{live} \wedge \texttt{self})$ **then**
    return (W, v)
  end
end

Algorithm 2.9: Permutation algorithm for reachability games

The remainder of this section is dedicated to the proof of its correctness:

Theorem 2.29 *Let $\mathcal{G} = (\mathcal{A}, \mathrm{Reach}(\circledcirc))$ be a $2\frac{1}{2}$-player reachability game. A run of Algorithm 2.9 on $\mathcal{G}$ terminates and returns the values of the states.*

Proof. The proof of Theorem 2.29 consists of two parts, which are proven separately:

- There is a live and self-consistent permutation (Lemma 2.30).
- If a permutation π is live and self-consistent, then v_π are the optimal values for the regions W_π (Lemma 2.33).

□

Lemma 2.30 *Let $\mathcal{G} = (\mathcal{A}, \mathrm{Reach}(\circledcirc))$ be a $2\frac{1}{2}$-player reachability game. At least one permutation is live and self-consistent in $\mathcal{G}$.*

Proof. The proof of Lemma 2.30 is itself in two parts: first, we show that there is a live permutation consistent with the values of the game (Proposition 2.31); then we show that such a permutation is self-consistent (Proposition 2.32). □

Proposition 2.31 *There is a live permutation consistent with the values of* $\mathcal{G}$.

Proof. The permutation is chosen starting from π_k, and going down to π_1. At each step, the state π_i is chosen so that:

- $\mathbf{v}(\pi_i) = \max\{\mathbf{v}(q) \mid q \in \mathcal{Q} \setminus \cup_{j>i} W_\pi[i]\}$
- $\delta(\pi_i)(\cup_{j>i} W_\pi[i]) > 0$

The existence of a suitable random state is proved by contradiction: as $\mathcal{Q} \setminus \cup_{j>i} W_\pi[i]$ is a deterministic trap for Eve, a trapping strategy for Adam ensures that the token can enter $\cup_{j>i} W_\pi[i]$ only through a random transition. Thus, a (non-positional) strategy for Adam which consists in playing the trapping strategy until the token enters $\cup_{j>i} W_\pi[i]$ and then switch to an optimal strategy bounds the probability of reaching $\circledcirc$ to $\max\{\mathbf{v}(r) \mid r \in \mathcal{Q}_R \setminus \{\pi_{i+1}, \ldots, \pi_k\} \wedge \delta(r)(\cup_{j>i} W_\pi[i]) > 0\}\}$ □

Proposition 2.32 *Let* $\mathcal{G} = (\mathcal{A}, \mathrm{Reach}(\circledcirc))$ *be a* $2\frac{1}{2}$*-player reachability game, and* π *be a live permutation consistent with the optimal values* $\mathbf{v}$ *of* $\mathcal{G}$*. Then* v_π *is self-consistent.*

Proof. Quite naturally, we prove that π is self-consistent by showing that the π-values are the optimal values. The key arguments is that the expected (optimal) value after n moves is constant, when the initial state is fixed and the players play the π-strategies. We fix an initial state $q \in \mathcal{Q}$, and we define $e : \mathbb{N} \to [0, 1]$ by $e(n) = \sum_{s \in \mathcal{Q}} \mathbf{v}(s) \cdot \mathbb{P}_q^{\sigma_\pi, \tau_\pi}(\rho_n = s)$. We have:

- $\forall s \in \mathcal{Q}_E, \mathbf{v}(s) = \mathbf{v}(\sigma_\pi(s))$ —by Proposition 2.24
- $\forall s \in \mathcal{Q}_A, \mathbf{v}(s) = \mathbf{v}(\tau_\pi(s))$ —by Proposition 2.24
- $\forall r \in \mathcal{Q}_R, \mathbf{v}(r) = \sum_{s \in \mathcal{Q}} \delta(r)(s) \cdot \mathbf{v}(t)$ —by (2.1)

Thus, for all $n \in \mathbb{N}, e(n) = e(n+1)$, and so $\mathbf{v}(q) = e(0) = e(n) = \sum_{s \in \mathcal{Q}} \mathbf{v}(s) \cdot \mathbb{P}_q^{\sigma_\pi, \tau_\pi}(\rho_n = s)$. As π is live, Proposition 2.28 yields $\mathbb{P}_q^{\sigma_\pi, \tau_\pi}(\mathrm{Reach}(\circledcirc) \vee \mathrm{Reach}(\otimes)) = 1$. It follows that $\mathbf{v}(q) = \mathbb{P}_q^{\sigma_\pi, \tau_\pi}(\mathrm{Reach}(\circledcirc)) = v_\pi(q)$. By hypothesis, π is consistent with $\mathbf{v}$. It follows that π is self-consistent, which completes the proof of Proposition 2.32. □

Lemma 2.33 *Let $\mathcal{G} = (\mathcal{A}, \text{Reach}(\circledcirc))$ be a $2\frac{1}{2}$-player reachability game, and π be a live and self-consistent permutation. Then, the π-strategies are optimal.*

Proof. The proof is close to the one of Proposition 2.32. We fix an initial state q and two positional strategies σ and τ for Eve and Adam, and we define the functions f and g by: $f(n) = \sum_{s \in \mathcal{Q}} v_\pi(s) \cdot \mathbb{P}_q^{\sigma_\pi, \tau}(\rho_n = s)$ and $g(n) = \sum_{s \in \mathcal{Q}} v_\pi(s) \cdot \mathbb{P}_q^{\sigma, \tau_\pi}(\rho_n = s)$. We have:

- $\forall s \in \mathcal{Q}_E, v_\pi(s) = v_\pi(\sigma_\pi(s))$ —by (2.2)
- $\forall s \in \mathcal{Q}_A, v_\pi(s) = v_\pi(\tau_\pi(s))$ —by (2.2)
- $\forall s \in \mathcal{Q}_E, v_\pi(s) \geq v_\pi(\sigma(s))$ —by self-consistency of π.
- $\forall s \in \mathcal{Q}_A, v_\pi(s) \leq v_\pi(\tau(s))$ —by self-consistency of π.
- $\forall r \in \mathcal{Q}_R, v_\pi(r) = \sum_{s \in \mathcal{Q}} \delta(r)(s) \cdot v_\pi(t)$ —by (2.2)

It follows that $\forall n \in \mathbb{N}, f(n) \leq f(n+1)$ and $g(n) \geq g(n+1)$. We get immediately $v_\pi(q) = g(0) \geq \lim_{n \to \infty} g(n) \geq \mathbb{P}_q^{\sigma, \tau_\pi}(\text{Reach}(\circledcirc))$. As π is live, Proposition 2.28 yields $\mathbb{P}_q^{\sigma_\pi, \tau}(\text{Reach}(\circledcirc) \vee \text{Reach}(\otimes)) = 1$, and thus $v_\pi(q) = f(0) \leq \lim_{n \to \infty} f(n) = \mathbb{P}_q^{\sigma_\pi, \tau}(\text{Reach}(\circledcirc))$.

Thus, σ_π and τ_π are co-optimal, and Lemma 2.33 follows. □

2.2.4 Complexity analysis

Theorem 2.34 *The values and optimal strategies of a normalised reachability game $\mathcal{G} = (\mathcal{A}, \text{Reach}(\circledcirc))$ are computable in time $O(|\mathcal{Q}_R|! \cdot (|\mathcal{T}| \cdot \log |\mathcal{Q}| + |\delta|))$, where $|\delta|$ is the maximal bit-length of a transition probability in δ.*

Proof. In the worst case, Algorithm 2.9 enumerates all the $|\mathcal{Q}_R|!$ permutations of $\mathcal{Q}_R$. For each permutation π, the algorithm computes the π-regions and π-strategies (in time $O(|\mathcal{T}| \cdot \log |\mathcal{Q}|)$, see [Cha06]). It computes then the values of the resulting $\frac{1}{2}$-player reachability game (in time $O(|\mathcal{Q}_R^3| \cdot |\delta|)$, see [Dix82]). The tests for liveness and self-consistency can then be performed in time $O(|\mathcal{Q}_R|)$. Theorem 2.34 follows. □

The number of iterations is similar to what we get with strategy-based algorithms, but it depends on different figures ($\mathcal{Q}_R$ in our algorithm, $\mathcal{Q}_E$

and the outdegree of Eve's states in the strategy improvement). This difference is interesting when dealing with unbalanced arenas. For example, Corollary 2.35 presents an extreme case where our algorithm is polynomial:

Corollary 2.35 *For each k, the values and optimal strategies of a normalized reachability game $\mathcal{G} = (\mathcal{A}, \mathrm{Reach}(\circledcirc))$ such that $|\mathcal{Q}_R| \leq k$ are computable in time $O(|\mathcal{Q}| \cdot |\mathcal{T}| + |\delta|)$, where $|\delta|$ is the maximal bit-length of a transition probability in δ.*

The advantage of our algorithm is the simplicity of the internal loop: in complexity terms, it is much simpler to solve reachability games on $\frac{1}{2}$-player arenas than on $1\frac{1}{2}$-player ones; we will see in Chapter 3 that this simplicity also allows us to adapt our algorithm to a very general class of winning conditions.

2.3 Heuristics for permutation algorithms

The theoretical bounds on the number of loops in the permutation algorithm and the strategy improvement algorithm are different, yet similar. However, an important asset of the strategy improvement is its efficiency in practice. Although there is no proof for Conjecture 2.20, the study of practical cases suggests that the number of iterations is *linear* in the number of states.

The aim of this section is to consider similar heuristics in the update of permutations. We first describe a very natural heuristic (2.3.1), which works only for $1\frac{1}{2}$-player games (2.3.2). We present then a "mixed" heuristic, using both our permutation techniques and the improvement step of Algorithm 2.3 (2.3.3). The resulting algorithm is correct for all $2\frac{1}{2}$-player reachability games.

2.3.1 Value-based improvement

We first consider a very simple heuristic: in each iteration, the new permutation is consistent with the values of the former one. The resulting algorithm is described as Algorithm 2.10.

Notice that at line 2, we require that the chosen permutation is live, as well as consistent with the former values. This avoids getting stuck in a self-consistent unlive permutation, like the one presented in Figure 2.7. The **until** condition of line 4 can thus only be met by a live and self-consistent

Input: a game $\mathcal{G} = (\mathcal{A}, \mathrm{Reach}(\circledcirc))$
Output: the values of $\mathcal{G}$
repeat
  choose π live and consistent with v
  $v \leftarrow v_\pi$
until π is consistent with v
return v

Algorithm 2.10: Value-based permutation improvement

permutation, so Algorithm 2.10 returns only correct results. In the case of $1\frac{1}{2}$-player games, such a choice is always possible:

Lemma 2.36 *Let $\mathcal{G} = (\mathcal{A}, \mathrm{Reach}(\circledcirc))$ be a $1\frac{1}{2}$-player reachability game and π be a live permutation. There is a live permutation μ consistent with v_π.*

Proof. By application of Lemma 2.30 to the $\frac{1}{2}$-player game $\mathbb{G} = \mathcal{G}^{\sigma_\pi}$, we can define a live and self-consistent permutation μ in $\mathbb{G}$. As $\mathbb{G}$ is a $\frac{1}{2}$-player game, its values do not depend on any strategies, so "self-consistency in $\mathbb{G}$" translates directly as "consistency with v_π". The interpretation of the liveness property is a little more involved. It guarantees that:

$$\forall i \in 1 \dots k, \delta(\mu_i) \left(\bigcup_{j>i} \mathbb{W}_\mu[j] \right) > 0 \ .$$

However, in general, $\mathbb{W}_\mu[j] \neq W_\mu[j]$. Rather, we have $\mathbb{W}_\mu[j] = W_\pi[\pi(\mu_j)]$. And, as $\mathcal{G}$ is an $1\frac{1}{2}$-player game, we get $W_\pi[\pi(\mu_j)] \subseteq \mathrm{Det}_E(\mu_j, \mathcal{A})$. So:

$$\bigcup_{j>i} \mathbb{W}_\mu[j] = \bigcup_{j>i} W_\pi[\pi(\mu_j)] \subseteq \bigcup_{j>i} \mathrm{Det}_E(\mu_j, \mathcal{A}) = \bigcup_{j>i} W_\mu[j]$$

Thus μ is live in the game $\mathcal{G}$, and Lemma 2.36 follows. □

We need then to show that Algorithm 2.10 cannot have an endless run. Again in the case of $1\frac{1}{2}$-player games, Lemma 2.38 shows that the values computed through a run are growing, ensuring that each permutation is considered at most once. We first need to establish Proposition 2.37:

Proposition 2.37 *Let $\mathcal{G} = (\mathcal{A}, \mathrm{Reach}(\circledcirc))$ be a $1\frac{1}{2}$-player reachability game, π be a live permutation and μ be a live permutation consistent with v_π. Then, for any state $q \in \mathcal{Q}$, $v_\pi(q) \leq v_\pi(\mu_{\mu(q)})$.*

Proof. As $\mathcal{G}$ is a $1\frac{1}{2}$-player game, $\mu(q)$ is equal to $\max\{i \mid q \in \mathrm{Det}_E(\mu_i, \mathcal{A})\}$. Since $q \in \mathrm{Det}_E(\pi_{\pi(q)}, \mathcal{A})$, it follows that $\mu(q) \geq \mu(\pi_{\pi(q)})$. By consistency of μ and v_π, we get $v_\pi(\mu_{\mu(q)}) \geq v_\pi(\mu_{\mu(\pi_{\pi(q)})}) = v_\pi(q)$. Proposition 2.37 follows. □

Lemma 2.38 *Let $\mathcal{G} = (\mathcal{A}, \mathrm{Reach}(\circledcirc))$ be a $1\frac{1}{2}$-player reachability game, π be a live permutation and μ be a live permutation consistent with v_π. Then $v_\pi \leq v_\mu$*

Proof. For a given initial state q, we define the function f by:

$$f(n) = \sum_{s \in \mathcal{Q}} v_\pi(\mu_{\mu(s)}) \cdot \mathbb{P}_q^{\sigma_\mu}(\rho_n = s) \ .$$

If s is a state of Eve, the definition of σ_μ yields $v_\pi(\mu_{\mu(s)}) = v_\pi(\mu_{\mu(\sigma_\mu(s))})$. If r is a random state, the situation is more complex:

$$\begin{array}{rcll} v_\pi(\mu_{\mu(r)}) & = & v_\pi(r) & \text{—as } r \text{ is a random state} \\ & = & \sum_{s \in \mathcal{Q}} \delta(r)(s) v_\pi(s) & \text{—by (2.2)} \\ & \leq & \sum_{s \in \mathcal{Q}} \delta(r)(s) v_\pi[\mu_{\mu(s)}] & \text{—by Proposition 2.37} \end{array}$$

We get:

$$v_\pi(q) \leq v_\pi(\mu_{\mu(q)}) = f(0) \leq \lim_{n \to \infty} f(n) \leq \mathbb{P}_q^{\sigma_\mu}(\mathrm{Reach}(\circledcirc)) = v_\mu(q)$$

Lemma 2.38 follows. □

Lemmas 2.36 and 2.38 yield Theorem 2.39:

Theorem 2.39 *Let $\mathcal{G} = (\mathcal{A}, \mathrm{Reach}(\circledcirc))$ be a $1\frac{1}{2}$-player reachability game. Algorithm 2.10 terminates, and returns correct values and regions.*

Proof. Lemma 2.36 guarantees that the update process is sound. Lemma 2.33 ensures that Algorithm 2.10 returns only correct values. Lemma 2.38 shows that the values are growing. Notice that the inequality is not strict: the values of two successive permutations can be equal. In this case, though, the later is self-consistent, so Algorithm 2.9 terminates. Theorem 2.39 follows. □

We have no proof that the worst case complexity of this algorithm is actually better than the complexity of Algorithm 2.2. However, we conjecture that it is actually polynomial:

Conjecture 2.40 *There is a polynomial P such that a run of Algorithm 2.10 on a $1\frac{1}{2}$-player game with n states executes at most $P(n)$ loops.*

This conjecture is actually equivalent to the classical conjecture for the strategy improvement algorithm:

Proposition 2.41 *Conjectures 2.19 and 2.40 are equivalent.*

Proof. We prove this equivalence by showing that if either conjecture does not hold, then the other does not hold. In both cases, the proof relies on a transformation of a counter-witness game $\mathcal{G} = (\mathcal{A}, \mathrm{Reach}(\circledcirc))$ into another game $\mathbb{G}$:

- $\neg 2.19 \Rightarrow \neg 2.40$: $\mathbb{G}$ is a copy of $\mathcal{G}$, except that each transition $q \to s$ is replaced by $\mathfrak{q} \leftrightarrow \mathfrak{r} \to \mathfrak{s}$, where $\mathfrak{r}$ is a new random state with equal chances to send the token to $\mathfrak{q}$ and $\mathfrak{s}$. The strategies (and their values) are the same in both games, but in $\mathbb{G}$ all of them are permutation strategies. Likewise, the strategy improvement of Algorithm 2.2 corresponds to the permutation improvement of Algorithm 2.10. Thus, any run of Algorithm 2.2 on the game $\mathcal{G}$ is matched step-by-step with a run of Algorithm 2.10 on the game $\mathbb{G}$.

- $\neg 2.40 \Rightarrow \neg 2.19$: $\mathbb{G}$ is a copy of $\mathcal{G}$ with "shortcut" transitions: whenever the player can make two successive moves in $\mathcal{G}$, *e.g.* $q \to r \to s$ with $q, r \in \mathcal{Q}_A$, there is a direct transition $\mathfrak{q} \to \mathfrak{s}$ in $\mathbb{G}$. The values of the permutations are the same in both games. Furthermore, if π can be transformed in μ in a run of Algorithm 2.10 on $\mathcal{G}$, then any π-strategy can be transformed into a μ-strategy in a run of Algorithm 2.2 on $\mathbb{G}$. Thus, any run of Algorithm 2.10 on the game $\mathcal{G}$ is matched step-by-step with a run of Algorithm 2.2 on the game $\mathbb{G}$.

Proposition 2.41 follows. □

We have shown the correctness of Algorithm 2.10 for $1\frac{1}{2}$-player games with states of Eve. The straightforward adaptation for $1\frac{1}{2}$-player games with states of Adam ($1\frac{1}{2}$-player *safety* games) works just as well, and the proofs require only minor modifications.

2.3.2 Value-based improvement and $2\frac{1}{2}$-player games

The simple heuristic of Algorithm 2.10 does not work in the case of $2\frac{1}{2}$-player games. The first problem is that Lemma 2.36 does not hold anymore, as witnessed by Figure 2.8.

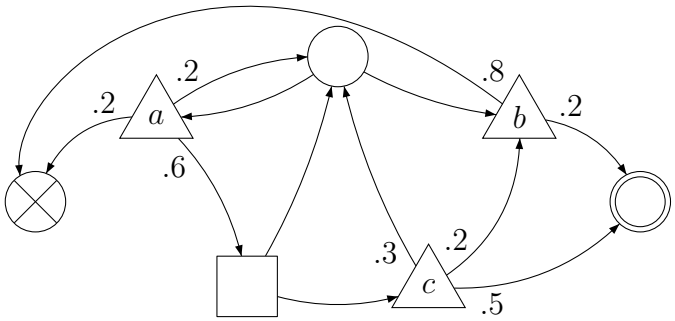


Figure 2.8: Unlive values

In this game, a permutation is live if and only it ranks a lower than b. But, if we start from the live permutation $\pi = cab$, a problem arises: Eve sends the token from $\bigcirc$ to b, and Adam sends it from $\square$ to c. The resulting values are $v_\pi(a) = .4$, $v_\pi(b) = .2$, and $v_\pi(c) = .6$. These values are totally ordered, and the only consistent permutation is $\mu = bac$, which is not live.

This problem could be circumvented by backtracking to the case of stopping games, as self-consistency guarantees optimality in this case. This allow us to lift the liveness restriction in line 2, while guaranteeing the correctness of the result. This works correctly in the game of Figure 2.8 —which is stopping: as μ is unlive, it is not self-consistent. Indeed, the corresponding values are $v_\mu(a) = 0$, $v_\mu(b) = .2$, and $v_\mu(c) = .54$. The only permutation consistent with v_μ is $\kappa = abc$, which is self-consistent: $v_\kappa(a) = .16$, $v_\kappa(b) = .2$, and $v_\kappa(c) = .6$.

However, our proof of Lemma 2.38 cannot be adapted, as it relies on a notion of "progress" which does not make sense in $2\frac{1}{2}$-player games. Other invariants could (and have) been considered, but to no avail: once again, we found a counter-example, presented in Figure 2.9, where Algorithm 2.10 gets stuck in an infinite cycle.

The game of Figure 2.9 is stopping: as long as the token has not reached one of the final states, it is bound to visit again one of the random states;

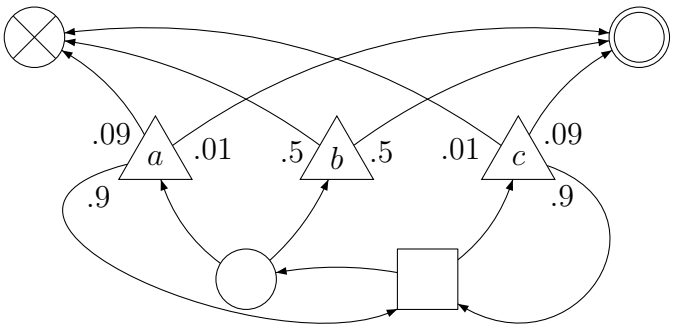


Figure 2.9: Infinite run

and each of these states has a positive probability to send the token to the final states. Thus, the token reaches a final state with probability one.

In this game, the only self-consistent permutation is $\pi = abc$. The corresponding strategies are $\bigcirc \to b$ and $\square \to \bigcirc$, and the values are $v_\pi(a) = .46$, $v_\pi(\square) = v_\pi(\bigcirc) = v_\pi(b) = .5$, and $v_\pi(c) = .54$

But, if we consider a run where the permutation $\mu = acb$ is chosen at the first visit to line 2, we get stuck in an infinite run:

- The μ-strategies send the token from $\bigcirc$ to b and from $\square$ to c. The resulting values are $v_\mu(a) = .82$, $v_\mu(\bigcirc) = v_\mu(b) = .5$, and $v_\mu(\square) = v_\mu(c) = .9$. When the **repeat** loop ends, and the modified Algorithm 2.10 goes back to line 2, its only choice is $\kappa = bac$.

- The κ-strategies send the token from $\bigcirc$ to a and from $\square$ to $\bigcirc$. The resulting values are $v_\kappa(\square) = v_\kappa(\bigcirc) = v_\kappa(a) = .1$, $v_\kappa(b) = .5$, and $v_\kappa(c) = .18$. When the **repeat** loop ends, and the modified Algorithm 2.10 goes back to line 2, its only choice is $\mu = acb$.

The algorithm oscillates endlessly between μ and κ, leading to an infinite run. This prohibits any straightforward adaptation of Algorithm 2.10 to $2\frac{1}{2}$-player games.

2.3.3 Mixed improvement

In order to get a working permutation-improvement algorithm for the general case of $2\frac{1}{2}$-player games, we need to consider an *asymmetric* improvement

step, alike to the one used in the strategy improvement algorithm. The idea is that only Eve uses her π-strategy from (2.2.1), whereas Adam plays an optimal counter-strategy to σ_π: instead of using the π-values $v_\pi = v_{\sigma_\pi,\tau_\pi}$, we use $\mathbb{v}_\pi = v_{\sigma_\pi}$. Apart from this, Algorithm 2.11 works exactly as Algorithm 2.10

Input: a game $\mathcal{G} = (\mathcal{A}, \mathrm{Reach}(\circledcirc))$
Output: the values of $\mathcal{G}$
1 **repeat**
2 | choose π live and consistent with v
3 | v $\leftarrow v_{\sigma_\pi}$
4 **until** π is consistent with v
5 **return** v

Algorithm 2.11: Mixed permuation improvement

Notice that at the end of the computation, π is consistent with $\mathbb{v}_\pi$, which is not self-consistency in the sense of Definition 2.26. We need thus to prove anew that the values returned by Algorithm 2.11 are correct, although the proof is almost identical to the proof of Lemma 2.33.

Lemma 2.42 *Let $\mathcal{G} = (\mathcal{A}, \mathrm{Reach}(\circledcirc))$ be a $2\frac{1}{2}$-player reachability game and π be a live permutation such that π is consistent with $\mathbb{v}_\pi$. Then $\mathbb{v}_\pi$ are the values of $\mathcal{G}$.*

Proof. As Eve can ensure $\mathbb{v}_\pi$ by playing σ_π, we just need to show that Adam can confine the probability of $\mathrm{Reach}(\circledcirc)$ to $\mathbb{v}_\pi$. In general an optimal counter-strategy to σ_π is not satisfying in that respect. However, we can use the π-strategy τ_π of Adam, just as in the proof of Lemma 2.33. We fix an initial state q and a positional strategy σ for Eve, and we define the function f by: $f(n) = \sum_{s\in\mathcal{Q}} \mathbb{v}_\pi(s) \cdot \mathbb{P}_q^{\sigma,\tau_\pi}(\rho_n = s)$. We have:

- $\forall s \in \mathcal{Q}_A, \mathbb{v}_\pi(s) = \mathbb{v}_\pi(\tau_\pi(s))$ —by consistency of π and $\mathbb{v}_\pi$
- $\forall s \in \mathcal{Q}_E, \mathbb{v}_\pi(s) \geq \mathbb{v}_\pi(\sigma(s))$ —by consistency of π and $\mathbb{v}_\pi$
- $\forall r \in \mathcal{Q}_R, \mathbb{v}_\pi(r) = \sum_{s\in\mathcal{Q}} \delta(r)(s) \cdot \mathbb{v}_\pi(t)$

It follows that f is increasing, so:

$$\mathbb{v}_\pi(q) = f(0) \geq \lim_{n\to\infty} f(n) \geq \mathbb{P}_q^{\sigma,\tau_\pi}(\mathrm{Reach}(\circledcirc)) \ .$$

Lemma 2.42 follows. □

Another consequence of using this notion of π-values is that the loop's inner complexity is much higher: we need to compute the values of a $1\frac{1}{2}$-player reachability game, instead of a $\frac{1}{2}$-player reachability game. This can be done by any $1\frac{1}{2}$-player game algorithm: strategy improvement, linear programming, or value-based permutation improvement (Algorithms 2.2, 2.4, and 2.10).

The remainder of the proof of correctness for Algorithm 2.11 is very close to the proof of Algorithm 2.10, with some extra complexity to account for the presence of Adam's states. For starters, the soundness of line 2, is resolved by Lemma 2.43:

Lemma 2.43 *Let $\mathcal{G} = (\mathcal{A}, \mathrm{Reach}(\circledcirc))$ be a $2\frac{1}{2}$-player reachability game and π be a live permutation. There is a live permutation μ consistent with v_π.*

Proof. By application of Lemma 2.30 to the $1\frac{1}{2}$-player game $\mathbb{G} = \mathcal{G}^{\sigma_\pi}$, we can define a live and self-consistent permutation μ in $\mathbb{G}$. By Lemma 2.33, μ is consistent with $\mathbb{v}_\pi$. The liveness of μ in $\mathbb{G}$ guarantees that:

$$\forall i \in 1 \dots k, \delta(\mu_i) \left(\bigcup_{j>i} \mathbb{W}_\mu[j] \right) > 0 \ .$$

By definition of the μ-regions in $\mathbb{G}$, $\cup_{j>i}\mathbb{W}_\mu[j]$ is equal to $\mathrm{Det}_E(\cup_{j>i}\{\mu_j\}, \mathcal{A}^{\sigma_\pi})$, which is a subset of $\mathrm{Det}_E(\cup_{j>i}\{\mu_j\}, \mathcal{A}) = \cup_{j>i} W_\mu[j]$. Thus μ is live in $\mathcal{G}$, and Lemma 2.43 follows. □

The absence of cycles is proved through a notion of progress:

We need then to show that Algorithm 2.10 cannot have an endless run. Again in the case of $1\frac{1}{2}$-player games, Lemma 2.38 shows that the values computed through a run are growing, ensuring that each permutation is considered at most once. We first need to establish Proposition 2.44:

Proposition 2.44 *Let $\mathcal{G} = (\mathcal{A}, \mathrm{Reach}(\circledcirc))$ be a reachability game, π be a live permutation and μ be a live permutation consistent with $\mathbb{v}_\pi$. Then, for any state $q \in \mathcal{Q}$, $\mathbb{v}_\pi(q) \leq \mathbb{v}_\pi(\mu_{\mu(q)})$.*

Proof. By definition, we have $\mathbb{\mu}(q) = \max\{i \mid q \in \mathrm{Det}_E(\cup_{j\geq i}\{\mu_j\}, \mathcal{A}^{\sigma_\pi})\}$, while $\mu(q) = \max\{i \mid q \in \mathrm{Det}_E(\cup_{j\geq i}\{\mu_j\}, \mathcal{A})\}$. Thus, $\mathbb{\mu}(q) \leq \mu(q)$. As μ is consistent with v_{σ_π}, we get:

$$v_{\sigma_\pi}(\mu_{\mu(q)}) \geq v_{\sigma_\pi}(\mu_{\mathbb{\mu}(q)}) = v_{\sigma_\pi, \mathbb{v}_\mu}(q) \geq v_{\sigma_\pi}(q) \ .$$

Proposition 2.44 follows. □

Lemma 2.45 *Let $\mathcal{G} = (\mathcal{A}, \mathrm{Reach}(\circledcirc))$ be a reachability game, π be a live permutation and μ be a live permutation consistent with v_{σ_π}. Then $v_{\sigma_\pi} \leq v_{\sigma_\mu}$*

Proof. We fix an initial state q and a strategy τ for Adam, and we define the function f by $f(n) = \sum_{s \in \mathcal{Q}} v_{\sigma_\pi}(\mu_{\mu(s)}) \cdot \mathbb{P}_q^{\sigma_\mu, \tau}(\rho_n = s)$. For a state s of Eve, we have by definition $\mu(s) = \mu(\sigma_\mu(s))$, so $v_{\sigma_\pi}(\mu_{\mu(s)}) = v_{\sigma_\pi}(\mu_{\mu(\sigma_\mu(s))})$. For a state of Adam, we have $\mu(s) \leq \mu(\tau(s))$, so the consistency of μ and v_{σ_π} yields $v_{\sigma_\pi}(\mu_{\mu(s)}) \leq v_{\sigma_\pi}(\mu_{\mu(\tau(s))})$. For a random random state r, the argument is:

$$\begin{array}{rcll} v_{\sigma_\pi}(\mu_{\mu(r)}) & = & v_{\sigma_\pi}(r) & \text{—as } r \text{ is a random state} \\ & = & \sum_{s \in \mathcal{Q}} \delta(r)(s) v_{\sigma_\pi}(s) & \text{—by (2.1)} \\ & \leq & \sum_{s \in \mathcal{Q}} \delta(r)(s) v_{\sigma_\pi}(\mu_{\mu(s)}) & \text{—by Proposition 2.44} \end{array}$$

We get:

$$v_{\sigma_\pi}(q) \leq v_{\sigma_\pi}(\mu_{\mu(q)}) = f(0) \leq \lim_{n \to \infty} f(n) \leq \mathbb{P}_q^{\sigma_\mu, \tau}(\mathrm{Reach}(\circledcirc)) = v_{\sigma_\mu, \tau}(q) \ .$$

As τ is an arbitrary strategy of Adam, we can conclude that $v_{\sigma_\pi}(q) \leq v_{\sigma_\mu}(q)$, and Lemma 2.45 follows. □

Lemmas 2.43 and 2.45 yield the correctness of Algorithm 2.11:

Theorem 2.46 *Algorithm 2.11 terminates and returns the values of its input.*

Proof. Lemma 2.43 guarantees that the update process is sound. Lemma 2.42 ensures that Algorithm 2.11 returns only correct values. Lemma 2.45 shows that the values are growing, so there are no infinite runs. Theorem 2.46 follows. □

We also conjecture that Algorithm 2.11 is polynomial:

Conjecture 2.47 *Algorithm 2.11 runs in polynomial time in the size of its input.*

However, we were only able to establish that this conjecture is stronger than its equivalent for strategy improvement:

Proposition 2.48 *If Conjecture 2.47 holds, then Conjecture 2.20 holds.*

Proof. This side of the proof of Proposition 2.41 works just as well in the case of $2\frac{1}{2}$-player games. □

2.4 Afterword

We proposed a new approach to the quantitative solution of $2\frac{1}{2}$-player reachability games. Our motivation in doing so is twofold.

- First, the complexity we get is orthogonal with the usual strategy-based approach: permutation algorithms are fixed-parameter tractable when the parameter is the number of random states in the game, whereas the complexity of strategy-improvement algorithms depends on the number of possible strategies for either player.
- Second, the removal of the stopping hypothesis makes our approach much more flexible, as we demonstrate in Chapter 3 by extending a permutation algorithm for all prefix-independent games.

An intriguing question, en route to the huge endeavour of finding a polynomial algorithm computing the values of $2\frac{1}{2}$-player reachability games, is whether our permutation-improvement algorithm is (strongly) polynomial on $1\frac{1}{2}$-player games.

Chapter 3

Prefix-independent conditions

"Those who do not remember the past are condemned to repeat it."

The Life of Reason
George Santayana

After our considerations on the most specific case of reachability games, we take the opposite direction in this chapter, and ponder the very general case of games with prefix-independent winning conditions. A condition is prefix-independent if adding a finite prefix to a play does not change the winner. In the even more general case of prefix-closed conditions, adding a finite prefix may change a play winning for Adam into one winning for Eve, but not the other way round.

One of the main motivation for studying prefix-independent conditions is that they subsume parity conditions. So, even though not all regular conditions are prefix-independent, our results have direct consequences for regular games. On a verification point of view, prefix-independence corresponds to cases where local glitches are tolerated in the beginning of a run, as long as the specification is met in the limit, in the spirit of self-stabilising protocols. Finally, one of the most popular payoff functions in economic games, the mean-payoff function, is also prefix-independent.

In Section 3.1, we study the relations between the different winning regions in prefix-independent games, while Section 3.2 uses them from an algorithmic point of view. Section 3.3 takes on the quantitative problems, and shows that many results of Chapter 2 carry over to prefix-independent games.

3.1 Winning regions

In this section, we contemplate the qualitative problems of prefix-independent games from an abstract point of view, and look for relations between the different qualitative regions. These relations can be sorted in three categories:

Loose inclusions: Our first question is whether the three "weak" regions (heroic, positive, and bounded) and the three "strong" regions (limit-one, almost-sure, and sure) really are different for prefix-independent games. In safety games, the "strong" regions are equal [dAHK98]; in regular games, the limit-sure and almost-sure regions are equal, but not the sure region [dAH00].

Existential and universal properties: A second type of properties relates the emptiness or completeness of two different regions for the same player. For example, the universal and existential bounded-limit properties of prefix-independent games [Cha07a][1] are:

$$\mathrm{Win}_E^{\mathcal{W},\gg 0}(\mathcal{A}) = \mathcal{Q} \Longrightarrow \mathrm{Win}_E^{\mathcal{W},\sim 1}(\mathcal{A}) = \mathcal{Q}$$
$$\mathrm{Win}_E^{\mathcal{W},\sim 1}(\mathcal{A}) = \emptyset \Longrightarrow \mathrm{Win}_E^{\mathcal{W},\gg 0}(\mathcal{A}) = \emptyset$$

Determinacy: Last but not least, determinacy properties state that from any state of the game, either Eve or Adam has a winning strategy — for dual notions of winning. In 2-player games, there is not much to do beyond the pure determinacy of Borel games by Martin [Mar75]. His quantitative determinacy of $2\frac{1}{2}$-player games[Mar98], however, is not wholly satisfying: the regions $\mathrm{Win}_E^{\mathcal{W},\geq .5}(\mathcal{A})$ and $\mathrm{Win}_A^{\mathcal{W},\geq .5}(\mathcal{A})$ cover the whole graph, but they are not disjoint.

We first discuss the evolution of values and σ-values in prefix-independent games (3.1.1). This prompts us to define *reset strategies*, a construction which builds conditional almost-sure strategies (3.1.2). In particular, we use them to prove: that positive and bounded regions, limit-one and almost-sure regions, are equal in prefix-independent games; universal and existential positive-almost properties for prefix-independent games; and the qualitative determinacy of prefix-independent games (3.1.3).

[1] It is called a positive-limit property in the paper, but depends on the existence of a state with positive value: this is a "bounded" state, according to [dAH00]'s taxonomy.

3.1.1 Values and σ-values

In prefix-independent games, as in reachability games, the value of a prefix is the value of its last state. We can thus use strategy translations to derive the value of a state from its owner and the value of its successors:

$$\begin{array}{rcl} \forall q \in \mathcal{Q}_E, \mathbf{v}(q) & = & \max\{\mathbf{v}(s) \mid (q,s) \in \mathcal{T}\} \\ \forall q \in \mathcal{Q}_A, \mathbf{v}(q) & = & \min\{\mathbf{v}(s) \mid (q,s) \in \mathcal{T}\} \\ \forall q \in \mathcal{Q}_R, \mathbf{v}(q) & = & \displaystyle\sum_{(q,s)\in\mathcal{T}} \delta(q)(s) \cdot \mathbf{v}(s) \end{array} \tag{3.1}$$

However, there is no "target state", whose value is fixed to one, nor a notion of "stopping games", with a unique solution to (3.1). This system is thus insufficient to the task of computing the values. Still, it yields Proposition 3.1:

Proposition 3.1 *Let $\mathcal{A}$ be an arena, and $\mathcal{W}$ a prefix-closed winning condition. The region $\mathrm{Win}_A^{\mathcal{W},\sim 1}(\mathcal{A})$ — the region with value 0 — is a trap for Eve, and the $\mathrm{Win}_E^{\mathcal{W},\sim 1}(\mathcal{A})$ — the region with value 1 — is a trap for Adam.*

If pure and positional strategies were sufficient for prefix-independent games, we could use similar equations for the values of the strategies. As this is not the case, we have to satisfy ourselves with infinite systems on the σ-values of the prefixes consistent with a pure[2] strategy σ:

Definition 3.2 *The σ-value of a finite play w consistent with a pure strategy σ for Eve is the infimum of the $\{\sigma,\tau\}$-values under the assumptions that w is a prefix of the ensuing play:*

$$v_\sigma(w) = \inf_\tau \mathbb{P}_{w_0}^{\sigma,\tau}(\mathcal{W} \mid \rho_0 = w_0, \rho_1 = w_1, \ldots) \ .$$

We can derive an infinite system of equations on the σ-values:

[2]Most of the results on σ-values and reset strategies could be adapted for semi-randomised strategies — with some extra caution. However, they are useless for strategies with random memory.

$$\begin{array}{rcl} \text{if } q \in \mathcal{Q}_E, v_\sigma(wq) & = & v_\sigma(wq \cdot \sigma(wq)) \\ \text{if } q \in \mathcal{Q}_A, v_\sigma(wq) & = & \min\{v_\sigma(wqs) \mid (q,s) \in \mathcal{T}\} \\ \text{if } q \in \mathcal{Q}_R, v_\sigma(wq) & = & \sum_{(q,s)\in\mathcal{T}} \delta(q)(s) \cdot v_\sigma(wqs) \end{array} \tag{3.2}$$

Using σ-values, we can give an educated opinion on the outcome on the play. In particular, for any positive real number $\eta < 1$, we define:

$$\mathcal{L}_\eta \;=\; \{\rho \in \Omega^\sigma \mid \exists i, v_\sigma(\rho_0 \dots \rho_i) \leq \eta\} \ .$$

Proposition 3.3 *Let q be a state of $\mathcal{Q}$, τ be a strategy for Adam, and $\eta < \nu \leq v_\sigma(q)$ be two positive real numbers. We have:*

$$\mathbb{P}_q^{\sigma,\tau}(\mathcal{L}_\eta) \leq \frac{1-\nu}{1-\eta} \ .$$

Proof. For any finite play u such that $v_\sigma(u) \leq \eta$, we define a strategy τ_u such that $v_{\sigma,\tau_u}(u) \leq \eta$. Consider now the strategy θ, defined by:

- if for any prefix u of x, $v_\sigma(u) > \eta$, $\theta(x) = \tau(x)$;
- if u is the shortest prefix of x such that $v(u) \leq \eta$, $\theta(x) = \tau_u(x)$.

It is clear that $\mathbb{P}_q^{\sigma,\tau}(\mathcal{L}_\eta) = \mathbb{P}_q^{\sigma,\theta}(\mathcal{L}_\eta)$, and that $\mathbb{P}_q^{\sigma,\theta}(\mathcal{W} \mid \mathcal{L}_\eta) \leq \eta$. As $\mathbb{P}_q^{\sigma,\theta}(\mathcal{W}) \geq \nu$, we get:

$$\nu \leq \eta \cdot \mathbb{P}_q^{\sigma,\tau}(\mathcal{L}_\eta) + (1 - \mathbb{P}_q^{\sigma,\tau}(\mathcal{L}_\eta)) \ .$$

Proposition 3.3 follows. □

Proposition 3.4 *Let q be a state of $\mathcal{Q}$, τ be a strategy for Adam, and η be a positive real number. We have:*

$$\mathbb{P}_q^{\sigma,\tau}(\mathcal{W} \mid \neg\mathcal{L}_\eta) = 1 \ .$$

Proof. For any integer n, we define the function φ_n, from $\Omega_q^{\sigma,\tau}$ to $[0,1]$ by $\varphi_n(\rho) = v_{\sigma,\tau}(\rho_0 \dots \rho_n)$. By Levy's law [Dur96],

$$\mathbb{P}_q^{\sigma,\tau}(\lim_{n\to\infty} \mathbb{E}_q^{\sigma,\tau}\varphi_n = 1_{\mathcal{W}}) = 1 \ .$$

Now, if $\rho \not\models \mathcal{L}_\eta$, we get,

$$\forall n, \varphi_n(\rho) = v_{\sigma,\tau}(\rho_0 \dots \rho_n) \geq v_\sigma(\rho_0 \dots \rho_n \geq \eta \ ,$$

so $\lim_{n\to\infty} \varphi_n(\rho) \neq 0$, $\mathbb{P}_q^{\sigma,\tau}(\mathcal{W} \mid \neg\mathcal{L}_\eta) = 1$, and Proposition 3.4 follows. □

3.1.2 Reset strategies

This suggests a way to improve a pure strategy with a "reset" procedure for a given η: if the value of the prefix drops below η, while the value of the current state is strictly greater than η, it is a better idea to forget the past and restart with a clean slate.

Definition 3.5 *The reset strategy of σ with respect to η, denoted by $\sigma_{\downarrow\eta}$, is a strategy with memory, whose memory states are plays of $\mathcal{A}$ consistent with σ. Its memory-update and next-move function are defined as follows:*

$$\sigma^{\mathtt{n}}_{\downarrow\eta}(w,q) = \begin{cases} \sigma(q) & \text{if } v_\sigma(wq) \leq \eta \wedge v_\sigma(q) > \eta \\ \sigma(wq) & \text{otherwise} \end{cases}$$

$$\sigma^{\mathtt{u}}_{\downarrow\eta}(w,q) = \begin{cases} q & \text{if } v_\sigma(wq) \leq \eta \wedge v_\sigma(q) > \eta \\ wq & \text{otherwise} \end{cases}$$

We define some shorthand notation to simplify the manipulation of reset-related events:

$$\mathcal{R}^i_\eta = \{\rho \in \Omega^{\sigma_{\downarrow\eta}} \mid \text{ there are } i \text{ resets in } \rho\} \ ,$$

$$\mathcal{R}^\infty_\eta = \bigcap_{i\in\mathbb{N}} \mathcal{R}^i_\eta \ .$$

Proposition 3.6 *Let q be a state of $\mathcal{Q}$ and τ be a strategy for Adam. We have:*

$$\mathbb{P}^{\sigma_{\downarrow\eta},\tau}_q(\mathcal{R}^\infty_\eta) = 0 \ .$$

Proof. Let $\nu = \min\{v_\sigma(s) \mid s \in \mathcal{Q} \wedge v_\sigma(s) > \eta\}$. The key observation is that:

$$\forall i, \mathbb{P}^{\sigma_{\downarrow\eta},\tau}_q(\mathcal{R}^{i+1}_\eta \mid \mathcal{R}^i_\eta) \leq \frac{1-\nu}{1-\eta} \ . \tag{3.3}$$

Indeed, after the ith reset, the token is in a state whose σ-value is greater than η (and thus greater or equal than ν), and Eve plays σ as if the play just started. Thus, by Proposition 3.3, the probability that the σ-value of the finite play in memory will ever drop below η is at most $\frac{1-\nu}{1-\eta}$, and (3.3) follows. This completes the proof of Proposition 3.6.

□

Proposition 3.7 *Let q be a state of $\mathcal{Q}$, and τ be a strategy for Adam. We have:*

$$\mathbb{P}_q^{\sigma_{\downarrow\eta},\tau}(\mathcal{W} \mid \exists i, \forall j \geq i, v_\sigma(\rho_j) > \eta) = 1 \ .$$

Proof. By Proposition 3.6, $\mathbb{P}_q^{\sigma_{\downarrow\eta},\tau}(\mathcal{R}_\eta^\infty) = 0$, so we can[3] consider only the plays with a finite number of resets. Let us consider the "final" memory after the play: it is a play consistent with σ which does not verify $\mathcal{L}_\eta$. By Proposition 3.4 it is winning for Eve with probability one, and Proposition 3.7 follows from the fact that $\mathcal{W}$ is prefix-independent.

□

3.1.3 Links

We can now use reset strategies in order to expose several links between the different notions of winning for prefix-independent games. Our first result is that, in prefix-independent games, there is no need to distinguish between positive and bounded regions, nor between limit-one and almost-sure regions:

Theorem 3.8 *Let $\mathcal{A}$ be an arena, and $\mathcal{W}$ a prefix-independent winning condition. We have:*

$$\begin{aligned} \mathrm{Win}_E^{\mathcal{W},\sim 1}(\mathcal{A}) &= \mathrm{Win}_E^{\mathcal{W},1}(\mathcal{A}) \\ \mathrm{Win}_E^{\mathcal{W},>0}(\mathcal{A}) &= \mathrm{Win}_E^{\mathcal{W},\gg 0}(\mathcal{A}) \end{aligned}$$

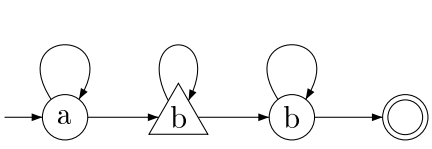
Proof. Let us start with the proof of $\mathrm{Win}_E^{\mathcal{W},\sim 1}(\mathcal{A}) = \mathrm{Win}_E^{\mathcal{W},1}(\mathcal{A})$. We choose a real number η such that $\forall q \notin \mathrm{Win}_E^{\mathcal{W},\sim 1}(\mathcal{A}), v(q) < \eta < 1$ and a strategy σ such that $\forall q \in \mathrm{Win}_E^{\mathcal{W},\sim 1}(\mathcal{A})$, $v_\sigma(q) > \eta$. The proof consists then in showing that $\sigma_{\downarrow\eta}$ is almost-sure in $\mathrm{Win}_E^{\mathcal{W},\sim 1}(\mathcal{A})$. By Proposition 3.1, neither Adam nor Random can leave $\mathrm{Win}_E^{\mathcal{W},\sim 1}(\mathcal{A})$, and by Definition 3.5, Eve does not: she could leave only if the value of the prefix was below η, and she would sooner reset her memory. So, for any play ρ starting in $\mathrm{Win}_E^{\mathcal{W},\sim 1}(\mathcal{A})$ and consistent with $\sigma_{\downarrow\eta}$, $\forall i, v_\sigma(\rho_i) > \eta$, and by Proposition 3.7, $\mathbb{P}_q^{\sigma_{\downarrow\eta},\tau}(\mathcal{W}) = 1$. The second equation, $\mathrm{Win}_E^{\mathcal{W},>0}(\mathcal{A}) = \mathrm{Win}_E^{\mathcal{W},\gg 0}(\mathcal{A})$ follows from the first applied to Adam, as $\neg\mathcal{W}$ also is prefix-independent:

$$\mathrm{Win}_E^{\mathcal{W},\gg 0}(\mathcal{A}) = \mathcal{Q} \setminus \mathrm{Win}_A^{\mathcal{W},\sim 1}(\mathcal{A}) = \mathcal{Q} \setminus \mathrm{Win}_A^{\mathcal{W},1}(\mathcal{A}) \supseteq \mathrm{Win}_E^{\mathcal{W},>0}(\mathcal{A})$$

[3] Yes we can!

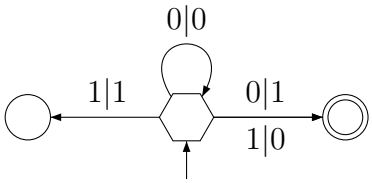
□

Theorem 3.8 does not hold for games with context-free conditions, infinite arenas, or concurrent moves: in each of the three games of Figure 3.1, the value of the initial state is 1, yet Eve has no almost-sure strategy.



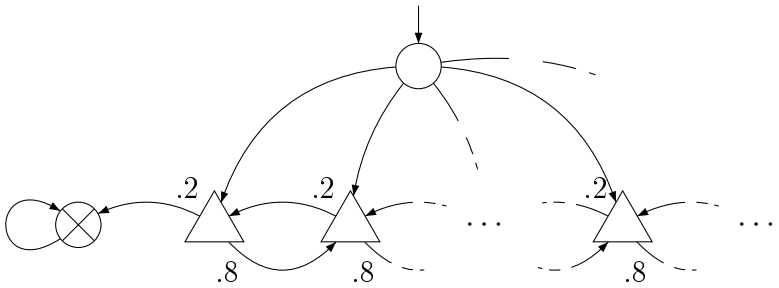
$\mathcal{W} = a^n b^n \circledcirc$

(a) Context-free condition



$\mathcal{W} = \text{Reach} \circledcirc$

(b) Concurrent moves



$\mathcal{W} = \text{Avoid} \otimes$

(c) Infinite arena

Figure 3.1: Limit-one is not almost-sure

Our second result is the positive-almost property of prefix-independent games.

Theorem 3.9 (Positive-almost property) *Let $\mathcal{A}$ be an arena, and $\mathcal{W}$ a prefix-independent winning condition. We have:*

$$\begin{aligned} \mathrm{Win}_E^{\mathcal{W},>0}(\mathcal{A}) = \mathcal{Q} &\implies \mathrm{Win}_E^{\mathcal{W},1} = \mathcal{Q} \\ \mathrm{Win}_E^{\mathcal{W},>0}(\mathcal{A}) \neq \emptyset &\implies \mathrm{Win}_E^{\mathcal{W},1} \neq \emptyset \end{aligned}$$

Proof. By Theorem 3.8, $\mathrm{Win}_E^{\mathcal{W},>0}(\mathcal{A}) = \mathcal{Q} \Rightarrow \mathrm{Win}_E^{\mathcal{W},\gg 0}(\mathcal{A}) = \mathcal{Q}$. As $\mathcal{Q}$ is finite, we can choose a real number ν such that $\forall q \in \mathcal{Q}, \nu < \mathbf{v}(q)$, and a strategy σ such that $\forall q \in \mathcal{Q}, v_\sigma(q) > \nu$. Let η be a real number such that $\eta < \nu$. For any play ρ of $\mathcal{A}$, $\forall i, v_\sigma(\rho_i) > \eta$, so Proposition 3.7 yields the almost sureness of $\sigma_{\downarrow\eta}$.

The second equation follows from the first and Theorem 3.8 to Adam:

$$\begin{aligned}
\mathrm{Win}_E^{\mathcal{W},>0}(\mathcal{A}) \neq \emptyset &\implies \mathrm{Win}_A^{\mathcal{W},1}(\mathcal{A}) \neq \mathcal{Q} \\
&\implies \mathrm{Win}_A^{\mathcal{W},>0}(\mathcal{A}) \neq \mathcal{Q} \\
&\implies \mathrm{Win}_A^{\mathcal{W},\gg 0}(\mathcal{A}) \neq \mathcal{Q} \\
&\implies \mathrm{Win}_E^{\mathcal{W},\sim 1}(\mathcal{A}) \neq \emptyset \\
&\implies \mathrm{Win}_E^{\mathcal{W},1}(\mathcal{A}) \neq \emptyset
\end{aligned}$$

This concludes the proof of Theorem 3.9. □

Although the formal prof is out of the scope of this work, Proposition 3.7 and a large part of the proof of Theorem 3.9 hold in the more general case of concurrent prefix-closed games. Theorem 3.8 does not, so the proof cannot be fully translated. Indeed, the games of Figure 3.1 are also counter-examples for Theorem 3.9. Still, we could derive a *universal bounded-almost property* and an *existential positive-limit property* for these games:

Claim 3.10 *Let $\mathcal{A}$ be a concurrent arena, and $\mathcal{W}$ a prefix-closed winning condition. We have:*

$$\begin{aligned}
\mathrm{Win}_E^{\mathcal{W},\gg 0}(\mathcal{A}) = \mathcal{Q} &\implies \mathrm{Win}_E^{\mathcal{W},1}(\mathcal{A}) = \mathcal{Q} \\
\mathrm{Win}_A^{\mathcal{W},>0}(\mathcal{A}) \neq \emptyset &\implies \mathrm{Win}_A^{\mathcal{W},\sim 1}(\mathcal{A}) \neq \emptyset
\end{aligned}$$

Last, but not least of our triptych is Theorem 3.11, which extends quantitative determinacy for prefix-independent games:

Theorem 3.11 (Qualitative determinacy) *Let $\mathcal{G} = (\mathcal{A}, \mathcal{W})$ be a prefix-independent $2\frac{1}{2}$-player game. We have:*

$$\mathrm{Win}_E^{\mathcal{W},>0}(\mathcal{A}) \cup \mathrm{Win}_A^{\mathcal{W},1}(\mathcal{A}) = \mathcal{Q}$$

$$\mathrm{Win}_E^{\mathcal{W},1}(\mathcal{A}) \cup \mathrm{Win}_A^{\mathcal{W},>0}(\mathcal{A}) = \mathcal{Q}$$

Proof. Theorem 3.11 follows directly from Theorem 3.8 and the quantitative determinacy of Borel games. □

By contrast with Theorem 3.8, we were not able to find counter-examples for natural extensions of Theorem 3.11. In particular, the three games of Figure 3.1 are qualitatively determined, and the qualitative determinacy of all Blackwell games is still an open problem.

3.2 Fix-points algorithms

We consider now the problems from an algorithmic point of view, and show how we can use some algorithms as recursive procedures in others. We consider first prefix-closed games, and introduce the notion of partial algorithm, which unifies some classical proof techniques used as much in pure games [Zie98, JPZ06, Hor07b] as in stochastic games [CdAH04, Hor07a] (3.2.1). In prefix-independent games, almost-sure algorithms are partial algorithms, which yields several results on the complexity of almost-sure and positive problems, as well as almost-sure and positive strategies.

3.2.1 Partial algorithms

In prefix-closed games, if Eve has a positive strategy from one state, her attractor to this state also belongs to her positive region. This is very usefull from an algorithmic point of view, since the remainder of the arena is a strictly smaller sub-arena, which allows recursive computations. Partial algorithms are oracles tailored specially to take advantage of this:

Definition 3.12 *A partial algorithm of Eve for a prefix-independent condition $\mathcal{W}$ over $\mathcal{C}$ takes as argument an arena over $\mathcal{C}$, and returns a region X such that:*

- $X \subseteq \mathrm{Win}_E^{\mathcal{W},>0}(\mathcal{A})$;
- $X = \emptyset \iff \mathrm{Win}_E^{\mathcal{W},>0}(\mathcal{A}) = \emptyset$.

Algorithm 3.1 uses a partial algorithm for $\mathcal{W}$-games as a parameter, and takes a $\mathcal{W}$-game as input. It returns the positive region of Eve, and the almost-sure region of Adam.

Parameter: A partial algorithm `partial` for $\mathcal{W}$-games
Input: A game $\mathcal{G} = (\mathcal{A}, \mathcal{W})$
Output: $(\mathrm{Win}_E^{\mathcal{W},>0}(\mathcal{A}), \mathrm{Win}_A^{\mathcal{W},1}(\mathcal{A}))$

$W_E = \emptyset$;
$\mathcal{B} = \mathcal{A}$;
while `partial`$(\mathcal{B}, \mathcal{W}) \neq \emptyset$ **do**
    $W_E \leftarrow W_E \cup \mathrm{Attr}_E(\texttt{partial}(\mathcal{B}), \mathcal{B})$;
    $\mathcal{B} \leftarrow \mathcal{B} \setminus \mathrm{Attr}_E(\texttt{partial}(\mathcal{B}), \mathcal{B})$;
end
return $(W_E, \mathcal{B})$

Algorithm 3.1: Fix-point algorithm

We can define a positive strategy σ for Eve based on a run of Algorithm 3.2: in the ith iteration, we denote the region $\mathrm{Win}_E^{\mathcal{W},1}(\mathcal{B})$ by X_i, an almost-sure strategy for Eve from X_i in $(\mathcal{B}, \mathcal{W})$ by σ_i, and the region $\mathrm{Attr}_E(X_i, \mathcal{B}) \setminus X_i$ by Y_i. The strategy σ uses a top-level memory which tells what is the lowest (*i.e.* earliest) i for which the token has already visited X_i, and plays according to σ_i, unless

- either the token is in a region Y_j with $j < i$: Eve plays her attractor to $X_j \cup \cup_{\ell<j}(X_\ell \cup Y_\ell)$ and resets — at each step — her memory to the initial memory state of σ_i;
- or the token is in a region X_j with $j < i$: Eve switches her top-level memory to j (and starts playing according to σ_j).

Notice that partial algorithms can also be used in 2-player games, to compute the (sure) regions of the players. In terms of complexity, the Algorithm 3.1 requires only the computation of $|\mathcal{Q}|$ attractors and partial algorithms:

Lemma 3.13 *Let $\mathcal{W}$ be a prefix-closed winning condition on $\mathcal{C}$. If there is a partial algorithm of Eve for $\mathcal{W}$-games whose time complexity on an arena $\mathcal{A}$ on $\mathcal{C}$ is $t(\mathcal{A})$ then Algorithm 3.1 computes the positive winning region of Eve (and thus the almost-sure region of Adam) in time $|\mathcal{Q}| \cdot (|\mathcal{T}| + t(\mathcal{A}))$.*

3.2.2 Switching algorithm

In prefix-independent games almost-sure algorithms are partial algorithms: it is clear that the almost-sure region of Eve is a subset of her positive region, and by the positive-almost property (Theorem 3.9, page 54), the former is empty if and only if the latter is. The fixpoint algorithm transforms then any almost-sure algorithm into a positive algorithm. Notice that $\neg\mathcal{W}$ is prefix-independent, so we can also transform a positive algorithm into an almost-sure algorithm: hence the name "switching algorithm".

Parameter: An algorithm computing $(\mathrm{Win}_E^{\mathtt{W},\mathtt{1}}, \mathrm{Win}_A^{\mathtt{W},>\mathtt{0}})$
Input: A game $\mathcal{G} = (\mathcal{A}, \mathcal{W})$
Output: $(\mathrm{Win}_E^{\mathcal{W},>\mathtt{0}}(\mathcal{A}), \mathrm{Win}_A^{\mathcal{W},\mathtt{1}}(\mathcal{A}))$
1 $W_E = \emptyset$;
2 $\mathcal{B} = \mathcal{A}$;
3 **while** $\mathrm{Win}_E^{\mathcal{W},\mathtt{1}}(\mathcal{B}) \neq \emptyset$ **do**
4 $\quad W_E \leftarrow W_E \cup \mathrm{Attr}_E(\mathrm{Win}_E^{\mathcal{W},\mathtt{1}}(\mathcal{B}), \mathcal{B})$;
5 $\quad \mathcal{B} \leftarrow \mathcal{B} \setminus \mathrm{Attr}_E(\mathrm{Win}_E^{\mathcal{W},\mathtt{1}}(\mathcal{B}), \mathcal{B})$;
6 **end**
7 **return** $(W_E, \mathcal{B})$

Algorithm 3.2: Switching algorithm

Theorem 3.14 follows directly from Lemma 3.13:

Theorem 3.14 *Let $\mathcal{W}$ be a prefix-independent winning condition on $\mathcal{C}$. If, for any arena $\mathcal{A}$ on $\mathcal{C}$, we can compute $\mathrm{Win}_E^{\mathcal{W},\mathtt{1}}(\mathcal{A})$ (and $\mathrm{Win}_A^{\mathcal{W},>\mathtt{0}}(\mathcal{A})$) in time $t(\mathcal{A})$, then we can compute $\mathrm{Win}_E^{\mathcal{W},>\mathtt{0}}(\mathcal{A})$ (and $\mathrm{Win}_A^{\mathcal{W},\mathtt{1}}(\mathcal{A})$) in time $|\mathcal{Q}| \cdot (|\mathcal{T}| + t(\mathcal{A}))$.*

Another consequence of the switching algorithm is that positive strategies require no more memory than almost-sure strategies. We define first the concept of residually almost-sure strategies:

Definition 3.15 *A strategy with memory is said to be residually almost sure if and only if for any state q and memory state m, $v_\sigma(q, m) = 1$.*

Lemma 3.16 *In any prefix-independent game $\mathcal{G}$, if there is a pure (resp. semi-randomised, resp. randomised) almost-sure strategy with memory Υ,*

there is a pure (resp. semi-randomised, resp. randomised) residually almost-sure strategy with memory Υ.

Proof. Let σ be a almost-sure strategy with memory. We build the residually almost-sure strategy ς on the same memory states. For any state q and memory state m, we have:

- if there is a state q_0 and a strategy τ such that $\mathbb{P}_{q_0}^{\sigma,\tau}(\mathrm{Reach}(q,m)) > 0$, $\varsigma(q,m) = \sigma(q,m)$;
- otherwise, $\varsigma(q,m) = \sigma(q,m_0)$.

If σ is pure or semi-randomised, it is clear for any state q and memory state m, (q,m) is reachable implies $v_\sigma(q,m) = 1$, as Adam could monitor the memory and start playing a counter-strategy when the value drops below one. If σ is a strategy with random memory, all he could do would be to try a prefix and guess Eve's memory, but this is enough to guarantee a positive probability of winning. Notice that this construction could be done for *bounded* pure and semi-randomised strategies, but not for bounded strategies with random memory. □

Residually almost-sure strategies can then be used as components for a positive strategy on $\mathrm{Win}_E^{\mathcal{W},>0}(\mathcal{A})$:

Theorem 3.17 *If there are almost-sure strategies with memory at most* Υ*, there are bounded strategies with memory at most* Υ.

Proof. By contrast with the prefix-closed case, there is no need to remember the smallest i such that X_i has been visited: the composition of residually almost-sure and attractor strategies is solely spatial:

- if q belongs to X_i, $\sigma(q,m) = \sigma_i(q,m)$;
- if q belongs to Y_i, $\sigma(q,m) = (\overrightarrow{a_E}(X_i \cup \bigcup_{\ell<j}(X_\ell \cup Y_\ell))(q), m)$.

□

3.3 Values and optimal strategies

The values of prefix-independent games $\mathcal{G} = (\mathcal{A}, \mathcal{W})$ are usually computed by hybrid algorithms, which merge a qualitative algorithm for $\mathcal{W}$-games with

a quantitative algorithm for reachability games. For example, one can guess the values of the states, and use a qualitative algorithm to check necessary and sufficient conditions on the value regions: see [CdAH05] for Rabin and Streett games, [Cha07c] for Muller games, and [CHH08] for finitary games. It is also possible to adapt the strategy improvement algorithm when one of the players has positional strategies: see [CJH04] for parity, and [CH06b] for Rabin games. Finally, the problem of prefix-independent $1\frac{1}{2}$-player games can be solved by computing first the almost-sure region, and then the values of the reachability game to this region [Cha07a].

We use our permutation algorithm as an universal converter: from an almost-sure algorithm for $\mathcal{W}$-games, we derive a meta-algorithm computing the values. As a matter of fact, the resulting algorithm is exceedingly close to the permutation algorithm for reachability games. The only difference is in the computation of the regions: instead of using deterministic attractors, Procedure Metaregion$(\mathcal{G}, \pi)$ computes almost-sure winning regions. Apart from that, Algorithm 3.3 is a carbon copy of Algorithm 2.9.

Input: A prefix-independent game G
Output: A partition of $\mathcal{Q}$ and the corresponding values
forall $\pi \in \mathcal{S}_k$ **do**
  $W \leftarrow$ `Metaregions`$(\mathcal{G}, \pi)$
  $v \leftarrow$ `Values`$(\mathcal{G}, \pi, W)$
  `self` $\leftarrow$ `Consistent`(π, v)
  `live` $\leftarrow$ `Live`$(\mathcal{G}, \pi, W)$
  if (`live` $\wedge$ `self`) **then**
    return (W, v)
  end
end

Algorithm 3.3: Permutation Algorithm for prefix-independent games

All the π-concepts of Section 2.2 can be adapted for prefix-independent conditions, most of the time with only minute differences. However, the intuitions behind these concepts are gone: regions can be empty, π-values may be different from the values of the π-strategies, and so forth. So, although regions, strategies, and values are defined for any permutation (3.3.1), they do not make much sense in general. On the other hand, the key properties of live and/or self-consistent permutations still hold *mutatis mutandis* (3.3.2).

We use them to prove the correctness of Algorithm 3.3, and study its complexity (3.3.3). A direct consequence is that optimal strategies need exactly as much memory as almost-sure strategies (3.3.4).

3.3.1 π-concepts for prefix independent conditions

As in the case of $2\frac{1}{2}$-player reachability games, our first step is to normalise the games we consider, by merging all the states of $\mathrm{Win}_A^{\mathcal{W},1}(\mathcal{A})$ into the sink state $\otimes$, and all the states of $\mathrm{Win}_E^{\mathcal{W},1}(\mathcal{A})$ into the target state $\circledcirc$. The winning condition $\mathcal{W}$ is modified accordingly: $\mathrm{Reach}\,\circledcirc \Longrightarrow \mathcal{W}$ and $\mathrm{Reach}\,\otimes \Longrightarrow \neg\mathcal{W}$.

The definition of the π-regions is also close to the case of reachability games:

- $W_\pi[k+1] = \{\circledcirc\}$
- $W_\pi[i] = \mathrm{Win}_E^{\mathcal{W}\vee\mathrm{Reach}(\cup_{j\geq i}\{\pi_j\}),1}(\mathcal{A}) \setminus \cup_{j>i} W_\pi[j]$
- $W_\pi[0] = \{\otimes\}$

Input: A prefix-independent game $\mathcal{G}$ and a permutation π
Output: The π-regions of $\mathcal{G}$

```
1 W[k+1] ← {◎}
2 W[0] ← {⊗}
3 for (i = 1, i ≤ k, i++) do
4 |  W[i] ← Win_E^{W∨Reach(∪_{j≥i}{π_j}),1}(A) \ ∪_{j>i} W_π[j]
5 end
6 return W
```

Procedure `Metaregions`$(\mathcal{G}, \pi)$

However, we need to compute an almost-sure region, in lieu of a deterministic attractor: a random state π_i may thus belong to a region $W_\pi[j]$ with $i < j$ (but not $i > j$). In this case, the region $W_\pi[i]$ is empty.

Eve's π-strategy σ_π is a spatial combination of residually almost-sure strategies: in $W_\pi[i]$, she plays a residually almost-sure strategy with respect to the objective $\mathcal{W} \vee \mathrm{Reach}(\cup_{j\geq i}\{\pi_j\})$.

Adam's π-strategy τ_π is a spatial combination of reset strategies: in $W_\pi[i]$, he plays a bounded strategy of value η with respect to the objective $\mathcal{W} \vee$

$\mathrm{Reach}(\cup_{j\geq i}\{\pi_j\}))$, which is reset when the value of the prefix drops below $\frac{\eta}{2}$. By Proposition 3.7, if any region is visited infinitely often, Adam wins with probability one, and Proposition 3.18 follows:

Proposition 3.18 *Let π be a permutation, and τ_π be the corresponding π-strategy for Adam. For any initial state q and strategy σ of Eve, we have:*

$$\mathbb{P}_q^{\sigma,\tau_\pi}(\neg\mathcal{W} \vee \mathrm{Reach}\circledcirc) = 1$$

The vector of π-values for the states of $\mathcal{Q}_R$ is computed from the $\frac{1}{2}$-player reachability game $\mathfrak{G}^\pi$ defined as follows:

- $\mathfrak{Q} = \mathcal{Q}_R \cup \{\otimes\} \cup \{\circledcirc\}$
- $\mathfrak{d}(\pi_i)(\otimes) = \delta(\pi_i)(\otimes)$
- $\mathfrak{d}(\pi_i)(\circledcirc) = \delta(\pi_i)(\circledcirc)$
- $\mathfrak{d}(\pi_i)(\pi_j) = \delta(\pi_i)(W_\pi[j])$

For any $1 \leq i \leq k$, $\mathfrak{v}_\pi[i]$ is the value of π_i in $\mathfrak{G}^\pi$. The associated values for the states are defined by: $q \in W_\pi[i] \Rightarrow \mathfrak{v}_\pi(q) = \mathfrak{v}_\pi[i]$. Notice that if $\pi_i \in W_\pi[j]$, $\mathfrak{v}_\pi(\pi_i) = \mathfrak{v}_\pi[j]$, and not necessarily $\mathfrak{v}_\pi(\pi_i) = \mathfrak{v}_\pi[i]$. By contrast with the case of reachability game, it is not true in general that $\mathfrak{v}_\pi = v_{\sigma_\pi,\tau_\pi}$.

3.3.2 Liveness and self-consistency

The notions of (self-)consistency and liveness need no tinkering from reachability: Definitions 3.19, 3.21, and 3.23 are carbon copies of Definitions 2.27, 2.25, and 2.26. In the same way, we prove equivalents of the key properties: Propositions 3.20 and 3.22 replace Propositions 2.28 and 2.31. An extra proposition, Proposition 3.24, deals with displaced random states.

Definition 3.19 *A permutation π over the set $\mathcal{Q}_R$ is live if and only if for any $1 \leq i \leq k$, $\delta(\pi_i)(\cup_{j>i}W_\pi[j]) > 0$.*

Proposition 3.20 *Let π be a live permutation, and σ_π be the corresponding π-strategy for Eve. For any strategy τ of Adam, we have:*

$$\mathbb{P}_q^{\sigma_\pi,\tau}(\mathcal{W} \vee \mathrm{Reach}\otimes) = 1$$

Proof. Let q be a state of $\mathcal{Q}$, τ be a strategy for Adam, and $\texttt{Stuck}(i)$ be the event "$\mathrm{Inf}(\rho) \cap W_\pi[i] \neq \emptyset \wedge \mathrm{Inf}(\rho) \cap \{\pi_i, \ldots, \pi_k\} = \emptyset$". By definition of σ_π, for any $1 \leq i \leq k$, we have $\mathbb{P}_q^{\sigma_\pi,\tau}(\texttt{Stuck}(i) \wedge \neg\mathcal{W}) = 0$. By the liveness property, for any $1 \leq i \leq k$, we have $\mathbb{P}_q^{\sigma_\pi,\tau}(\pi_i \in \mathrm{Inf}(\rho) \wedge \mathrm{Inf}(\rho) \cap \cup_{j>i} W_\pi[j] = \emptyset) = 0$. Proposition 3.20 follows. □

Definition 3.21 *A permutation π is consistent with a set of values v if and only if for any two states π_i and π_j in $\mathcal{Q}_R$, $i < j \Rightarrow v(\pi_i) \leq v(\pi_j)$.*

Proposition 3.22 *There is a live permutation consistent with the values of $\mathcal{G}$.*

Proof. The permutation is chosen starting from π_k, and going down to π_1. At each step, the state π_i is chosen among the ones such that:

- $\mathbf{v}(\pi_i) = \max\{\mathbf{v}(q) \mid q \in \mathcal{Q}_R \setminus \{\pi_{i+1}, \ldots, \pi_k\}\}$
- $\delta(\pi_i)(\cup_{j>i} W_\pi[i]) > 0$

There is always such a state: otherwise, the set X of states whose value is maximal in $\mathcal{Q} \setminus \cup_{j>i} W_\pi[j]$ would be a trap for Adam, and the states of X have value 1, in contradiction with the "normalised" hypothesis. □

Definition 3.23 *A permutation π is self-consistent if and only if it is consistent with $\boldsymbol{\mathbf{v}}_\pi$: for any two states π_i and π_j in $\mathcal{Q}_R$, $i < j \Rightarrow \boldsymbol{\mathbf{v}}_\pi[i] \leq \boldsymbol{\mathbf{v}}_\pi[j]$.*

Proposition 3.24 *Let π be a self-consistent permutation, and i and j be two integers such that $i < j$ and $\pi_i \in W_\pi[j]$. Then for all ℓ such that $\delta(\pi_i)(W_\pi[\ell]) > 0$, $\boldsymbol{\mathbf{v}}_\pi[i] = \boldsymbol{\mathbf{v}}_\pi[j] = \boldsymbol{\mathbf{v}}_\pi[\ell]$.*

Proof. As $\pi_i \in W_\pi[j]$, $\delta(\pi_i)(W_\pi[\ell]) > 0 \Rightarrow \ell \geq j$. By self-consistency, $\ell \geq j \Rightarrow \boldsymbol{\mathbf{v}}_\pi[\ell] \geq \boldsymbol{\mathbf{v}}_\pi[j]$, so $\boldsymbol{\mathbf{v}}_\pi[i] \geq \boldsymbol{\mathbf{v}}_\pi[j]$. But, again by self-consistency, $\boldsymbol{\mathbf{v}}_\pi[i] \leq \boldsymbol{\mathbf{v}}_\pi[j]$. So $\boldsymbol{\mathbf{v}}_\pi[i] = \boldsymbol{\mathbf{v}}_\pi[j]$, and, $\delta(\pi_i)(W_\pi[\ell]) > 0 \Rightarrow \boldsymbol{\mathbf{v}}_\pi[i] = \boldsymbol{\mathbf{v}}_\pi[\ell]$. Proposition 3.24 follows. □

3.3.3 Correctness of Algorithm 3.3

Now that all the pieces are there, we can proceed to the main theorem:

Theorem 3.25 *Let $\mathcal{G} = (\mathcal{A}, \mathcal{W})$ be a prefix-independent game. A run of Algorithm 3.3 on $\mathcal{G}$ terminates and returns the values of the states.*

Proof. Theorem 3.25 is proved as Theorem 2.29, by two independent lemmas:

- there is a live and self-consistent permutation (Lemma 3.26);
- if a permutation π is live and self-consistent, then v_π are the optimal values for the regions W_π (Lemma 3.27).

□

Lemma 3.26 *There is a live and self-consistent permutation.*

Proof. The first part of this proof was to show that there is a live permutation π consistent with the values of the game (Proposition 3.22). The point is now to prove that the π-values *are* the values of $\mathcal{G}$. These values are constant over the π-regions:

$$q \in \mathrm{Win}_E^{\mathcal{W} \vee \mathrm{Reach}\, \mathtt{X}, \mathtt{1}}(\mathcal{A}) \Rightarrow \mathbf{v}(q) \geq \min\{\mathbf{v}(q) \mid q \in X\}$$

$$q \notin \mathrm{Win}_E^{\mathcal{W} \vee \mathrm{Reach}\, \mathtt{X}, \mathtt{1}}(\mathcal{A}) \Rightarrow \mathbf{v}(q) \leq \max\{\mathbf{v}(r) \mid r \in \mathcal{Q}_R \setminus X\}$$

Thus, the relations between the values of the π-regions which follow from (3.1) are exactly the relations between the values of the states in $\mathfrak{G}^\pi$. So $\mathbf{v} = \mathbf{v}_\pi$, and Lemma 3.26 follows. □

Lemma 3.27 *If π is a live and self-consistent permutation, then the π-strategies are optimal and $\mathbf{v}_\pi = \mathbf{v}$.*

Proof. We fix an initial state q and prove independently that $v_{\sigma_\pi}(q) \geq \mathbf{v}_\pi(q)$ and $v_{\tau_\pi}(q) \leq \mathbf{v}_\pi(q)$. Let τ be a strategy for Adam. We define an "expected π-value" function f by $f(n) = \sum_{s \in \mathcal{Q}} \mathbf{v}_\pi(s) \cdot \mathbb{P}_q^{\sigma_\pi, \tau}(\rho_n = s)$. This function is waxing:

- a move of Eve consistent with σ_π remains in the same π-region;
- a move of Adam sends the token to a state with greater or equal π-value (self-consistency);
- the value of a random state π_i such that $\pi_i \in W_\pi[i]$ is the average value of its successors;

- a random state π_i such that $\pi_i \in W_\pi[j]$ and $i < j$ sends the token to a state with equal π-value (Proposition 3.24).

Thus, $f(n) \leq f(n+1)$. Furthermore, as $f(n) \leq 1 - \mathbb{P}_q^{\sigma_\pi,\tau}(\rho_n = \otimes)$, we get $\lim f(n) \leq 1 - \mathbb{P}_q^{\sigma_\pi,\tau}(\mathrm{Reach}\,\otimes)$. By Proposition 3.20, $\mathbb{P}_q^{\sigma,\tau_\pi}(\mathrm{Reach}\,\otimes) = 1 - v_{\sigma_\pi,\tau}(q)$, so $\mathbf{v}_\pi(q) = f(0) \leq \lim_{n\to\infty} f(n) \leq v_{\sigma_\pi,\tau}(q)$. As τ is an arbitrary strategy for Adam, we get $v_{\sigma_\pi} \geq \mathbf{v}_\pi$.

Likewise, for a strategy σ for Eve, we define the function g by $g(n) = \sum_{s\in Q} \mathbf{v}_\pi(s) \cdot \mathbb{P}_q^{\sigma,\tau_\pi}(\rho_n = s)$. This function is waning:

- a move of Eve sends the token to a state with lower or equal π-value (self-consistency);
- a move of Adam consistent with τ_π remains in the same π-region;
- the value of a random state π_i such that $\pi_i \in W_\pi[i]$ is the average value of its successors;
- a random state π_i such that $\pi_i \in W_\pi[j]$ and $i < j$ sends the token to a state with equal π-value (Proposition 3.24).

Thus, $g(n) \geq g(n+1)$. Furthermore, as $g(n) \geq \mathbb{P}_q^{\sigma,\tau_\pi}(\rho_n = \circledcirc)$, we get $\lim g(n) \geq \mathbb{P}_q^{\sigma,\tau_\pi}(\mathrm{Reach}\,\circledcirc)$. By Proposition 3.18, $\mathbb{P}_q^{\sigma,\tau_\pi}(\mathrm{Reach}\,\circledcirc) = v_{\sigma,\tau_\pi}(q)$, so $\mathbf{v}_\pi(q) = g(0) \geq \lim_{n\to\infty} g(n) \geq v_{\sigma,\tau_\pi}(q)$. As σ is an arbitrary strategy for Eve, we get $v_{\tau_\pi} \leq \mathbf{v}_\pi$.

It follows that $v_{\sigma_\pi} = v_{\tau_\pi} = \mathbf{v}_\pi$, so σ_π and τ_π are optimal strategies, and $\mathbf{v}_\pi = \mathbf{v}$. This concludes the proof of Lemma 3.27. □

Theorems 3.28 and 3.29 are direct consequences of Theorem 3.25:

Theorem 3.28 *Let $\mathcal{G}$ be a prefix-independent game. If there is an algorithm computing the almost-sure region of Eve in time $t(\mathcal{G})$, then Algorithm 3.3 computes the values of $\mathcal{G}$ in time $|Q_R + 1|! \cdot (|\delta| + t(\mathcal{G}))$.*

Proof. In Procedure `Metaregions`, the time-consuming operations are the computation of the almost-sure regions (in time $t(\mathcal{G})$) and the computation of the attractor (in time $|\delta|$). Each is done $|sr|$ times in each call, and in the worst case, Algorithm 3.3 calls Procedure `Metaregions` $|Q_R|!$ times. Theorem 3.28 follows. □

Theorem 3.29 *Let $\mathfrak{W}$ be a class of prefix-independent winning conditions. If the qualitative problems of $\mathfrak{W}$-games belong to the complexity class $\mathcal{K}$, then the quantitative problems belong to the classes $\mathtt{NP}^{\mathcal{K}}$ and co-$\mathtt{NP}^{\mathcal{K}}$.*

Proof. There is a non-deterministic variant of Algorithm 3.3 which guesses the correct permutation instead of searching for it. The verification can then be done in polynomial time with $|\mathcal{Q}_R|$ calls to a $\mathcal{K}$-oracle. □

3.3.4 Optimal strategies

One of the assets of Algorithm 3.3 is that we can derive optimal strategies from a live and self-consistent permutation, so Theorem 3.30 follows from Lemma 3.27:

Theorem 3.30 *Prefix-independent games are optimally determined.*

Furthermore, the strategy σ_π is defined as a spatial composition of residually almost-sure strategies, and does not use more memory than its components:

Theorem 3.31 *Let $\mathcal{W}$ be a prefix-independent winning condition. If Eve has pure (resp. semi-randomised, randomised) qualitative strategies with finite memory Υ, she has pure (resp. semi-randomised, randomised) optimal strategies with finite memory Υ.*

Notice that Theorem 3.31 does not hold without the hypothesis that $\mathcal{W}$ is prefix-independent, even for regular winning conditions: a counter-example is the weak parity game of Figure 3.2.

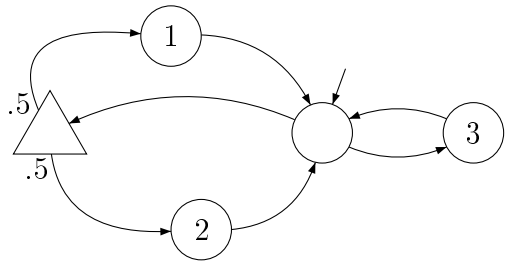


Figure 3.2: Optimal strategies require memory in weak parity games

In this game, the value of the initial state is .5: if Eve sends the token once to the left, and then always to the right, the lowest occurring colour has equal chances to be 1 or 2. However, this value cannot be achieved by means of a positional strategy:

- if Eve has a positive probability to send the token to the left, the lowest occurring colour is almost surely 1;
- if Eve never sends the token to the right, the lowest occurring colour is surely 3.

There are positional almost-sure winning strategies for both players in $2\frac{1}{2}$-player weak parity games [GZ05]. Optimal strategies for weak parity games with d colours may require up to $d-1$ memory states, even in $1\frac{1}{2}$-player games.

3.4 Valediction

We showed that prefix-independent games are optimally determined, and provided a general algorithm computing the values of any prefix-independent games with a single non-deterministic guess and a qualitative algorithm.

The determinacy result is very sensitive to each of our hypotheses, as demonstrated by Figure 3.1. However, the quantitative determinacy of Borel games may still be extended, by the qualitative determinacy to begin with, and by similar questions for arbitrary values.

Chapter 4

Muller Games

"You can't have a strategy against telepaths: you have to act randomly. You have to not know what you're going to do next. You have to shut your eyes and run blindly. The problem is: how can you randomise your strategy, yet move purposefully towards your goal?"

Solar Lottery
Philip K. Dick

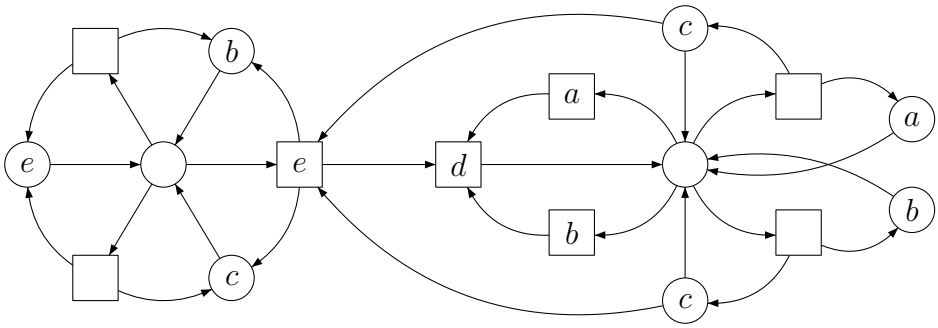
With this chapter, we go back to the origins of infinite games: Church's original synthesis problem amounts to solving Muller games. Muller games subsume the other classical normal forms of regular games such as parity, Rabin, and Streett games.

The Muller condition is prefix-independent, so they provide us with an application of the results of Chapter 3.

We apply our results on prefix-independent conditions to the setting of Muller games, where the winner depends only on the states that are visited infinitely often. They subsumes other classical normal forms of regular games such as parity, Rabin, and Streett games

The qualitative problems of Muller games can usually[1] be solved in polynomial space [McN93, NRY96]. However, this complexity is not necessarily tight, depending on how the winning condition is represented. As in Chapter 2, we present in Figure 4.1 an example of Muller game to demonstrate several interesting notions.

[1] As long as deciding the winner of a limit set is in PSPACE.



$\mathfrak{F} = \{\{a,b,c,d\},\{a,b,c\},\{a,b\},\{b,c,d,e\},\{b,c,e\},\{b,d,e\},\{c,d,e\},\{e\}\}$

Figure 4.1: Muller game example: the game $\mathfrak{G} = (\mathfrak{A}, \mathfrak{F})$

We first present, in Section 4.1, a polynomial algorithm for the qualitative problems of explicit Muller games. Section 4.2 describes the notion of Zielonka tree of a coloured Muller condition, and shows how to use it to define a reduction to parity conditions. This tree is again central in Section 4.3, which defines a recursive PSPACE algorithm for $2\frac{1}{2}$-player Muller games. The analysis of this algorithm also provide upper bounds in memory for pure as well as randomised strategies. We use then the Zielonka *DAG* in Section 4.4 to show that these bounds are tight.

4.1 Explicit games

Our first result about Muller games is a polynomial algorithm computing the winning regions of *explicit* Muller games. The explicit representation of a Muller condition $\mathcal{F}$ consists simply in the sequence $\mathcal{F}_1 \cdots \mathcal{F}_\ell$ of all the sets in $\mathcal{F}$. Notice that this definition precludes the use of a (non-trivial) colouring function: the winner problem of coloured Muller games, which we study in the next sections, is PSPACE-complete.

We introduce the notions of semi-alternation and sensibleness for explicit Muller games, and show that any explicit game can be translated in polynomial time into a semi-alternating and sensible game (4.1.1). We use then these notions to describe a polynomial algorithm for explicit Muller games (4.1.2).

4.1.1 Normal form

We first define three properties of explicit Muller games. A game is:

1. semi-alternating if there is no transition between two states of Adam (but there can be one between two states of Eve);
2. sensible if each set in $\mathcal{F}$ is an end-component of $\mathcal{A}$;
3. ordered for inclusion if $i < j \Rightarrow \mathcal{F}_i \not\supseteq \mathcal{F}_j$.

Our algorithm for explicit Muller games, Algorithm 4.1, relies on the fact that its input satisfies these three properties. However, this does not restrict the generality of our result, since any explicit Muller game can be transformed in polynomial time into an equivalent semi-alternating, sensible, and ordered game of polynomial size. The semi-alternation transformation consists in replacing each state $q \in \mathcal{Q}_A$ of Adam by a pair of states $r \in Q_E, s \in Q_A$, as in Figure 4.2. Each set containing q in the winning condition is modified accordingly: $\mathcal{F} \leftarrow (\lambda q.(r, s))\mathcal{F}$. This is where the classical alternation transformation fails: adding a state to each transition leads to an exponential blow-up in the size of the winning condition.

(a) Original arena (b) Semi-alternating arena

Figure 4.2: Semi-alternating arena construction

A game can be made sensible by removing from $\mathcal{F}$ all the sets that are not end-components of $\mathcal{A}$: by Lemma 1.3, whatever the strategies of Eve and Adam, the limit of the play is an end-component with probability one. This modification is thus transparent with respect to stochastic concepts —the sure and heroic regions do change, however. Finally, ordering the sets for inclusion can be done in quadratic time.

The games of the form $(\mathcal{A}, \{\mathcal{Q}\})$, where Eve wins if and only if the token visits all the states infinitely often, play an important part in our solution to explicit Muller games. These games, which have also been studied in routing problems [DK00, IK02], are easy to solve and there is always only one winner in the whole game:

Proposition 4.1 *Let $\mathcal{A}$ be a $2\frac{1}{2}$-player arena, and $\mathcal{G}$ be the explicit Muller game $(\mathcal{A}, \{\mathcal{Q}\})$. Either, for any state $q \in \mathcal{Q}$, Eve's attractor to q is equal to $\mathcal{Q}$, and Eve wins almost-surely everywhere in $\mathcal{G}$, or there is a state $q \in \mathcal{Q}$ such that $\mathrm{Attr}_E(\{q\}, \mathcal{A}) \neq \mathcal{Q}$, and Adam wins surely everywhere in $\mathcal{G}$.*

Proof. In the first case, Eve can win almost-surely by playing the uniform strategy $\mathtt{uni}_{\mathcal{A}}$: infinite visits to all the states of $\mathcal{Q}$ ensues [CdAH04]. In the second case, Adam can win surely with any trapping strategy out of $\mathrm{Attr}_E(\{q\}, \mathcal{A})$: if the token ever gets out of $\mathrm{Attr}_E(\{q\}, \mathcal{A})$, it never goes back. □

Following Proposition 4.1, we say that "Eve wins $(\mathcal{A}, \{\mathcal{Q}\})$" if she wins almost-surely from any state of $\mathcal{Q}$, and that "Adam wins $(\mathcal{A}, \{\mathcal{Q}\})$" if he wins surely from any states of $\mathcal{Q}$. This could be misleading if we were to consider the sure region of Eve or the heroic region of Adam, but we do not.

4.1.2 Algorithm

Our algorithm takes as input a semi-alternating, sensible $2\frac{1}{2}$-player explicit Muller game whose winning condition is ordered for inclusion; it returns the positive region of Eve and the almost-sure region of Adam. Each set in $\mathcal{F}$ is considered at most once, starting with the (smallest) set $\mathcal{F}_1$. At each step, the operation of a set $\mathcal{F}_i$ modifies the arena and the winning condition in one of the following ways:

If Adam wins $(\mathcal{A}_{|\mathcal{F}_i}, \{\mathcal{F}_i\})$**,** $\mathcal{F}_i$ is removed from $\mathcal{F}$.

If Eve wins $(\mathcal{A}_{|\mathcal{F}_i}, \{\mathcal{F}_i\})$**, and** $\mathcal{F}_i$ **is a trap for Adam in** $\mathcal{A}$**,** Eve's attractor to $\mathcal{F}_i$ in $\mathcal{A}$, $\mathrm{Attr}_E(\mathcal{F}_i, \mathcal{A})$, is removed from $\mathcal{A}$ (and added to the winning region of Eve), and all the sets intersecting $\mathrm{Attr}_E(\mathcal{F}_i, \mathcal{A})$ are removed from $\mathcal{F}$.

If Eve wins $(\mathcal{A}_{|\mathcal{F}_i}, \{\mathcal{F}_i\})$**, and** $\mathcal{F}_i$ **is not a trap for Adam in** $\mathcal{A}$**,** a new state $\mathbb{F}_i$, described in Figure 4.3, is added to $\mathcal{A}$ with the following attributes:

- $\mathbb{F}_i$ is a state of Adam;

- the predecessors of $\mathbb{F}_i$ are all the states of Eve in $\mathcal{F}_i$;
- the successors of $\mathbb{F}_i$ are the successors outside $\mathcal{F}_i$ of the states of Adam in $\mathcal{F}_i$.

Furthermore, the state $\mathbb{F}_i$ is added to all the supersets of $\mathcal{F}_i$ in $\mathcal{F}$, and $\mathcal{F}_i$ itself is removed from $\mathcal{F}$.

$\mathcal{M} = \{\{a, b\}, \{a, b, c\}\}$ $\mathcal{M} = \{\{a, b, c, \{\mathtt{a}, \mathbb{b}\}\}\}$

(a) Before (b) After

Figure 4.3: Removal of a set in an explicit Muller condition

The important case, from an intuitive point of view, is the last one: it corresponds to a "threat" of Eve to win by visiting exactly the states of $\mathcal{F}_i$. Adam has to answer by getting out, but he can choose his exit from any of his states. Notice that it would not do to simply replace the whole region $\mathcal{F}_i$ by the state $\mathbb{F}_i$: as in Figure 4.3, Adam may be able to avoid a state of $\mathcal{F}_i$ in a larger arena, even if he is incapable of doing so in $\mathcal{A}_{|\mathcal{F}_i}$.

As only one state is added each step, the number of states in the game is bounded by $|\mathcal{A}| + |\mathcal{F}|$. The whole procedure is described as Algorithm 4.1.

In the proof of correctness, we use `typewriter` fonts to denote the modified arena and condition, and *calligraph* fonts to denote the original game. Furthermore, we denote by $\mathcal{F}_{|\mathcal{F}_i}$ the intersection of $\mathcal{F}$ and $\mathcal{P}(\mathcal{F}_i)$, *i.e.* the sets of $\mathcal{F}$ that are also subsets of $\mathcal{F}_i$. We can now proceed to the three main lemmas:

Input: An explicit Muller game $(\mathcal{A}, \mathcal{F})$
Output: $\mathrm{Win}_E^{\mathcal{F},>0}(\mathcal{A})$ and $\mathrm{Win}_A^{\mathcal{F},1}(\mathcal{A})$

```
1  A = (Q, Q_E, Q_A, Q_R, T, p) ← A = (Q, Q_E, Q_A, Q_R, T, p);
2  F ← F;
3  W_E ← ∅;
4  while F ≠ ∅ do
5      F_i ← pop(F);
6      if Eve wins (A_|F_i, {F_i}) then
7          if F_i is a trap for Adam in A then
8              remove Attr_E(F_i, A) from A and add it to W_E;
9              remove all the sets intersecting Attr_E(F_i, A) from F;
10         else
11             add a state 𝔽_i to Q_A;
12             add transitions from F_i ∩ Q_E to 𝔽_i;
13             add transitions from 𝔽_i to T(F_i ∩ Q_A) \ F_i;
14             add 𝔽_i to all the supersets of F_i in F;
15         end
16     end
17 end
18 return W_E ∩ Q, Q ∩ Q
```

Algorithm 4.1: Polynomial algorithm for explicit Muller games

Lemma 4.2 *If, in the course of a run of Algorithm 4.1, the game* $(\mathtt{A}_{|\mathtt{F}_i}, \{\mathtt{F}_i\})$ *is winning for Eve at line 6, then Eve wins almost-surely everywhere in the game* $(\mathcal{A}_{|\mathcal{F}_i}, \mathcal{F}_{|\mathcal{F}_i})$.

Proof. Let $\mathcal{H}^1, \ldots, \mathcal{H}^k = \mathcal{F}_i$ be the sets of $\mathcal{F}_{|\mathcal{F}_i}$ such that $(\mathtt{A}_{|\mathtt{H}^j}, \{\mathtt{H}^j\})$ was winning for Eve in the run of Algorithm 4.1. Notice that $\mathcal{F}_i$ itself is one of these states, say $\mathcal{H}^k$. The σ^j's denote her corresponding almost-sure strategies. We build a strategy σ for Eve in $\mathcal{A}_{|\mathcal{F}_i}$, whose memory states are stacks of pairs $(\mathcal{H}^j, \rho^j)$. At any time, ρ^j is a play of $\mathtt{A}_{|\mathtt{H}^j}$ which can be extended by the current state q. The initial memory state is $(\mathcal{H}^k, \varepsilon)$, and the operation of σ when the memory state is $(\mathcal{H}^j, w)$ and the current state is q is described below:

1. If $q \notin \mathcal{H}^j$, the top pair is removed, and the procedure restarts at step 1 with the new memory. Notice that it may involve further pops if q still does not belong to the top set.

2. If q is a state of Eve, and $\sigma^j(wq)$ is a new state $\mathbb{H}^h$, the memory is modified as follows: w becomes $wq\mathbb{H}^h$, and a new pair $(\mathcal{H}^h, \varepsilon)$ is pushed at the top of the stack. The procedure restarts at step 2. with the new memory. Notice that it may involve further pushes if $\sigma^h(q)$ is also a new state.

3. The new memory state is $(\mathcal{H}^j, wq)$; if q belongs to Eve, she plays $\sigma^j(wq)$.

We claim that σ is almost-sure for Eve in the game $(\mathcal{A}_{|\mathcal{F}_i}, \mathcal{F}_{|\mathcal{F}_i})$. Let ρ be a play consistent with σ, and $\mathcal{H}^j$ the highest set that is never unstacked. We denote by ρ^j the (infinite) limit of the "play" part. As ρ^j is consistent with σ^j, $\mathrm{Inf}(\rho^j) = \mathtt{H}^j$ with probability one. Furthermore, $\mathrm{Inf}(\rho) \supseteq \mathrm{Inf}(\rho^j) \cap \mathcal{Q}$ and $\mathrm{Inf}(\rho) \subseteq \mathcal{H}^j$. So, $\mathrm{Inf}(\rho) = \mathcal{H}^j$ with probability one, and Lemma 4.2 follows. □

For Adam, the problem is a little more complex: we need two lemmas, whose proofs are mutually recursive:

Lemma 4.3 *If, in the course of a run of Algorithm 4.1, the game* $(\mathtt{A}_{|\mathtt{F}_i}, \{\mathtt{F}_i\})$ *is winning for Adam at line 6, then Adam wins surely everywhere in the game* $(\mathcal{A}_{|\mathcal{F}_i}, \mathcal{F}_{|\mathcal{F}_i})$.

Lemma 4.4 *If, in the course of a a run of Algorithm 4.1, the game* $(\mathtt{A}_{|\mathtt{F}_i}, \{\mathtt{F}_i\})$ *is winning for Eve at line 6, then Adam wins surely everywhere in the game* $(\mathcal{A}_{|\mathcal{F}_i}, \mathcal{F}_{|\mathcal{F}_i} \setminus \{\mathcal{F}_i\})$.

Proof. We start with the (simpler) proof of Lemma 4.4. Let $\mathcal{H}^1, \ldots, \mathcal{H}^k$ be the maximal sets, with respect to inclusion, of $\mathcal{F}_{|\mathcal{F}_i}$. There is a sure strategy τ^j for Adam in each $\mathcal{H}^j$: if Adam won $(\mathtt{A}_{|\mathtt{H}^j}, \{\mathtt{H}^j\})$, it is a winning strategy for the game $(\mathcal{A}_{|\mathcal{H}^j}, \mathcal{F}_{|\mathcal{H}^j})$ (recursive use of Lemma 4.3); if Eve won $(\mathtt{A}_{|\mathtt{H}^j}, \{\mathtt{H}^j\})$, it is a strategy for the game $(\mathcal{A}_{|\mathcal{H}^j}, \mathcal{F}_{|\mathcal{H}^j} \setminus \mathcal{H}^j)$ (recursive use of Lemma 4.4). The strategy τ for Adam in $(\mathcal{A}_{|\mathcal{F}_i}, \{\mathcal{F}_{|\mathcal{F}_i}\})$ uses k top-level memory states to switch between the $\{\tau^j\}_{1 \leq j \leq k}$. Adam remains in a top-level memory state j only as long as the token is in $\mathcal{H}^j$. As soon as it gets out, he updates it to $(j \mod k) + 1$. His actions when the top-level memory state is j are described below:

- if he won $(\mathtt{A}_{|\mathtt{H}^j}, \{\mathtt{H}^j\})$, he plays τ^j;
- if Eve won $(\mathtt{A}_{|\mathtt{H}^j}, \{\mathtt{H}^j\})$, he plays τ^j unless he can get out of $\mathcal{H}^j$.

We claim that τ is surely winning for Adam in $(\mathcal{A}_{|\mathcal{F}_i}, \mathcal{F}_{|\mathcal{F}_i})$. Any play ρ consistent with τ falls in exactly one of the three following categories:

- The top-level memory of τ is not ultimately constant; thus $\mathrm{Inf}(\rho)$ is not included in any of the $\mathcal{H}^j$'s, and ρ is winning for Adam.
- The top-level memory of τ is ultimately constant at j, and $(\mathtt{A}_{|\mathtt{H}^j}, \{\mathtt{H}^j\})$ was winning for Adam; ρ is ultimately a play of $\mathcal{A}_{|\mathcal{H}^j}$ consistent with τ^j, so ρ is winning for Adam.
- the top-level memory of τ is ultimately constant at j, and $(\mathtt{A}_{|\mathtt{H}^j}, \{\mathtt{H}^j\})$ was winning for Eve; ρ is ultimately a play of $\mathcal{A}_{|\mathcal{H}^j}$ consistent with τ^j, so Eve can win only by visiting all the states of $\mathcal{H}^j$. But $\mathcal{H}^j$ is not a trap for Adam, and the definition of τ implies that Adam leaves as soon as possible. So, at least one of the states of $\mathcal{H}^j$ was not visited, and ρ is winning for Adam.

This completes the proof of Lemma 4.4. The proof of Lemma 4.3 is more involved, due to the necessity to avoid at least one of the states of $\mathcal{F}_i$. By Proposition 4.1 there is a state q in $\mathtt{F}_i$ such that $X = \mathrm{Attr}_E(\{q\}, \mathtt{A}_{|\mathtt{F}_i})$ is not equal to $\mathtt{A}_{|\mathtt{F}_i}$. It follows from the definition of $\mathtt{A}_{|\mathtt{F}_i}$ that neither $\mathcal{F}_i \cap X$ nor $\mathcal{F}_i \setminus X$ is empty. Adam's strategy is then exactly the same than in the proof of Lemma 4.4, with the provision that Adam never moves from $\mathcal{F}_i \setminus X$ to X: this guarantees that the token cannot visit infinitely often all the states of $\mathcal{F}_i$, and completes the proof of Lemma 4.3. □

The correctness of Algorithm 4.1 follows from Lemmas 4.2, 4.3, and 4.4: the first one guarantees that the states in $\mathtt{W}_E \cap \mathcal{Q}$ are winning for Eve, and the last one that the states remaining at the end of Algorithm 4.1 are winning for Adam.

About complexity, there are at most $|\mathcal{F}|$ loops in a run, and the most time-consuming operation is to compute the winner of the games $(\mathtt{A}_{|\mathtt{F}_i}, \{\mathtt{F}_i\})$, which are quadratic in $|\mathtt{A}| \leq (|\mathcal{A}| + |\mathcal{F}|)$. Thus, the worst-case time complexity of Algorithm 4.1 is $O(|\mathcal{F}| \cdot (|\mathcal{A}| + |\mathcal{F}|)^2)$, which completes the proof of Theorem 4.5:

Theorem 4.5 *The winner problem of explicit $2\frac{1}{2}$-player Muller games belongs to* PTIME.

We can use Theorem 3.29 to directly derive a complexity class for the quantitative problems of explicit games:

Theorem 4.6 *The value problem of explicit $2\frac{1}{2}$-player Muller games belongs to* NP *and co-*NP.

Notice that these complexity results depend on the fact that there is no colouring function: even in the very restricted case of the *win-set representation* [McN93], where the only difference is the introduction of a set of irrelevant states, the winner problem becomes PSPACE-hard [HD05].

4.2 Solution through reductions

A first approach to the solution of $2\frac{1}{2}$-player Muller games uses successive reductions from the well-studied problem of 2-player *parity*[2] *games* (4.2.1). It is possible to reduce the qualitative solution of $2\frac{1}{2}$-player parity games to this problem (4.2.2), and Muller conditions to parity conditions (4.2.3).

4.2.1 Solving 2-player parity games

The complexity of 2-player parity games is one of the central questions in game theory. One of the motivation is the link between these games and logics: there is a polynomial reduction from μ-calculus to 2-player parity games, and vice versa. Another one is that parity games admit positional strategies (under some hypothesis, they are even the only games to admit pure and positional strategies [CN06]).

Theorem 4.7 ([EJ91], [Mos91]) *In a 2-player parity game, both players have positional winning strategies.*

An immediate consequence is that parity game can be solved with NP or co-NP algorithm, by guessing a strategy for one or the other player.

Theorem 4.8 *The problem of the winner in 2-player parity games belongs to* NP *and to co-*NP.

It is even possible to adapt the strategy improvement for $2\frac{1}{2}$-player reachability games, in a discrete fashion [VJ00]. The resulting algorithm is polynomial for 1-player games, and conjectured to be also polynomial for 2-player games. It is also possible to solve parity games with a recursive algorithm

[2] "Priority" would be a much better name.

like the one we describe in Section 4.3. Other approaches introduced *small progress measure* [Jur00] (later extended to Rabin and Streett games in [PP06]), or mixed these with a round of exhaustive exploration of small subarenas [JPZ06, Sch07].

4.2.2 $2\frac{1}{2}$-player parity games to 2-player parity games

It is possible to reduce qualitative problems for $2\frac{1}{2}$-player Muller games to the winner problem of 2-player games: see [JKH02] for Büchi and co-Büchi conditions, [CJH03] for parity conditions, and [CdAH05] for Rabin and Streett conditions. The principle is to replace the random states with a gadget where Adam and Eve "barter" for the right to choose the next state. For example, Figure 4.4 presents the reduction of [CJH03] for parity games.

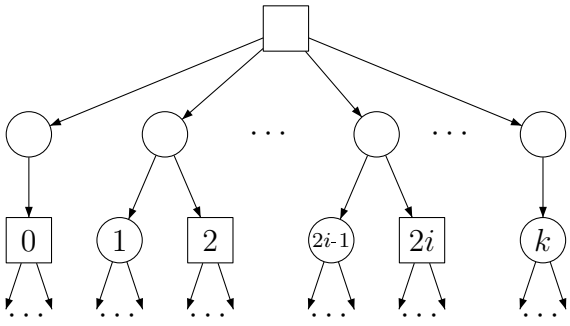


Figure 4.4: Parity gadget: from $2\frac{1}{2}$-player to 2-player

Each visit to a random state is replaced by this "gadget", where Adam chooses first a rank i, and Eve can:

- either visit to a $2i-1$ priority and decide the next state;
- or visit to a $2i$ priority and let Adam decide the next state.

This reduction is polynomial, and preserves the winning region (Eve's winning region corresponds to her almost-sure region, and Adam's region to his positive region). Furthermore, the positional strategies of the reduced game translate as positional strategies in the original game. Theorem 4.9 and 4.10 follow:

Theorem 4.9 *In a $2\frac{1}{2}$-player parity game, both player have positional strategies.*

Theorem 4.10 *The qualitative problems of $2\frac{1}{2}$-player parity games are in* NP $\cap$ *co-*NP

4.2.3 Muller conditions to parity conditions

Muller conditions can be translated as parity conditions by adding information to the states. The first data structure considered to this effect was the *Latest Appearance Records* (LAR) of McNaughton, which were used by Gurevich and Harrington as memory for winning strategies in Muller games [GH82]. Thomas use them in [Tho95] to reduce Muller games to parity games. However, the size of the LAR structure is totally insensitive to the actual winning condition. Zielonka's insightful construction [Zie98] represents a Muller condition by a *split tree* whose nodes are labelled by sets of colours, focusing on the alternation between the sets winning for Eve and those winning for Adam:

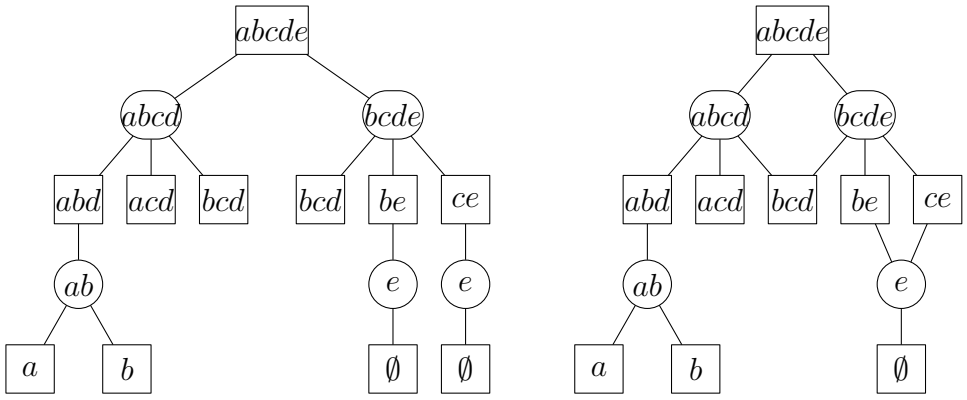
Definition 4.11 (Zielonka tree [Zie98]) *The Zielonka tree of a Muller condition $\mathcal{F}$ over $\mathcal{C}$, denoted $\mathcal{Z}_{\mathcal{F},\mathcal{C}}$, is the rooted tree with the following properties:*

- *each node is labelled by a set of colours, and two different siblings have different labels; if the label of a node is winning for Eve, the node* belongs to Eve*, otherwise it* belongs to Adam*;*
- *the root of $\mathcal{Z}_{\mathcal{F},\mathcal{C}}$ is labelled by $\mathcal{C}$;*
- *if n is a node labelled by $C \subseteq \mathcal{C}$, and $C_1, \ldots, C_h$ are the maximal subsets of C such that $C \in \mathcal{F} \nLeftrightarrow C_i \in \mathcal{F}$, then the children of n are labelled by the C_i's.*

Hunter and Dawar derive a DAG from this tree (the *Zielonka DAG*), by identifying the nodes with the same labels [HD05].

The Zielonka tree and DAG of $\mathfrak{S}$, from Figure 4.1, are represented in Figure 4.5(a).

Dziembowski, Jurdziński, and Walukiewicz presented in [DJW97] a reduction from Muller games to parity games using the branches of the Zielonka tree as data structure. Their construction builds a parity game $\mathbb{G}$ from a



(a) $\mathcal{Z}_{\mathfrak{F},\mathfrak{C}}$ (b) $\mathcal{D}_{\mathfrak{F},\mathfrak{C}}$

Figure 4.5: Zielonka representations of $\mathfrak{F}$

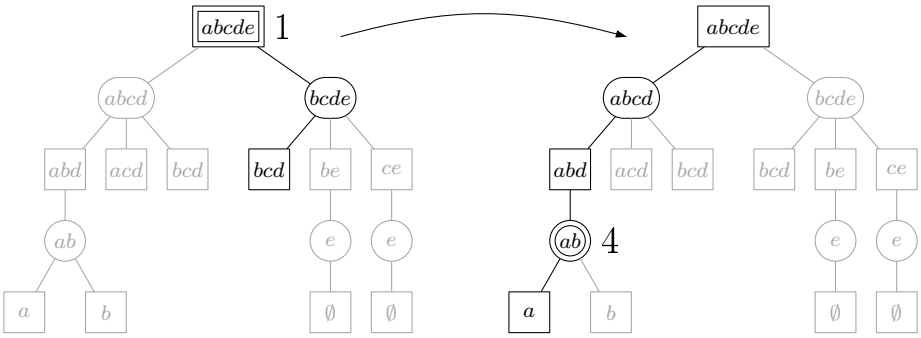
Muller game $\mathcal{G} = (\mathcal{A}, \mathcal{F})$. The states of $\mathbb{G}$ are pairs (q, b), with q a state of $\mathcal{A}$, and b a branch of $\mathcal{Z}_{\mathcal{F},\mathcal{C}}$. The support of q and b is the lowest node where $\chi(q)$ appears. The colour of a state (q, b) is the depth of the support of q and b in $\mathcal{Z}_{\mathcal{F},\mathcal{C}}$. There is a transition from (q, b) to (q', b') if and only if there is a transition from q to q' in $\mathcal{G}$, and b' goes through the next child of the support of q and b. A transition, taken from the translation of $\mathfrak{G}$, is represented in Figure 4.6.

A play ρ is winning in $\mathbb{G}$ if and only if its projection on $\mathcal{Q}$ is winning in $\mathcal{G}$. The size of $\mathbb{G}$ is polynomial in the size of the game if the condition is represented by its Zielonka tree, so the complexity of these games is in NP $\cap$ co-NP :

Theorem 4.12 *The qualitative problems of $2\frac{1}{2}$-player Muller games whose winning condition are represented by their Zielonka tree belong to* NP *and co-*NP*.*

Furthermore, if we define $\ell_{\mathcal{F}}$ as the number of branches of the Zielonka tree:

Definition 4.13 (Number ℓ of a Muller condition) *Let $\mathcal{F}$ be a Muller condition on $\mathcal{C}$, and $\mathcal{C}_1 \cdots \mathcal{C}_k$ be the maximal subsets of $\mathcal{C}$ such that $\mathcal{C} \in \mathcal{F} \Leftrightarrow C_i \in \mathcal{F}$. We denote by $\mathcal{F}_i$ the Muller condition $\mathcal{F}_{|\mathcal{C}_i}$, and we define the number*



(a) $\chi(q) = a$ (b) $\chi(q') = b$

Figure 4.6: Coloration and transitions of the generalized LAR reduction

$\ell_{\mathcal{F}}$ *inductively as follows:*

$$\ell_{\mathcal{F}} = \begin{cases} 1 & \text{if } \mathcal{Z}_{\mathcal{F},\mathcal{C}} \text{ does not have any subtrees,} \\ \sum_{i=1}^{k} l_{\mathcal{F}_i} & \text{if } \mathcal{C} \in \mathcal{F} \text{ otherwise.} \end{cases}$$

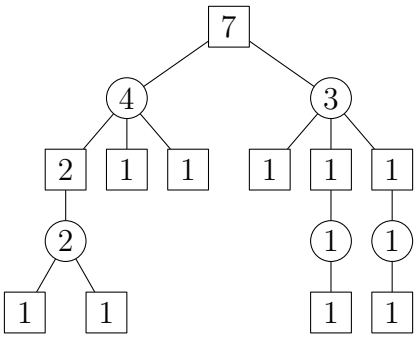


Figure 4.7: Computation of $\ell_{\mathfrak{S}}$

Both players have winning strategies with memory $\ell_{\mathcal{F}}$: by keeping the current branch of in memory, a player can determine where the token would be in $\mathbb{G}$, and play accordingly.

Theorem 4.14 *Let $\mathcal{F}$ be a Muller condition over $\mathcal{C}$, $\mathcal{A}$ be an arena on $\mathcal{C}$,*

and q be a state of $\mathcal{Q}$. If either player has a winning strategy from q, they have a pure winning strategy with memory $\ell_{\mathcal{F}}$.

4.3 Recursive algorithm

However, the Zielonka tree is not more succinct than any of the representations we presented, and the reduction uses thus an exponential space in these cases. However, it is possible to simulate this reduction on the fly, in space polynomial in the size of the arena. This approach is recursive, and has been implemented first by Dziembowski, Jurdziński, and Walukiewicz for 2-player Muller games (and infinite arenas) in [DJW97]. The special case of 2-player Streett games was studied in [Hor05], and extended to $2\frac{1}{2}$-player games in [Hor07a]. Chatterjee later extended the extension to $2\frac{1}{2}$-player Muller games in [Cha07c].

The recursive calls of this recursive algorithm corresponds to the structure of the Zielonka tree: the final solution is the one we get at the root, and solving a node require to solve its children. However, we do not need to remember the structure of the tree, and computing the children of a node in the Zielonka tree can be done in polynomial space regardless of the representation of the Muller condition, as long as deciding the winner of a set of colours can be done in `PSPACE`.

As Muller conditions are prefix-independent, we use the results of Chapter 3, to describe only a partial algorithm. The role of the players in this algorithm depends on who wins if all the colours are visited: if it is Eve, we compute a subset of Adam's positive region; if it is Adam, we compute a subset of Eve's winning region. In order to get the actual regions, we use the fix-point algorithm, and the switching algorithm if needed.

The study of the Zielonka tree $\mathcal{Z}_{\mathcal{F},\mathcal{C}}$ of a Muller condition $\mathcal{F}$ on $\mathcal{C}$ enables us to define two asymmetric numbers $m_{\mathcal{F}}$ and $r_{\mathcal{F}}$, which are tight bounds for the memory needed in $\mathcal{F}$-games (Theorems 4.16 and 4.18).

Definition 4.15 (Number m of a Muller condition) *Let $\mathcal{F}$ be a Muller condition on $\mathcal{C}$, and $\mathcal{C}_1 \dots \mathcal{C}_k$ be the maximal subsets of $\mathcal{C}$ such that $\mathcal{C} \in \mathcal{F} \Leftrightarrow \mathcal{C}_i \in \mathcal{F}$. We denote by $\mathcal{F}_i$ the Muller condition $\mathcal{F}_{|\mathcal{C}_i}$, and we define the number $m_{\mathcal{F}}$ inductively as follows:*

$$m_{\mathcal{F}} = \begin{cases} 1 & \text{if } \mathcal{Z}_{\mathcal{F},\mathcal{C}} \text{ does not have any subtrees,} \\ \max\{1, m_{\mathcal{F}_1}, m_{\mathcal{F}_2}, \ldots, m_{\mathcal{F}_k}\} & \text{if } \mathcal{C} \notin \mathcal{F} \text{ (Adam node),} \\ \sum_{i=1}^{k} m_{\mathcal{F}_i} & \text{if } \mathcal{C} \in \mathcal{F} \text{ (Eve node).} \end{cases}$$

Theorem 4.16 ([DJW97, Cha07c]) *Let $\mathcal{F}$ be a Muller condition over $\mathcal{C}$. In any $\mathcal{A}$ on $\mathcal{C}$, if Eve has a winning strategy in the game $(\mathcal{A}, \mathcal{F})$, she has a pure winning strategy with memory $m_{\mathcal{F}}$. Furthermore, there is a 2-player arena $\mathcal{A}_{\mathcal{F}}$ where Eve has a winning strategy in the game $(\mathcal{A}_{\mathcal{F}}, \mathcal{F})$, and none of her pure strategies with memory less than $m_{\mathcal{F}}$ is winning.*

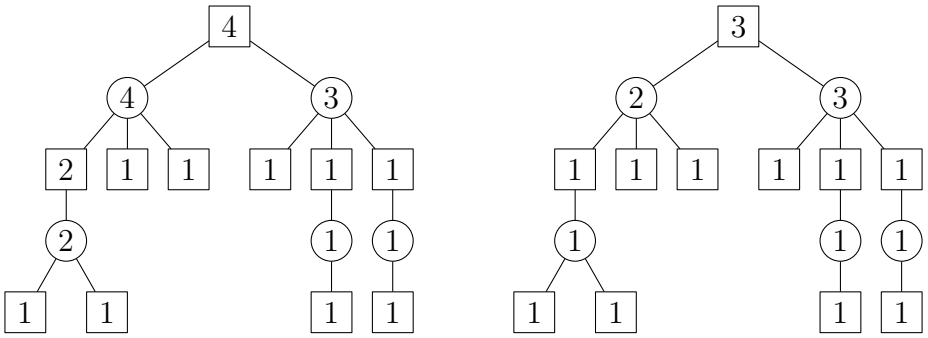
Definition 4.17 (Number r of a Muller condition) *Let $\mathcal{F}$ be a Muller condition on $\mathcal{C}$ such that the root of the Zielonka tree $\mathcal{Z}_{\mathcal{F},\mathcal{C}}$ has ℓ leaf and k non-leaf children. We denote by $\mathcal{C}_1 \ldots \mathcal{C}_k$ the labels of the non-leave children, and by $\mathcal{F}_i$ the Muller condition $\mathcal{F}_{|\mathcal{C}_i}$. The number $r_{\mathcal{F}}$ is defined inductively as follows:*

$$r_{\mathcal{F}} = \begin{cases} 1 & \text{if } \mathcal{Z}_{\mathcal{F},\mathcal{C}} \text{ does not have any subtrees,} \\ \max\{1, r_{\mathcal{F}_1}, r_{\mathcal{F}_2}, \ldots, r_{\mathcal{F}_k}\} & \text{if } \mathcal{C} \notin \mathcal{F} \text{ (Adam node),} \\ \sum_{i=1}^{k} r_{\mathcal{F}_i} & \text{if } \mathcal{C} \in \mathcal{F} \text{ (Eve node) and } \ell = 0, \\ \sum_{i=1}^{k} r_{\mathcal{F}_i} + 1 & \text{if } \mathcal{C} \in \mathcal{F} \text{ (Eve node) and } \ell \geq 1. \end{cases}$$

Theorem 4.18 ([Hor09]) *Let $\mathcal{F}$ be a Muller condition over $\mathcal{C}$. In any $\mathcal{A}$ on $\mathcal{C}$, if Eve has a winning strategy in the game $(\mathcal{A}, \mathcal{F})$, she has a randomised winning strategy with memory $r_{\mathcal{F}}$. Furthermore, there is a 2-player arena $\mathcal{A}_{\mathcal{F}}$ where Eve has a winning strategy in the game $(\mathcal{A}_{\mathcal{F}}, \mathcal{F})$, and none of her randomised strategies with memory less than $r_{\mathcal{F}}$ is winning.*

In the remainder of this chapter, we prove simultaneously Theorems 4.16 and 4.18. The study of the recursive algorithm of this section provide the upper bounds, while the lower bounds are proved in Section 4.4.

All the descriptions and properties of this section refer to a generic game $\mathcal{G} = (\mathcal{A}, \mathcal{F})$ over the set of colours $\mathcal{C}$. In order to simplify sentences, we suppose that the root of $\mathcal{Z}_{\mathcal{F},\mathcal{C}}$ belongs to Eve. The case where Adam owns the root works exactly in the same way, exchanging the roles of Eve and Adam.



(a) $m_{\mathfrak{F}}$ (b) $r_{\mathfrak{F}}$

Figure 4.8: Computation of $m_{\mathfrak{F}}$ and $r_{\mathfrak{F}}$

4.3.1 Partial algorithm

The partial algorithm itself is recursive: it involves the solution of sub-games, in the sense that the arena is a sub-arena of $\mathcal{A}$, and the winning condition corresponds to a subtree of $\mathcal{Z}_{\mathcal{F},\mathcal{C}}$. Let us start with some notations: we denote by $\mathcal{C}_1 \dots \mathcal{C}_\ell$ the labels of the children of the root. For each of them, $\mathcal{F}_i$ is the restriction of $\mathcal{F}$ to $\mathcal{C}_i$, and D_i is the set $\mathcal{C} \setminus \mathcal{C}_i$. These notations are summarised on Figure 4.9, which represents the top of $\mathcal{Z}_{\mathcal{F},\mathcal{C}}$.

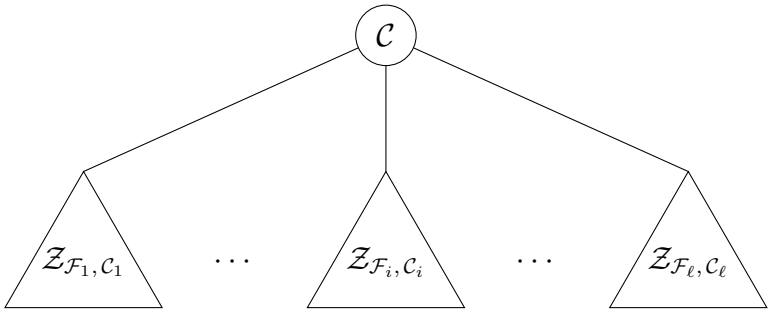


Figure 4.9: Generic top of a Zielonka Tree

Intuitively, in order to win, Adam must eventually stay clear of at least one of the D_i's, while winning with respect to the sub-condition $\mathcal{F}_i$. Otherwise, Eve can win either in one of the sub-conditions, or by visiting cyclically

each of the D_i's. For each i, the algorithm computes $\mathcal{A}_i = \mathcal{A} \setminus \mathrm{Attr}_E(D_i, ()\mathcal{A})$ and the Adam's almost-sure region in $(\mathcal{A}_i, \mathcal{F}_i)$. If one of these regions is not empty, it is returned. Otherwise, the algorithm returns $\emptyset$. This algorithm is described as Algorithm 4.2.

```
   Input: A Muller game G = (A, F) such that C ∈ F
   Output: A (non-empty) subset of Win_A^{F,>0}(A)
1  forall i ∈ {1, .., ℓ} do
2      A_i ← A \ Attr_E(D_i, A);
3      if Win_A^{F_i,1}(A_i) ≠ ∅ then
4          return Win_A^{F_i,1}(A_i);
5      end
6  end
7  return ∅
```

Algorithm 4.2: Partial algorithm for Muller games

Notice that in line 3, the recursive call computes the almost-sure winning region of Adam in $(\mathcal{A}_i, \mathcal{F}_i)$. Actually, we just need a partial algorithm for Adam with respect to $\mathcal{F}_i$, but, as Adam wins the root of $\mathcal{Z}_{\mathcal{F}_i, \mathcal{C}_i}$, we cannot use directly Algorithm 4.2. So, we use his almost-sure winning region, in the spirit of the switching algorithm (Algorithm 3.2). In terms of complexity, it means that we use a fix-point computation in each recursive call.

4.3.2 Non-empty output: spatial composition

When the output X of Algorithm 4.2 is non-empty, we have to show that it belongs to the positive region of Adam. Let i be the last value of $\mathtt{i}$ in the run. X is thus the almost-sure region of Adam in $(\mathcal{A}_i, \mathcal{F}_i)$. We claim that X belongs to Adam's almost-sure region in $\mathcal{G}$. Let τ_i be an almost-sure strategy for Adam from X in $(\mathcal{A}_i, \mathcal{F}_i)$, and consider what happens if a play of $\mathcal{G}$ starts in X and Adam plays τ_i:

- X is a trap for Eve in $\mathcal{A}_i$, which is itself a trap for her in $\mathcal{A}$, so the token remains surely in X;
- τ_i is almost-sure for Adam in $(\mathcal{A}_i, \mathcal{F}_i)$, so the set of colours visited infinitely often almost surely does not belong to $\mathcal{F}_i$.

As $\mathcal{F}_i$ is the restriction of $\mathcal{F}$ to $\mathcal{C}_i$, and $\chi(\mathcal{A}_i) \subseteq \mathcal{C}_i$, it follows that the set of colours visited infinitely often almost-surely does not belong to $\mathcal{F}$. So τ_i is almost-sure for Adam from X in $\mathcal{G}$, X belongs to his almost-sure region in $\mathcal{G}$, and consequently, to his positive region. Notice that, although X is almost-sure for Adam in $\mathcal{G}$, the region returned by the fix-point is only positive, since the iteration involves a positive attractor.

Furthermore, as Adam has pure (randomised) strategies with memory at most $m_{\overline{\mathcal{F}_i}}$ ($r_{\overline{\mathcal{F}_i}}$) for each condition $\mathcal{F}_i$, he has pure (randomised) strategies with memory $m_{\overline{\mathcal{F}}} = \max_{1 \leq i \leq \ell} m_{\overline{\mathcal{F}_i}}$ ($r_{\overline{\mathcal{F}}} = \max_{1 \leq i \leq \ell} r_{\overline{\mathcal{F}_i}}$) for $\mathcal{F}$.

4.3.3 Empty output: temporal composition

When the output of Algorithm 4.2 is empty, we need to show that Eve wins almost-surely everywhere in $\mathcal{G}$. The first remark is that Eve wins positively everywhere in each game $(\mathcal{A}_i, \mathcal{F}_i)$. By the positive-almost property, she also wins almost-surely everywhere in each game $(\mathcal{A}_i, \mathcal{F}_i)$. For each $1 \leq i \leq \ell$, let σ_i be an almost-sure strategy for Eve in $(\mathcal{A}_i, \mathcal{F}_i)$. The strategy σ uses a top-level memory with values in $1 \dots \ell$ to switch between these strategies. If the top level memory is equal to i, Eve plays as follows:

- in $\mathcal{A}_i$, play according to σ_i;
- in $\mathrm{Attr}_E(D_i, \mathcal{A}) \setminus D_i$, play the attractor strategy to D_i;
- in D_i, move to any state of $\mathcal{A}$ and update the top-level memory to $i \mod \ell + 1$.

Let ρ be a play consistent with σ. It falls in exactly one of these three cases:

1. the top-level memory is not ultimately constant;
2. the top level memory is ultimately constant at i, and $\mathrm{Inf}(\rho) \subseteq \mathcal{A}_i$;
3. the top level memory is ultimately constant at i, and $\mathrm{Inf}(\rho) \not\subseteq \mathcal{A}_i$.

In the first case, each of the D_i is visited infinitely often, so Eve wins surely; in the second case, ρ is ultimately a play of $(\mathcal{A}_i, \mathcal{F}_i)$ consistent with σ_i, so Eve wins almost surely; finally, by Proposition 1.5, the last case almost never occurs. Thus, σ is almost surely winning. Furthermore, if Eve has

pure strategies with memory at most $m_{\mathcal{F}_i}$ for each condition $\mathcal{F}_i$, she has pure strategies with memory $m_{\mathcal{F}} = \sum_{1 \leq i \leq \ell} m_{\mathcal{F}_i}$ for $\mathcal{F}$.

If some of the children are leaves, we can define a randomised strategy with less memory. Indeed, if $\mathcal{Z}_{\mathcal{F}_i,\mathcal{C}_i}$ is reduced to a leaf belonging to Adam, Eve cannot win in $(\mathcal{A}_i, \mathcal{F}_i)$, so her attractor to D_i covers $\mathcal{A}$ entirely. Instead of defining a different attractor strategy to each of these sets (which is necessary in pure strategies), we can define a generic strategy σ_0, which consists in always choosing the next state at random. As each attractor covers the whole arena, there is always the off-chance that the choices of Adam forces the token to the desired set. This is the core idea of [CdAH04], which shows that Eve has positional strategies for upward-closed Muller conditions. If *all* the children are leaves, σ_0 can directly replace σ, as Chatterjee showed in [Cha07b]. When some children are leaves, and others are not, the problem is to decide when the memory should be updated, as the strategy must guarantee that all the corresponding D_i's are visited infinitely often. Our solution is to randomise this update: each time Adam's top-level memory is 0, it has equal chances to remain 0 and to be updated to 1. The probability that the token visits a D_i is still positive: Adam just needs to do the correct moves *and* to remain in the correct memory state long enough. If the top-level memory is 0 infinitely often, the probability that the all the D_i's corresponding to a leaf are visited infinitely often is one. So, if Eve has randomised strategies with memory at most $r_{\mathcal{F}_i}$ for each condition $\mathcal{F}_i$, she has randomised strategies with memory $r_{\mathcal{F}} = \sum\{r_{\mathcal{F}_i} \mid \mathcal{Z}_{\mathcal{F}_i,\mathcal{C}_i} \text{ is not a leaf}\} + 1$ if at least one of the $\mathcal{Z}_{\mathcal{F}_i,\mathcal{C}_i}$ is a leaf.

4.4 Lower bounds for Muller conditions

In this section, we prove the lower bounds in Theorems 4.16 and 4.18. We first define a class of sub-DAGs, the *cropped DAGs* of the Zielonka DAG, which have a strong relation with the numbers $m_{\mathcal{F}}$ and $r_{\mathcal{F}}$ (4.4.1), and then derive from them 2-player arenas which follow roughly their structure (4.4.2). We show that these arenas are winning for Eve, and define "branch strategies" for Adam (4.4.3). Any pure strategy with less than $m_{\mathcal{F}}$ states, and any random strategy with less than $r_{\mathcal{F}}$ memory states fails against at least one of the branch strategies of Adam (4.4.4). Finally, we show that for many Muller conditions, the bounds $m_{\mathcal{F}}$ and $r_{\mathcal{F}}$ still hold when the arena is polynomial in the number of colours(4.4.5).

4.4.1 Cropped DAGs

The relation between the numbers $m_{\mathcal{F}}$ and $r_{\mathcal{F}}$ and the shape of $\mathcal{D}_{\mathcal{F},\mathcal{C}}$ is asymmetrical: they depend directly on the number of children of Eve's nodes, and not at all on the number of children of Adam's nodes. The notion of *cropped DAG* is the next logical step: a sub-DAG where Eve's nodes keep all their children, while each node of Adam keeps only one child. Definition 4.19 formalises this idea:

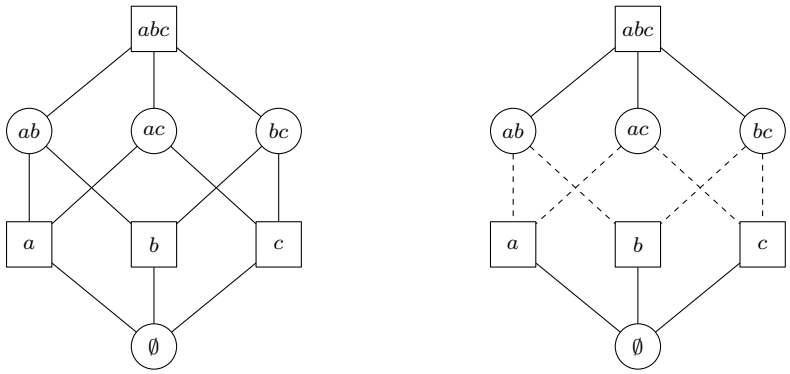
Definition 4.19 *A DAG $\mathcal{E}$ is a cropped DAG of a Zielonka DAG $\mathcal{D}_{\mathcal{F},\mathcal{C}}$ if and only if*

- *The nodes of $\mathcal{E}$ are a subset of the nodes of $\mathcal{D}_{\mathcal{F},\mathcal{C}}$. Furthermore, the owner and label of a node in $\mathcal{E}$ are its owner and label in $\mathcal{D}_{\mathcal{F},\mathcal{C}}$.*
- *There is only one node without predecessor in $\mathcal{E}$, which we call the root of $\mathcal{E}$. It is the root of $\mathcal{D}_{\mathcal{F},\mathcal{C}}$, if it belongs to Eve; otherwise, it is one of its children.*
- *The children of a node of Eve in $\mathcal{E}$ are exactly its children in $\mathcal{D}_{\mathcal{F},\mathcal{C}}$.*
- *A node of Adam has exactly one child in $\mathcal{E}$, chosen among his children in $\mathcal{D}_{\mathcal{F},\mathcal{C}}$, providing there is one. If it has no children in $\mathcal{D}_{\mathcal{F},\mathcal{C}}$, it has no children in $\mathcal{E}$.*

A cropped DAG has the general form of a Zielonka DAG: the nodes belong to either Eve or Adam, they are labelled by sets of states, and the label of a child is always a strict subset of the label of his parents. However, a cropped DAG is not necessarily the Zielonka DAG of a(nother) Muller condition: in the case of 'cardinal parity with three colours" —Figure 4.10— a cropped DAG contains at least two nodes on the "singleton" level, and each node labelled with a doubleton has only one child, while in a Zielonka DAG, a singleton node must have two parents.

On the other hand, when it comes to computing $m_{\mathcal{F}}$ and $r_{\mathcal{F}}$, it is enough to know who owns a state to decide which case of Definitions 4.15 and 4.17 is relevant. It is thus possible to define the numbers $m_{\mathcal{E}}$ and $r_{\mathcal{E}}$ of a cropped DAG $\mathcal{E}$ in exactly the same way.

In fact, these numbers have a more intuitive meaning in the case of a cropped DAG $\mathcal{E}$: $m_{\mathcal{E}}$ (and $r_{\mathcal{E}}$ when the leaves belong to Eve) is the number



(a) Zielonka DAG of cardinal parity 3

(b) cropped DAG

Figure 4.10: Cropped DAGs are not Zielonka DAGs

of branches in $\mathcal{E}$. When Adam owns the leaves, $r_{\mathcal{E}}$ is the number of branches in $\mathcal{E}$ without the leaves.

There is also a direct link between the cropped DAGs of a Zielonka DAG $\mathcal{D}_{\mathcal{F},\mathcal{C}}$ and the numbers $m_{\mathcal{F}}$ and $r_{\mathcal{F}}$: in a cropped DAG, there is one child for each internal node of Adam; in the recursive definition of $m_{\mathcal{F}}$ and $r_{\mathcal{F}}$, there is a maximum over the values of the children. Proposition 4.20 follows directly:

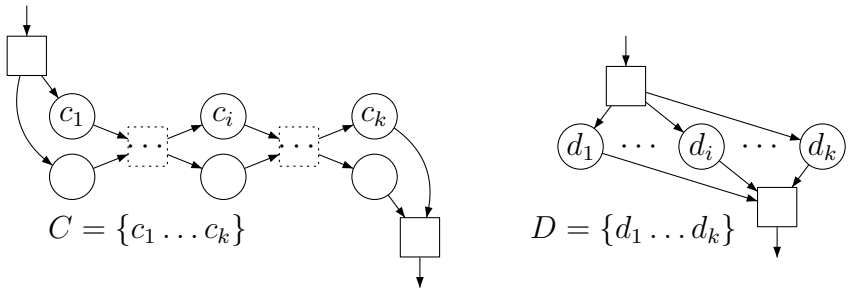
Proposition 4.20 *Let $\mathcal{F}$ be a Muller condition over the set of colours $\mathcal{C}$, and $\mathcal{D}_{\mathcal{F},\mathcal{C}}$ be its Zielonka DAG. Then for any cropped DAG $\mathcal{E}$ of $\mathcal{D}_{\mathcal{F},\mathcal{C}}$, we have $m_{\mathcal{E}} \leq m_{\mathcal{F}}$ and $r_{\mathcal{E}} \leq r_{\mathcal{F}}$. Furthermore, there two cropped DAGs $\mathcal{E}'$ and $\mathcal{E}^*$ such that $m_{\mathcal{E}'} = m_{\mathcal{F}}$ and $r_{\mathcal{E}^*} = r_{\mathcal{F}}$.*

4.4.2 From cropped DAGs to arenas

From any cropped DAG $\mathcal{E}$ of $\mathcal{D}_{\mathcal{F},\mathcal{C}}$, we define an arena $\mathcal{A}_{\mathcal{E}}$ which follows roughly the structure of $\mathcal{E}$: the token starts from the root, goes towards the leaves, and then restarts from the root. In her nodes, Eve can choose to which child she wants to go. Adam's choices, on the other hand, consists in either stopping the current traversal or allowing it to proceed.

We present first two "macros", depending on a subset of $\mathcal{C}$. They are represented in Figure 4.11, and are the only occasions where colours are visited in $\mathcal{A}_{\mathcal{E}}$: all the other states are colourless.

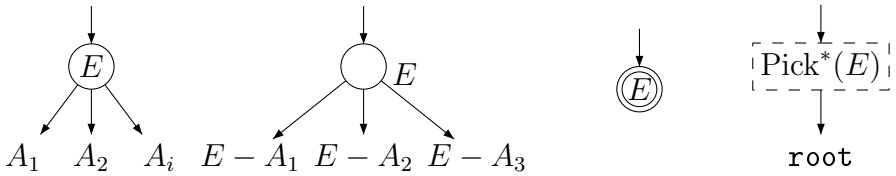
- In Pick$^*(C)$, Adam can visit any subset of colours in C;
- in Pick(D), he must visit exactly one colour in D.



(a) Pick$^*(C)$ (b) Pick(D)

Figure 4.11: Pick$^*(C)$ and Pick(D)

Eve's states in the arena $\mathcal{A}_\mathcal{E}$ are in bijection with the nodes of $\mathcal{E}$. Likewise, each outgoing transition corresponds to a child of the corresponding node. But the successors of these states are not themselves in bijection with the nodes of Adam: if a single node of Adam A is the child of two different nodes of Eve E and F, we must use the construction of Figure 4.13 twice: one for $E - A$ and one for $F - A$. In states corresponding to leaves, Eve has no decision to take; Adam can visit any colours in the label of the leaf (Pick* procedure). The token is then sent back to the root. These cases are described in Figure 4.12.



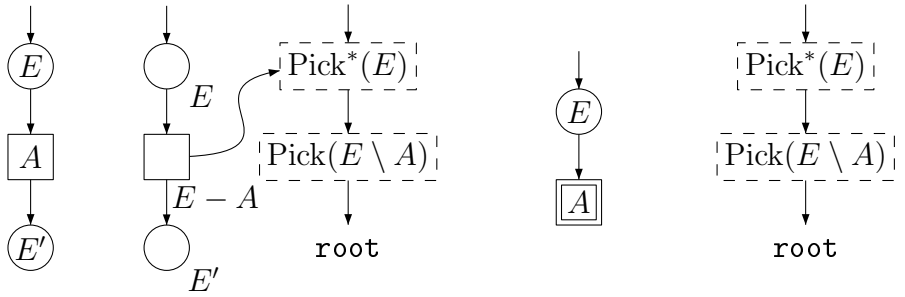
(a) Node "E" (b) Leaf "E"

Figure 4.12: Eve chooses where to go ...

Adam's options on a given node, on the other hand, do not involve the choice of a child: by Definition 4.19, Adam's nodes in $\mathcal{E}$ have but one child. Instead, he can either stop the current traversal, or, if the current node is not a leaf, allow it to proceed to its only child.

If he chooses to stop, Adam has to visit some coloured states before the token is sent back to the root. The available choices depend on the labels of both the current and the *former* nodes — which is why there are as many copies of Adam's nodes in $\mathcal{A}_\mathcal{E}$ as they have parents in $\mathcal{E}$. If the parent is labelled by E, and the current node by A, the token goes through $\mathrm{Pick}^*(E)$ and $\mathrm{Pick}(E \setminus A)$. Adam can thus choose any number of colours in E, as long as he chooses at least one outside of A.

Notice that if Adam does not stop the traversal, the token is sent to the *unique* state corresponding to the child of the current node. This is why the size of these arenas are roughly DAG-sized, instead of tree-sized.



(a) Edge "E" - "A" when "A" is a node (b) Edge "E" - "A" when "A" is a leaf

Figure 4.13: ... and Adam chooses when to stop.

4.4.3 Strategies in the DAG game

We first describe a winning strategy σ for Eve in the game $(\mathcal{A}_\mathcal{E}, \mathcal{F})$. Its memory states are the branches of $\mathcal{E}$, and do not change during a traversal. Her moves in the memory state $b = E_1 A_1 \dots E_\ell(A_\ell)$ follow the branch b: in the state E_i, Eve chooses the successor corresponding to the transition $E_i - A_i$. Notice that Adam cannot diverge from the branch, as his nodes have at most one child. When he chooses to stop the traversal, Eve updates

her memory. If he stops at the ith step, while Eve is in the memory state $b = E_1A_1 \dots E_\ell(A_\ell)$. There are two cases:

- if E_i has zero or one child in $\mathcal{E}$, the memory is unchanged;
- otherwise, the new memory branch has $E_1A_1 \dots E_iA$ as a prefix, where A is the next child of E_i, or the first one if A_i was the last.

Proposition 4.21 *The strategy σ is surely winning for Eve in the game $(\mathcal{A}_\mathcal{E}, \mathcal{F})$.*

Proof. Let ρ be a play consistent with σ. We denote by i the smallest integer such that Adam stops infinitely often a traversal at the ith step.

After a finite prefix, the first $2i-1$ nodes in the memory branch are constant, and we denote them by $E_1A_1E_2 \dots E_i$. From this point on, whatever Adam does, he can only choose colours in E_i. Furthermore, each time he chooses i, he must choose a state outside of the current A_i, which changes afterwards to the next, in a circular way.

So, in the end, $\mathrm{Inf}(\rho) \subseteq E_i$, and, for any child A of E_i in $\mathcal{E}$, $\mathrm{Inf}(\rho) \not\subseteq A$. Thus ρ is winning for Eve. Proposition 4.21 follows. □

Obviously, Adam has no winning strategy in $\mathcal{A}_\mathcal{E}$. However, we describe the class of *branch strategies* for him, whose point is to punish any attempt of Eve to win with less than $m_\mathcal{F}$ or $r_\mathcal{F}$ memory states. There is one such strategy τ_b for each branch b in $\mathcal{E}$ (whence the name), and the principle is that τ_b stops the traversal as soon as Eve deviates from b:

Definition 4.22 *The branch strategy τ_b for Adam in $\mathcal{A}_\mathcal{E}$, corresponding to the branch $b = E_1A_1E_2 \dots E_\ell(A_\ell)$ in $\mathcal{E}$, is a positional strategy whose moves are described below.*

- *In a state $E - A$ such that $\exists i, E = E_i \wedge A \neq A_i$: stop the traversal and visit the colours of A_i;*
- *in a state $E - A$ such that $\exists i, E = E_i \wedge A = A_i$: send the token to E_{i+1};*
- *in the state $E_\ell - A_\ell$: visit E_ℓ;*
- *in the leaf E_ℓ: visit E_ℓ.*

Notice that no move is given for a state $E - A$ such that $\forall i, E \neq E_i$. The reason is that these states are not reachable from the root when Adam plays τ_b, so, in the limit, what he does in these states doesn't matter. Notice also that when Adam chooses to stop a traversal in a state $E_i - A$, he *can* visit exactly the colours of A_i: as A and A_i are maximal subsets of E_i, there is at least one state in $A_i \setminus A$ that he can pick in the $\text{Pick}(E_i \setminus A)$ area.

We informally describe one last strategy for Adam: the *passive strategy*, in which he never stops a traversal before it reaches a leaf, and then plays at random in the Pick / Pick* part.

4.4.4 Winning against branch strategies

Let $\sigma = (M, \sigma^{\mathtt{n}}, \sigma^{\mathtt{u}})$ be a pure strategy for Eve. We define the *branch of a memory state* $m \in M$ as the unique branch that the token follows if it starts in the root while Eve is in the memory state m and Adam plays a passive strategy.

Proposition 4.23 *Let $\sigma = (M, \sigma^{\mathtt{n}}, \sigma^{\mathtt{u}})$ be a pure winning strategy for Eve in $(\mathcal{A}_{\mathcal{E}}, \mathcal{F})$. Then σ has memory at least $m_{\mathcal{E}}$.*

Proof. The principle of the proof is that Eve needs a different memory state to deal with each of the $m_{\mathcal{E}}$ branch strategies of Adam.

Let $b = E_1 A_1 \dots E_\ell(A_\ell)$ be a branch of $\mathcal{E}$ and τ_b be the corresponding branch strategy for Adam. We consider the unique play ρ consistent with σ and τ_b. By definition of τ_b, the set of colours visited in a traversal of ρ is one of the A_i's, or E_ℓ if and only if the branch of the current memory state is b.

Suppose now that there is no memory state whose branch is b. As $A_1 \supset A_2 \supset \cdots \supset A_{\ell-1}(\supset A_\ell)$, the set of colours visited infinitely often in the play is one of the A_i's, and Adam wins. This is in contradiction with the fact that σ is winning. It follows that for each branch b of $\mathcal{E}$, there must be a memory state in M whose branch is b. As there is only one branch per memory state, and there are $m_{\mathcal{E}}$ branches, it follows that there are at least $m_{\mathcal{E}}$ memory states in M. This concludes the proof of Proposition 4.23. □

By Proposition 4.20, there is a cropped DAG $\mathcal{E}$ of $\mathcal{D}_{\mathcal{F},\mathcal{C}}$ such that $m_{\mathcal{E}} = m_{\mathcal{F}}$. So, in general, Eve needs pure strategies with memory $m_{\mathcal{F}}$ in order to win games whose winning condition is $\mathcal{F}$. As we saw in Section 4.3 that she has such strategies, it completes the proof of Theorem 4.16.

The notion of "branch of a memory state" carries to the case of randomised strategies, but not its unicity: even if Eve starts in the same memory state and Adam plays with a passive strategy, the random decisions can lead to different branches. We consider thus the *set of branches of a memory state* m: they are the branches that have a positive probability to be traversed when Eve is in the memory state m and Adam plays with a passive strategy.

Proposition 4.24 *Let $\sigma = (M, \sigma^{\mathtt{n}}, \sigma^{\mathtt{u}})$ be an almost-sure winning strategy for Eve in $(\mathcal{A}_{\mathcal{E}}, \mathcal{F})$. Then σ has memory at least $r_{\mathcal{E}}$.*

Proof. Again, the idea is that the memory states are necessary to deal with the branch strategies. However, as we will see, a single memory state can sometimes deal with several branch strategies.

Let $b = E_1 A_1 \dots E_\ell(A_\ell)$ be a branch of $\mathcal{E}$ and τ_b be the corresponding branch strategy for Adam. Consider what happens if Eve plays σ and Adam plays τ_b. By definition of τ_b, the set of colours visited in a traversal of ρ is one of the A_i's, or E_ℓ if and only if Eve plays along b. So, as σ wins against τ_b, there is at least one memory state m such that b is a branch of m.

Contrary to what happens in the pure case, m can have other branches than b, as long as they lead to visits to A_ℓ, and not another A_i *i.e.* when the other branches are siblings or nephews to b. Consequently, a memory state m is suitable against τ_b if

- b is a branch of m, and
- $E_1 A_1 \dots E_\ell$ is a prefix of all the branches of m

It follows that a single memory state can be suitable against two strategies τ_b and $\tau_{b'}$ corresponding to the branches $b = E_1 A_1 \dots E_\ell A_\ell$ and $b' = E'_1 A'_1 \dots E'_{\ell'} A'_{\ell'}$ only if they are siblings:

- $\ell = \ell'$
- $\forall i < \ell, E_i = E'_i$

There are $r_{\mathcal{E}}$ equivalence classes for this relation in $\mathcal{E}$. Hence, there must be at least $r_{\mathcal{E}}$ memory states in M. Proposition 4.24 follows. □

By Proposition 4.20, there is a cropped DAG $\mathcal{E}$ of $\mathcal{D}_{\mathcal{F},\mathcal{C}}$ such that $r_{\mathcal{E}} = r_{\mathcal{F}}$. So, in general, Eve needs randomised strategies with memory $r_{\mathcal{F}}$ in order to

win games whose winning condition is $\mathcal{F}$. As we saw in Section 4.3 that he had such strategies, it completes the proof of Theorem 4.18.

In their original proof for pure strategies, the authors of [DJW97] use cropped trees, in lieu of our cropped DAG. Our result is thus a little better, since Zielonka DAGs are more compact than Zielonka trees. For example, in a "matching priority" winning condition of rank k, the size of the tree is $O(2^k)$, while the DAG is of size $O(k)$.

4.4.5 Arenas of polynomial size

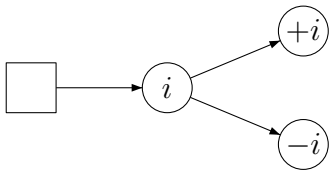
In general, the size of a cropped DAG is exponential in the number of colours. The question of whether the $m_{\mathcal{F}}$ and $r_{\mathcal{F}}$ bounds hold when the arenas are of polynomial size is still open. It does in several special cases: for example, the arenas for the "cardinal-guessing condition", used in [DJW97] and [Maj03] to prove global lower bounds for pure and random strategies are polynomial. It also holds for matching priority, Streett, matching conjunction, and cardinal parity. In each case, there is a witness arena of polynomial size where the plays consists in a succession of basic loops with parameters, which are chosen by Adam:

Matching priority: The parameter is a rank i. Eve must choose whether she visits $+i$ or $-i$ (Figure 4.14(a)). The size of the arena is linear in the maximal rank.

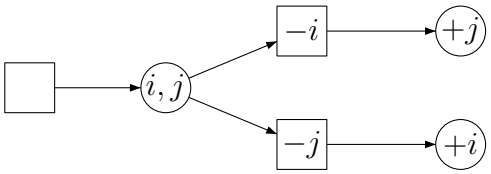
Streett: The parameters are two integers i and j. Eve visits either $-i$ and $+j$, or $+i$ and $-j$ (Figure 4.14(b)) a request for one of these and a response for the other. The size of the arena is quadratic in in the maximal rank.

Matching conjunction and cardinal parity: The parameters are two integers i and j. Eve can choose to visit either $+i$, $-i$, $+j$, or $-j$. Adam can then choose to visit either colour in the other pair (Figure 4.14(c)). The size of the arena is quadratic in in the maximal rank.

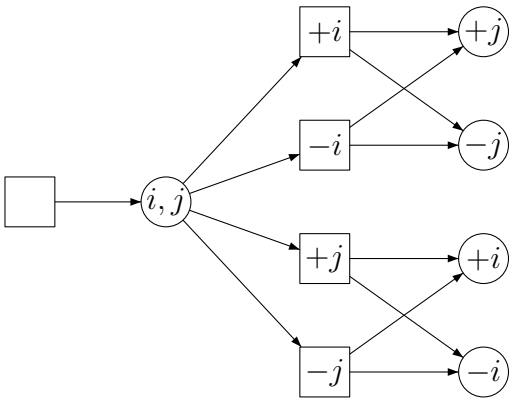
Finally, the only condition for which we did not found a polynomial arena was the majority condition — although Figure 4.14(a) shows that strategies whose memory is polynomial in the size of the arena are not enough. It is interesting to notice that it is the only condition we considered where the difference of cardinality between a node and one of its children is not bounded: in all the others, the change always depends on only one colour.



(a) Matching priority loop



(b) Streett loop



(c) Matching conjunction / cardinal parity loop

Figure 4.14: Polynomial-size arenas

4.5 Discussion

We found a polynomial algorithm for explicit Muller games, which provided us with a nice application for our results of Chapter 3. Using the standart equivalence, this algorithm can be used to decide the emptiness of explicit Muller tree automata. It would be interesting to know whether other problems on these automata can be solved in a similar fashion.

Our main result, the tight bound on the necessary memory for randomised strategies, raises four natural questions:

- Does these bounds still hold for arenas of polynomial size?
- Is it possible to find such bounds for any regular game, circumventing the product with an automata recognising the winning condition?

- Does our upper bound still hold for semi-randomised strategies?
- What are the links in terms of memory between our two models of randomised strategies with memory?

Chapter 5

Finitary winning in ω-regular games

"In the long run, we're all dead."

John Maynard Keynes

Every ω-regular specification (indeed, every specification) can be decomposed into a safety part and a liveness part [AS85]. The safety part ensures that the component will not do anything "bad" (such as violate an invariant) within any finite number of transitions. The liveness part ensures that the component will do something "good" (such as proceed, or respond, or terminate) within some finite number of transitions. Liveness can be violated only in the limit, by infinite sequences of transitions, as no bound is stipulated on when the "good" thing must happen. This infinitary, classical formulation of liveness has both strengths and weaknesses. A main strength is robustness, in particular, independence from the level of detail of the transitions. Another one is simplicity, allowing liveness to serve as an abstraction for complicated safety conditions. For example, a component may always respond in a number of transitions that depends, in some complicated manner, on the exact size of the stimulus. Yet, for correctness, we may be interested only that the component will respond "eventually". On the other hand, this also points to a weakness of the classical definition of liveness: it can be satisfied by components that in practice are quite unsatisfactory because no bound can be put on their response time. It is for this reason that alternative, stronger formulations of liveness have been proposed. One of these is *finitary* liveness

[AH98, DJP03], which requires the existence of a bound b such that every stimulus is followed by a response within b transitions. Notice that this is quite different from a specification which would insist on a response within a known bound b, as considered for example in [KPV07]. In the finitary case, the bound b may be arbitrarily large, but the response time must not grow forever from one stimulus to the next. In this way, finitary liveness still maintains the robustness (independence of step granularity) and simplicity (abstraction of complicated safety) of traditional liveness, while removing unsatisfactory implementations.

In this chapter, we study games with finitary winning conditions. The motivation is the same as for finitary liveness. Consider, for example, the synthesis of an elevator controller as a strategy in a game where one player represents the environment (*i.e.*, the pushing of call buttons on various floors, and the pushing of target buttons inside the elevators), and the other player represents the elevator control (*i.e.*, the commands to move an elevator up or down, and the opening and closing of elevator doors). Clearly, one objective of the controller is that whenever a call button is pushed on a floor, then an elevator will eventually arrive, and whenever a target button is pushed inside an elevator, then the elevator will eventually get to the corresponding floor. Note that this objective is formulated in an infinitary way (the key term is "eventually"). This is because, for robustness and simplicity, we do not wish to specify for each state the exact number of transitions until the objective must be met. However, a truly unbounded implementation of elevator control (where the response time grows from request to request, without bound) would be utterly unsatisfactory. A finitary interpretation of the objective prohibits such undesirable control strategies: there must exist a bound b such that the controller meets every call request, and every target request, within b transitions.

This chapter, whose results come from a joint work with Krishnendu Chatterjee and Thomas A. Henzinger [CHH09, CHH08], focuses on two types of objectives: the finitary parity condition, in Section 5.1; and the finitary Streett condition in Section 5.2.

5.1 Finitary Parity Games

We first consider the finitary version of the parity condition, which allow us to express finitary versions of the ω-regular conditions. It also subsumes finitary

reachability, finitary Büchi, and finitary co-Büchi objectives as special cases.

In the classical, infinitary parity objective, Eve wins by ensuring that every odd priority that repeats infinitely often is followed by a smaller even priority "eventually" (arbitrarily many transitions later). The *finitary* parity condition, by contrast, insists on the existence of a bound b such that every odd priority that repeats infinitely often is followed by a smaller even priority within b transitions. The finitary parity objective is strictly stronger than the classical parity objective, as is illustrated by the example of Figure 5.1.

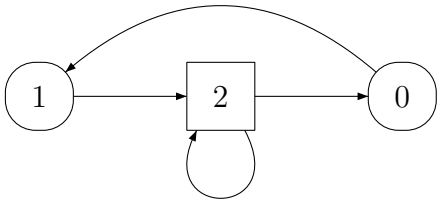


Figure 5.1: Finitary parity is not parity

In this parity arena, Eve wins with respect to the classical parity condition: the lowest colour of a play can be 0 if Adam chooses infinitely often to go right, or 2 if he eventually remains forever in the middle state, but it cannot by 1. However, Adam can win with respect to the finitary parity condition, by staying i times in the middle state the ith time he gets there from the left state: with this strategy, the distances grows without bound.

The finitary parity condition is formally defined through the notion of the *parity distance sequence* of an infinite play:

Definition 5.1 (Parity distance sequence of a play ρ) *Let $(\mathcal{A}, \chi)$ be a parity arena, and ρ be a play of $\mathcal{A}$. The parity distance sequence of ρ, denoted by $(\mathrm{Pdist}(\rho, i))_{i \in \mathbb{N}}$ is defined as follows: $\mathrm{Pdist}(\rho, i)$ is the smallest j such that $\chi(\rho_{i+j})$ is even and smaller or equal than $\chi(\rho_i)$. Notice that if $\chi(\rho_i)$ is even, $\mathrm{Pdist}(\rho, i)$ is equal to 0.*

Intuitively, the distance for a position i in a play with an odd priority at position i, denotes the shortest distance to a stronger even priority in the play. We assume the standard convention that the infimum of the empty set is ∞.

Definition 5.2 *Let $(\mathcal{A}, \chi)$ be a parity arena. A play ρ of $\mathcal{A}$ is winning for Eve in the finitary parity game $(\mathcal{A}, \chi)$ if and only if* $\limsup_i \mathrm{Pdist}(\rho, i) < \infty$.

By contrast, a play is winning for Eve in the infinitary parity game if there is only a finite number of positions with an infinite distance.

We present an algorithm computing the winning regions of a finitary parity game. Its correctness argument also proves directly the determinacy for these games, and establishes the existence of positional winning strategies for Eve; unsurprisingly, Adam needs infinite memory to win. This algorithm is polynomial time, and computes the winning region of a finitary parity games with n states and m transitions in time $O(n^2 \cdot m)$. This is in contrast to classical, infinitary parity games, for which the best known algorithms have time complexity $O(n^{\lfloor \frac{k}{3} \rfloor} \cdot m)$ [Sch07] or $n^{O(\sqrt{n})}$ [JPZ06].

We use two other notions of parity in our proofs: the well known *weak parity condition* (5.1.1) and a new *bounded parity condition* (5.1.2). Our algorithm for finitary parity games is obtained by iteratively solving bounded parity games, and the solution of bounded parity games is obtained by iteratively solving weak parity games (5.1.3).

5.1.1 Weak parity games

The notion of "weak" condition has been introduced by Staiger and Wagner in [SW74]. Weak conditions are ω-regular conditions that do not distinguish between plays with the same set of *occurring* colours:

Definition 5.3 *Let $\mathcal{A}$ be an arena on $\mathcal{C}$ and $\mathcal{F}$ be a subset of $\mathcal{P}(\mathcal{C})$. A play ρ of $\mathcal{A}$ is winning for Eve in the Staiger-Wagner game $(\mathcal{A}, \mathcal{F})$ if and only if* $\mathrm{Occ}(\rho) \in \mathcal{F}$.

This can be related with Muller conditions, which do not distinguish between plays with the same set of *infinitely occurring* colours. Staiger-Wagner games are thus often called *weak Muller* games. Likewise, we can define *weak Streett* games on Streett arenas, and *weak parity* games on parity arenas. In this section, we are mostly interested by this last case, where the winner is decided by the parity of the minimum priority occurring in the play:

Definition 5.4 *Let $(\mathcal{A}, \chi)$ be a parity arena. A play ρ of $\mathcal{A}$ is winning for Eve in the weak parity game $(\mathcal{A}, \chi)$ if and only if* $\min \chi(\mathrm{Occ}(\rho))$ *is even.*

We informally describe a recursive algorithm computing the winning regions of a 2-player weak parity game. The input is a 2-player arena $(\mathcal{A}, \chi)$, with $\mathcal{A} = (\mathcal{Q}, \mathcal{Q}_E, \mathcal{Q}_A, \mathcal{T})$ and $\chi : \mathcal{Q} \to [0 \dots k]$. The recursion step depends on the lowest colour i which appears in $\chi(\mathcal{A})$:

- If i is even, we start by computing the attractor of Eve to the states with priority i: these states clearly belong to the winning region of Eve. Furthermore, $\mathcal{A}_1 = \mathcal{A} \setminus \mathrm{Attr}_E(\chi^{-1}(i), \mathcal{A})$ is a trap for Eve, and thus a subarena. We can recursively compute $\mathrm{Win}_E^{\mathtt{wP}}(\mathcal{A}_1, \chi)$ and $\mathrm{Win}_A^{\mathtt{wP}}(\mathcal{A}_1, \chi)$. The winning regions of Eve and Adam in $\mathcal{G}$ are $\mathrm{Attr}_E(\chi^{-1}(i), \mathcal{A}) \cup \mathrm{Win}_E^{\mathtt{wP}}(\mathcal{A}_1, \chi)$ and $\mathrm{Win}_A^{\mathtt{wP}}(\mathcal{A}_1, \chi)$.
- If i is odd, we compute the attractor of Adam to the states with priority i: these states clearly belong to the winning region of Adam. Furthermore, $\mathcal{A}_1 = \mathcal{A} \setminus \mathrm{Attr}_A(\chi^{-1}(i), \mathcal{A})$ is a trap for Adam, and thus a subarena. We can recursively compute $\mathrm{Win}_E^{\mathtt{wP}}(\mathcal{A}_1, \chi)$ and $\mathrm{Win}_A^{\mathtt{wP}}(\mathcal{A}_1, \chi)$. The winning regions of Eve and Adam in $\mathcal{G}$ are $\mathrm{Win}_E^{\mathtt{wP}}(\mathcal{A}_1, \chi)$ and $\mathrm{Attr}_E(\chi^{-1}(i), \mathcal{A}) \cup \mathrm{Win}_A^{\mathtt{wP}}(\mathcal{A}_1, \chi)$.

The formal description of the complete algorithm can be found in [LT00]. At first sight, the time complexity appears to be $O(k \cdot |\mathcal{T}|)$. However, [Cha06] provides a detailed running time analysis and shows that, with adequate data structures, it runs in time $O(|\mathcal{T}|)$. Notice that as each attractor is defined on a different domain, they can be combined into positional winning strategies for both players. Theorem 5.5 summarises the results on games with weak parity objectives:

Theorem 5.5 (Weak parity games[LT00, Cha06]) *Let $(\mathcal{A}, \chi)$ be a 2-player parity arena. The following assertions hold:*

1. (Determinacy). *We have* $\mathrm{Win}_E^{\mathtt{wP}}(\mathcal{A}, \chi) = \mathcal{Q} \setminus \mathrm{Win}_A^{\mathtt{wP}}(\mathcal{A}, \chi)$.
2. (Strategy complexity). *Both players have positional winning strategies.*
3. (Time complexity). *The sets* $\mathrm{Win}_E^{\mathtt{wP}}(\mathcal{A}, \chi)$ *and* $\mathrm{Win}_A^{\mathtt{wP}}(\mathcal{A}, \chi)$ *can be computed in time* $O(|\mathcal{T}|)$.

5.1.2 Bounded parity games

We use the *bounded parity* condition as an intermediate step in our scheme to solve finitary parity games. This condition requires that whenever an odd priority is visited, then now or later a lower even priority is visited. The formal definition of the bounded parity condition uses the parity distance sequence: Eve must ensure that it takes only finite values. Adam wins if there is a position with an infinite distance:

Definition 5.6 *Let $(\mathcal{A}, \chi)$ be a parity arena. A play ρ of $\mathcal{A}$ is winning for Eve in the bounded parity game $(\mathcal{A}, \chi)$ if and only if $\forall i, \mathrm{Pdist}(\rho, i) < \infty$.*

We first show that we can use a fix-point algorithm to compute the winning regions of bounded parity games, using an algorithm for weak parity games as a partial algorithm (Definition 3.12, page 56):

Lemma 5.7 *Let $(\mathcal{A}, \chi)$ be a 2-player parity arena. The following assertions hold:*

1. *Adam's winning region for the weak parity condition is a subset of his winning region for the bounded parity condition:* $\mathrm{Win}_A^{\mathtt{wP}}(\mathcal{A}, \chi) \subseteq \mathrm{Win}_A^{\mathtt{bP}}(\mathcal{A}, \chi)$.

2. *If Eve wins from each state in $\mathcal{Q}$ for the weak parity condition, she wins from each state in $\mathcal{Q}$ for the bounded parity condition:* $\mathrm{Win}_E^{\mathtt{wP}}(\mathcal{A}, \chi) = \mathcal{Q} \Rightarrow \mathrm{Win}_E^{\mathtt{bP}}(\mathcal{A}, \chi) = \mathcal{Q}$.

Proof.

1. Consider a play ρ winning for Adam with respect to the weak parity condition, and denote by i the lowest colour occurring in ρ: $i = \min(\mathrm{Occ}(\rho))$. Let j be a position such that $\rho_j = i$. By Definition 5.1, $\mathrm{Pdist}(\rho, j) = \infty$. Thus ρ is winning for Adam with respect to the bounded parity condition.

2. By Theorem 5.5, Eve has a positional winning strategy σ with respect to the weak parity condition. Let ρ be a play consistent with σ. By contradiction, assume that there is a position i such that $\chi(\rho_i)$ is odd and $\forall i < j < i + |\mathcal{Q}|$, $\chi(\rho_j)$ is odd or greater than $\chi(\rho_i)$. There is a cycle c and a path w from ρ_i to c in $\mathcal{A}^\sigma$ such that all the colours appearing in c and w are greater than $\chi(\rho(i))$. The play $\rho_i w c^\omega$ is consistent with

σ, and winning for Adam with respect to the weak parity condition. This is in contradition with the fact that σ is winning for Eve with respect to this condition. Thus, for any play ρ consistent with σ, for any position k, we have $\mathrm{Pdist}(\rho, k) < |\mathcal{Q}|$, so ρ is winning for Eve with respect to the bounded parity condition. It follows that σ is winning for Eve with respect to the bounded parity condition.

□

The existence of positional winning strategies for Eve means that instead of asking for finite distance, we can ask that the distance is bounded by $|\mathcal{Q}|$:

Corollary 5.8 *For any 2-player parity arena $(\mathcal{A}, \chi)$, we have:*

$$\begin{aligned} \mathrm{Win}_E^{\mathtt{bP}}(\mathcal{A}, \chi) &= \{q \in \mathcal{Q} \mid \exists \sigma, \forall \tau, \forall i, \mathrm{Pdist}(\rho_q^{\sigma,\tau}, i) < \infty\} \\ &= \{q \in \mathcal{Q} \mid \exists \sigma, \forall \tau, \forall i, \mathrm{Pdist}(\rho_q^{\sigma,\tau}, i) < |\mathcal{Q}|\} \end{aligned}$$

We can thus use a fix-point algorithm to solve bounded games, as in Algorithm 5.1. By Lemma 3.13, the resulting complexity is $|\mathcal{Q}|$ times the complexity of the partial algorithm: $O(|\mathcal{Q}| \cdot |\mathcal{T}|)$.

Input: A parity arena $(\mathcal{A}, \chi)$
Output: The winning regions $\mathrm{Win}_E^{\mathtt{bP}}(\mathcal{A}, \chi)$ and $\mathrm{Win}_A^{\mathtt{bP}}(\mathcal{A}, \chi)$
1 $W_A = \emptyset$
2 $\mathcal{B} = \mathcal{A}$
3 **while** $\mathrm{Win}_A^{\mathtt{wP}}(\mathcal{B}, \chi) \neq \emptyset$ **do**
4 $\quad W_A \leftarrow W_A \cup \mathrm{Attr}_A(\mathrm{Win}_A^{\mathtt{wP}}(\mathcal{B}, \chi), \mathcal{B})$
5 $\quad \mathcal{B} \leftarrow \mathcal{B} \setminus \mathrm{Attr}_A(\mathrm{Win}_A^{\mathtt{wP}}(\mathcal{B}, \chi), \mathcal{B})$
6 **end**
7 **return** $(\mathcal{B}, W_A)$

Algorithm 5.1: Winning regions of a 2-player bounded parity game

Notice that the existence of positional winning strategies for Adam in weak parity games does not carry over to bounded parity games: Theorem 3.17 holds only for prefix-independent games. Indeed, there are arenas where Adam wins, but not with any positional winning strategy: consider, for example, the arena of Figure 5.1. On the other hand, the proof of Lemma 5.7 shows that Eve's positional winning strategies for the weak parity condition were still winning for the bounded parity condition.

Theorem 5.9 summarises our results on bounded parity games:

Theorem 5.9 (Bounded parity games) *Let $(\mathcal{A}, \chi)$ be a 2-player parity arena. The following assertions hold:*

1. (Determinacy). *We have* $\mathrm{Win}_E^{\mathtt{bP}}(\mathcal{A}, \chi) = \mathcal{Q} \setminus \mathrm{Win}_A^{\mathtt{bP}}(\mathcal{A}, \chi)$.
2. (Strategy complexity). *Eve has positional winning strategies, which bound the sequence distance to $|\mathcal{Q}|$.*
3. (Time complexity). *The sets* $\mathrm{Win}_E^{\mathtt{bP}}(\mathcal{A}, \chi)$ *and* $\mathrm{Win}_A^{\mathtt{bP}}(\mathcal{A}, \chi)$ *can be computed in time $O(|\mathcal{Q}| \cdot |\mathcal{T}|)$.*

5.1.3 Solving games with finitary parity objectives

The relations between the winning regions of bounded and finitary parity conditions are exactly the opposite of the relations between weak and bounded parity:

Lemma 5.10 *For any 2-player parity arena $(\mathcal{A}, \chi)$, the following assertions hold:*

1. *Eve's winning region for the bounded parity condition is a subset of her winning region for the finitary parity condition:* $\mathrm{Win}_E^{\mathtt{bP}}(\mathcal{A}, \chi) \subseteq \mathrm{Win}_E^{\mathtt{fP}}(\mathcal{A}, \chi)$.
2. *If Adam wins from all the state in $\mathcal{Q}$ for the bounded parity condition, then he wins from all the states in $\mathcal{Q}$ for the finitary parity condition:* $\mathrm{Win}_A^{\mathtt{bP}}(\mathcal{A}, \chi) = \mathcal{Q} \Longrightarrow \mathrm{Win}_A^{\mathtt{fP}}(\mathcal{A}, \chi) = \mathcal{Q}$.

Proof.

1. This is a direct consequence of Corollary 5.8.
2. Let τ be a winning strategy for Adam with respect to the bounded parity condition. We define the strategy τ' as follows:

   Step 1: Set a counter c to 1 and τ to its initial memory.

   Step 2: Play the strategy τ until the parity distance is equal to c.

   Step 3: Increment c.

   Step 4: Reset the memory for τ and go to to step 2.

Let ρ be a play consistent with τ'. We denote by w_c the factor corresponding to the cth iteration of τ'. Notice that if w_c is infinite, the $\{w_d \mid d > c\}$ are not defined. However, w_c is consistent with τ, and would be winning for Eve with respect to the bounded parity condition if it was infinite. Thus, each w_c is finite, and ρ is the concatenation of the $\{w_c \mid c \geq 1\}$. It follows that ρ and τ' are winning for Adam with respect to the finitary parity condition.

□

Algorithm 5.2 uses Algorithm 5.1 as a partial algorithm *for Eve*, for a resulting complexity is $|\mathcal{Q}|$ times the complexity of the partial algorithm: $O(|\mathcal{Q}|^2 \cdot |\mathcal{T}|)$.

Input: A parity arena $(\mathcal{A}, \chi)$
Output: The winning regions $\mathrm{Win}_E^{\mathtt{fP}}(\mathcal{A}, \chi)$ and $\mathrm{Win}_A^{\mathtt{fP}}(\mathcal{A}, \chi)$
1 $W_E = \emptyset$
2 $\mathcal{B} = \mathcal{A}$
3 **while** $\mathrm{Win}_E^{\mathtt{bP}}(\mathcal{B}, \chi) \neq \emptyset$ **do**
4 $\quad W_E \leftarrow W_E \cup \mathrm{Attr}_E(\mathrm{Win}_E^{\mathtt{bP}}(\mathcal{B}, \chi), \mathcal{B})$
5 $\quad \mathcal{B} \leftarrow \mathcal{B} \setminus \mathrm{Attr}_E(\mathrm{Win}_E^{\mathtt{bP}}(\mathcal{B}, \chi), \mathcal{B})$
6 **end**
7 **return** $(W_E, \mathcal{B})$

Algorithm 5.2: Winning regions of a 2-player finitary parity game.

Theorem 5.9 summarises our results on finitary parity games:

Theorem 5.11 (Finitary parity games) *For any 2-player parity arena $(\mathcal{A}, \chi)$, the following assertions hold:*

1. (Determinacy). *We have* $\mathrm{Win}_E^{\mathtt{fP}}(\mathcal{A}, \chi) = \mathcal{Q} \setminus \mathrm{Win}_A^{\mathtt{fP}}(\mathcal{A}, \chi)$.
2. (Strategy complexity). *Eve has memoryless winning strategies. In general, Adam has no strategy with finite memory.*
3. (Time complexity). *The sets* $\mathrm{Win}_E^{\mathtt{fP}}(\mathcal{A}, \chi)$ *and* $\mathrm{Win}_A^{\mathtt{fP}}(\mathcal{A}, \chi)$ *can be computed in time* $O(|\mathcal{Q}|^2 \cdot |\mathcal{T}|)$.

An interesting point is that the algorithm for 2-player bounded parity games is also a partial algorithm for $2\frac{1}{2}$-player finitary games. It can be used

to compute the sure region of Eve, interpreting the random states as states of Adam:

Lemma 5.12 *For any $2\frac{1}{2}$-player parity arena $(\mathcal{A}, \chi)$, the following assertions hold:*

1. *Eve's sure region for the bounded parity condition is a subset of her positive winning region for the finitary parity condition:* $\mathrm{Win}_E^{\mathtt{bP},\forall}(\mathcal{A}, \chi) \subseteq \mathrm{Win}_E^{\mathtt{fP},>0}(\mathcal{A}, \chi)$.

2. *If all the state in $\mathcal{Q}$ belongs to Adam's heroic region for the bounded parity condition, then he almost surely wins from all the states in $\mathcal{Q}$ for the finitary parity condition:* $\mathrm{Win}_A^{\mathtt{bP},\exists}(\mathcal{A}, \chi) = \mathcal{Q} \Rightarrow \mathrm{Win}_A^{\mathtt{fP},1}(\mathcal{A}, \chi) = \mathcal{Q}$.

Proof.

1. Lemma 5.10 states that $\mathrm{Win}_E^{\mathtt{bP},\forall}(\mathcal{A}, \chi) \subseteq \mathrm{Win}_E^{\mathtt{fP},\exists}(\mathcal{A}, \chi)$, and, for any winning condition, the sure winning region of Eve is a subset of her positive winning region.

2. A pure strategy τ' can be defined in a way similar to the 2-player case. However, as τ is a heroic strategy, it is possible that Adam's attempts to get a given distance fails. In this case, he tries again, without incrementing the counter:

   Step 1: Set a counter c to 1 and τ to its initial memory.

   Step 2: Play the strategy τ until either the parity distance is equal to c or a random move deviates from the prescription of τ.

   Step 3: Increment c *if and only if the distance is equal to the current value* c.

   Step 4: Reset the memory for τ and goto to step 2.

   Let z be the smallest positive probability in $\mathcal{A}$. For any value c of the counter, the probability that the counter gets incremented is greater than $z^{c \cdot |\mathcal{Q}|}$. It follows that the probability that the counter gets "stuck" at a finite value is zero, so τ' is almost surely winning. Notice that the same arguments proves that the uniform strategy $\mathtt{uni}_{\mathcal{A}}$ is also almost-sure.

□

Once again, Eve has positional positive winning strategies. We can also compute the almost-sure winning region of Eve, and derive the existence of positional almost sure winning strategies, with the help of the switching algorithm (Algorithm 3.2) and Theorem 3.17.

5.2 Finitary Streett Games

Although finitary versions of any regular condition can be reduced to finitary parity, we consider in this section the special calse of finitary Streett objectives. Indeed, infinitary Streett games are of particular interest in system design, as they correspond to strong fairness constraints [MP92]. The finitary Streett objectives, therefore, give the finitary formulation of strong fairness.

The definition of the finitary Streett condition is even more natural than the finitary parity one: Eve wins if she answers all the requests appearing infinitely often within an unspecified bound b. Figure 5.2, for example, describe a request-service situation:

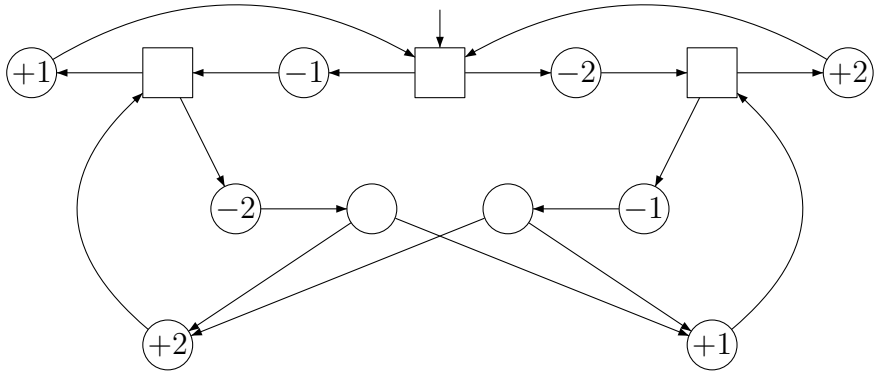


Figure 5.2: A request-service game

There are two requests -1 and -2, which are served by the corresponding responses $+1$ and $+2$. Whenever a request occurs, further requests of the same type are disabled until the request is served; then these requests are enabled again. The controller (Eve) needs to make decisions in the case where

two requests are unserved at the same time: she has to choose which one to serve. Clearly, no matter what the players do, the resulting play is winning for Eve with respect to the classical Streett condition. However, consider the two following strategies for Eve:

Stack strategy Answering first the most recent request, she goes $\swarrow$ from the left $\bigcirc$, and $\searrow$ from the right $\bigcirc$.

Queue strategy Answering first the most ancient request, she goes $\searrow$ from the left $\bigcirc$, and $\swarrow$ from the right $\bigcirc$.

With the stack strategy, the number of transitions between an occurrence of Q_1 and the next occurrence of R_1 can be ultimately unbounded. Hence the stack strategy is not a winning strategy with respect to the finitary Streett objective. The queue strategy, by contrast, ensures not only that every request that is received infinitely often is served, but it also ensures that the number of transitions between the arrival of a request and its serve is at most 6. It is thus winning with respect to the finitary Streett condition.

We define the finitary Streett condition through the notion of *Streett distance sequence*, which is a natural extension of the parity distance sequence:

Definition 5.13 (Streett distance sequence of a play ρ) *Let $(\mathcal{A}, \mathcal{S})$ be a Streett arena of order k, and ρ be a play of $\mathcal{A}$. The distance sequence of ρ for the pair $(-h, +h)$, denoted by $(\mathrm{Sdist}^h(\rho, i))_{i\in\mathbb{N}}$ is defined as follows:*

$$\mathrm{Sdist}^h(\rho, i) = \begin{cases} 0 & \text{if } \chi(\rho_i) \neq -h; \\ \inf\{j > 0 \mid \chi(\rho_{i+j}) = +h\} & \text{if } \chi(\rho_i) = -h \end{cases}$$

The Streett distance sequence of ρ, denoted $(\mathrm{Sdist}(\rho, i))_{i\in\mathbb{N}}$, is defined by $\mathrm{Sdist}(\rho, i) = \max_h\{\mathrm{Sdist}^h(\rho, i)\}$.

The distance for a position i in a play where one or more requests occurs at position i is the number of steps before each request has been satisfied.

Definition 5.14 *Let $(\mathcal{A}, \mathcal{S})$ be a Streett arena of order k. A play ρ of $\mathcal{A}$ belongs to* $\mathrm{finitaryStreett}(\mathcal{A}, \mathcal{S})$ *if and only if* $\limsup_i \mathrm{Sdist}(\rho, i) < \infty$.

We present an algorithm computing the winning regions of a finitary Streett game of degree k with n states and m transitions in time $O(n^2 \cdot m \cdot k \cdot 2^k)$. Hence, the winner problem can be decided in `EXPTIME`. We also show that it is `PSPACE`-hard. For comparison, the winner problem for (infinitary) Streett games is co-`NP`-complete [EJ88], and the winning regions can be computed in

time $O(n^k \cdot k! \cdot m)$ [Hor05]. We also prove, that Eve has strategies with finite memory: $k \cdot 2^k$ memory states are enough, and $2^{\lfloor \frac{k}{2} \rfloor}$ is sometimes necessary. This can be compared with infinitary Streett games, where the lower and upper bounds are $k!$. Once again, Adam may need infinite memory in order to win.

We use one other Streett condition in our proofs, which is called Request-Response condition, and fulfils the same role as the bounded parity condition (5.2.1). Here also, our algorithm for finitary Streett games is a fix-point using an algorithm for Request-Response games as partial algorithm (5.2.2).

5.2.1 Request-Response games

Request-Response conditions are a special case of ω-regular conditions, introduced by Wallmeier, Hütten, and Thomas in [WHT03]. They are defined on Streett arenas, and a play is winning for Eve if and only if for each pair, whenever a request is visited, then now or later a response is visited. Although they were not defined this way, Request-Response conditions can easily be expressed through the Streett distance sequence:

Definition 5.15 *Let $(\mathcal{A}, \mathcal{S})$ be a Streett arena of order k. A play ρ of $\mathcal{A}$ belongs to Request − Response$(\mathcal{A}, \mathcal{S})$ if and only if $\forall i, \mathrm{Sdist}(\rho, i) < \infty$.*

The authors of [WHT03] propose a solution to Request-Response games, which involves a reduction to generalised Büchi games. Starting from a 2-player Streett arena $(\mathcal{A}, \mathcal{S})$ of degree k, with $\mathcal{A} = (\mathcal{Q}, \mathcal{Q}_E, \mathcal{Q}_A, \mathcal{T})$, an expanded arena is built over the vertex set $\mathcal{S}' := \mathcal{S} \times \{0,1\}^k$: the bit vector signals which of the k conditions have an open request. The generalised Büchi condition requires that each bit assumes infinitely often the value 0. Winning strategies with memory k in the reduced game can be translated as strategies with memory $k \cdot 2^k$ in the original Request-Response game. Moreover, it is easy to see that the canonical "round robin" strategy bounds the Streett distance sequence of a play to $|\mathcal{Q}| \cdot k$: at any moment, the "next response" is reached in less than $|\mathcal{Q}|$ moves, and it can take k such response before the current request is served. These results are summarised as Theorem 5.16:

Theorem 5.16 (Request-Response games [WHT03]) *Let $(\mathcal{A}, \mathcal{S})$ be a 2-player Streett arena. The following assertions hold:*

1. (Determinacy). *We have* $\mathrm{Win}_E^{\mathtt{RR}}(\mathcal{A}, \mathcal{S}) = \mathcal{Q} \setminus \mathrm{Win}_E^{\mathtt{RR}}(\mathcal{A}, \mathcal{S})$.

2. (Strategy complexity). *Eve has winning strategies with memory* $k \cdot 2^k$ *which bound the Streett distance sequence to* $|\mathcal{Q}| \cdot k$. *Adam has winning strategies with memory* 2^k.

3. (Time complexity). *The sets* $\mathrm{Win}_E^{\mathtt{RR}}(\mathcal{A}, \mathcal{S})$ *and* $\mathrm{Win}_E^{\mathtt{RR}}(\mathcal{A}, \mathcal{S})$ *can be computed in time* $O(|\mathcal{Q}| \cdot |\mathcal{T}| \cdot 4^k \cdot k^2)$.

Note that, as the Request-Response condition is suffix-closed, Request-Response games could be solved by a fix-point scheme applied to a partial algorithm for Adam. Inspired by the results of (5.1.2), we sought to use the solution of weak Streett games in this role — this would yield a PSPACE algorithm for Request-Response games. However, if the weak Streett condition is indeed harder than the Request-Response condition, it does not comply to the other rule: there are arenas, like the one of Figure 5.3, where Adam wins nowhere with respect to the weak Streett condition, and still manages to win somewhere with respect to to Request-Response condition.

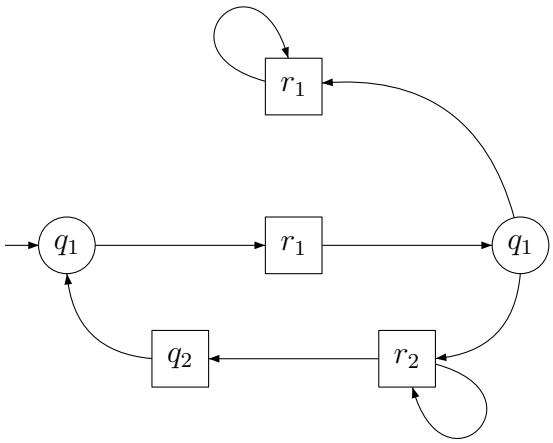


Figure 5.3: $\mathrm{Win}_A^{\mathtt{wS}}(\mathcal{G}) = \emptyset \wedge \mathrm{Win}_A^{\mathtt{RR}}(\mathcal{G}) \neq \emptyset$

5.2.1.1 Complexity and Memory

We give now some precision about the complexity of Request-Response games in terms of complexity classes. The reduction of [WHT03] yields the membership of the winner problem to EXPTIME. Proposition 5.17 shows that it is also PSPACE-hard.

Proposition 5.17 *The problem of the winner in Request-Response games is* PSPACE*-hard.*

Proof. Inspired by the reduction of [NSW02] for weak Streett games, we propose the following reduction from QBF to Request-Response games. Let F be the formula "$\exists x, \forall y, \exists z, (x \vee \overline{y} \vee \overline{z}) \bigwedge (\overline{x} \vee y \vee \overline{z})$". We reduce it to the Streett arena of Figure 5.4. There is a Streett pair for each literal: the request is in lowercase, and the response in uppercase. Furthermore, Σ is a request for *all* the pairs, and $\neg X$ is a response for all the pairs but X. It is clear that Eve can win if and only if F is true. □

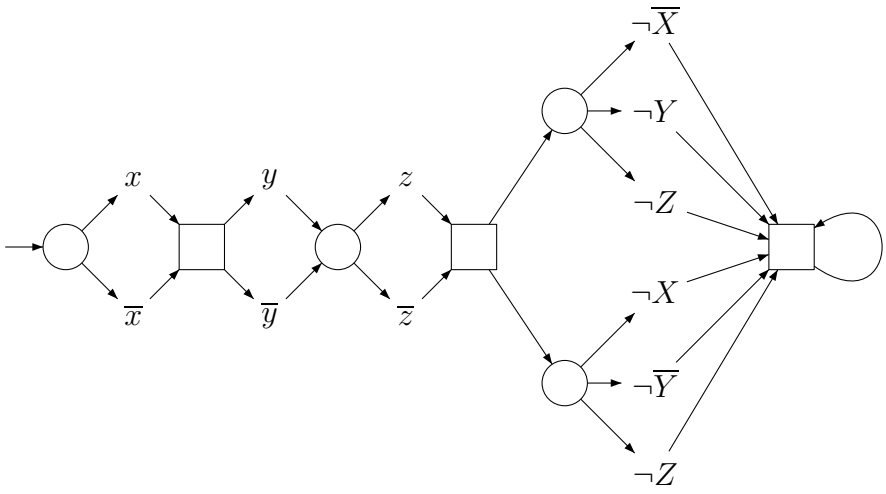


Figure 5.4: Request-Response games are PSPACE-hard

In the case of 1-player games with states of Eve, the problem is NP-complete:

Proposition 5.18 *The winner problem of 1-player Request-Response games with states of Eve is* NP*-hard.*

Proof. We reduce the formula $(x \vee \overline{y} \vee \overline{z}) \bigwedge (\overline{x} \vee y \vee \overline{z})$ to the Streett arena of Figure 5.5. We use the same Streett pairs than in Figure 5.4. In order to win, Eve must choose the dual of a satisfying valuation, then the correct literal in each clause, and finally the satisfying valuation itself. □

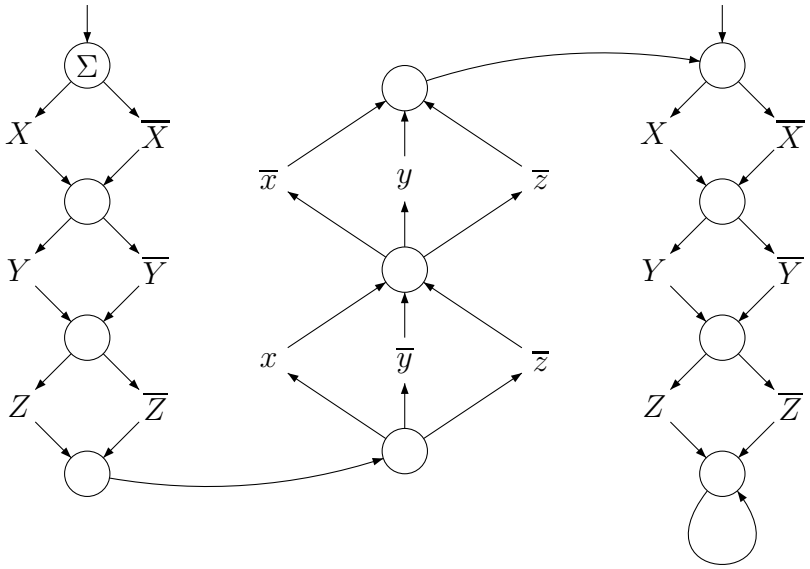


Figure 5.5: 1-player Request-Response games with states of Eve are NP-hard

Proposition 5.19 *The winner problem of 1-player Request-Response games with states of Eve is in* NP.

Proof. If Eve can win, she can do so by first following a path of length at most nk, and then visiting all the states of a strongly connected component. Both can be guessed non-deterministically in polynomial time. Proposition 5.19 follows. □

On the other hand, for 1-player games with states of Adam, the problem is polynomial:

Proposition 5.20 *The winner problem of 1-player Request-Response games with states of Adam is in* PTIME.

Proof. We propose a polynomial procedure to compute the winning regions in a 1-player game where the states belong to Adam:

Step 1: For each pair i, compute the set $X_i = Q_i \setminus \mathrm{Attr}_A(R_i, \mathcal{A})$.

Step 2: Let X be the union of the X_i's. The winning region of Eve is $\mathcal{Q} \setminus \mathrm{Attr}_A(X, \mathcal{A})$, and the winning region of Adam is $\mathrm{Attr}_A(X, \mathcal{A})$.

□

Theorem 5.21 subsumes our results:

Theorem 5.21 *Deciding the winner in 2-player Request-Response games can be done in* EXPTIME *and is* PSPACE*-hard. In the case of 1-player games with states of Eve, it is* NP*-complete. In the case of 1-player games with states of Adam, it is polynomial.*

Lemma 5.22 provides lower bounds for the memory:

Lemma 5.22 *For any k, there is a 2-player Streett arena $(\mathcal{A}, \mathcal{S})$ of order $2k$ such that Eve wins, but has no winning strategy with less than 2^k memory states; there is a $(\mathcal{A}, \mathcal{S})$ of order $2k$ such that Adam wins, but has no winning strategy with less than 2^k memory states.*

Proof. Both witness arenas for $k = 3$ are represented in Figure 5.6. Although there are no literals here, we use the same pairs than in Figure 5.4. In Figure 5.6(a), Eve must mimic the moves of Adam to answer all the request he makes. In Figure 5.6(b), all the requests are made to begin with, and Eve answers to k of them. Adam must request exactly the same ones to ensure that Eve cannot win with her last choice. □

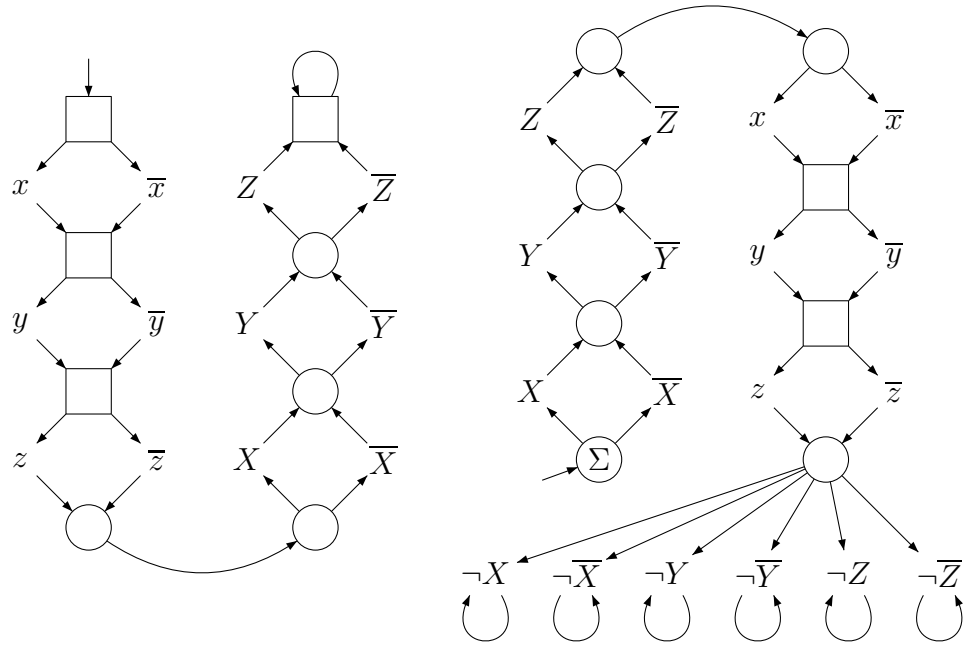
5.2.2 Solving games with finitary Streett objectives

All the arguments of (5.1.3) can be adapted for the case of finitary Streett games, using Request-Response conditions in lieu of bounded parity conditions.

For a given Streett arena $(\mathcal{A}, \mathcal{S})$, the winning regions of the players under the Request-Response and the finitary Streett conditions have the same relation than the winning regions of a parity arena under the bounded parity and the finitary parity conditions:

Lemma 5.23 *Let $(\mathcal{A}, \mathcal{S})$ be a 2-player Streett arena. The following assertions hold:*

1. *Eve's winning region for the Request-Response condition is a subset of her winning region for the finitary Streett condition:* $\mathrm{Win}_E^{\mathtt{RR}}(\mathcal{A}, \mathcal{S}) \subseteq \mathrm{Win}_E^{\mathtt{fS}}(\mathcal{A}, \mathcal{S})$.



(a) Eve (b) Adam

Figure 5.6: Both players need $2^{\lfloor \frac{k}{2} \rfloor}$ memory in Request-Response games

2. *If Adam wins from all the state in $\mathcal{Q}$ for the Request-Response condition, then he wins from all the states in $\mathcal{Q}$ for the finitary Streett condition:* $\mathrm{Win}_E^{\mathtt{RR}}(\mathcal{A}, \mathcal{S}) = \mathcal{Q} \Longrightarrow \mathrm{Win}_A^{\mathtt{fS}}(\mathcal{A}, \mathcal{S}) = \mathcal{Q}$.

Proof.

1. This is a direct consequence of Theorem 5.16.

2. Let τ be a winning strategy for Adam with respect to the Request-Response condition. We define the strategy τ' as follows:

   Step 1: Set a counter c to 1 and τ to its initial memory.

   Step 2: Play the strategy τ until the Streett distance is equal to c.

   Step 3: Increment c.

Step 4: Reset the memory for τ' and go to to step 2.

Let ρ be a play consistent with τ'. We denote by w_c the factor corresponding to the cth iteration of τ'. Notice that if w_c is infinite, the $\{w_d \mid d > c\}$ are not defined. However, w_c is consistent with τ, and would be winning for Eve with respect to the Request-Response condition if it was infinite. Thus, each w_c is finite, and ρ is the concatenation of the $\{w_c \mid c \geq 1\}$. It follows that ρ and τ' are winning for Adam with respect to the finitary Streett condition.

□

We can thus solve finitary Streett games, be they 2-player or $2\frac{1}{2}$-player, with fix-point arguments, using a Request-Response solver as a partial algorithm. Theorem 5.24 follows:

Theorem 5.24 (finitary Streett games) *Let $(\mathcal{A}, \mathcal{S})$ be a 2-player Streett arena. The following assertions hold:*

1. (Determinacy). *We have* $\mathrm{Win}_E^{\mathtt{fs}}(\mathcal{A}, \mathcal{S}) = \mathcal{Q} \setminus \mathrm{Win}_A^{\mathtt{fs}}(\mathcal{A}, \mathcal{S})$.
2. (Strategy complexity). *Eve has winning strategies with memory $k \cdot 2^k$ which ultimately bound the Streett distance sequence to $|\mathcal{Q}| \cdot k$. In general, Adam has no winning strategies with finite memory.*
3. (Time complexity). *The sets* $\mathrm{Win}_E^{\mathtt{fs}}(\mathcal{A}, \mathcal{S})$ *and* $\mathrm{Win}_A^{\mathtt{fs}}(\mathcal{A}, \mathcal{S})$ *can be computed in time $O(|\mathcal{Q}|^2 \cdot |\mathcal{T}| \cdot 4^k \cdot k^2)$.*

All the usual qualitative variations of these results still hold for $2\frac{1}{2}$-player Streett arenas. As in the case of finitary parity games, Adam has positional randomised winning strategies.

Most of the complexity results we got for Request-Response games carry to the case of finitary Streett games. In particular, a `PSPACE` algorithm for Request-Response games would immediately lead to a `PSPACE` algorithm for finitary Streett games.

Theorem 5.25 *The problem of the winner in 2-player finitary Streett games belong to* `EXPTIME`*. It is also* `PSPACE`*-hard. In the case of 1-player games, it is polynomial.*

Proof. The reduction of Figure 5.4 can be easily adapted, by adding a transition from the last state to the first. In this game, Adam can wait for as long as he wishes with an open request between two successive rounds. Notice that this is not possible in Figure 5.5, since the all the states belongs to Eve. Indeed, for any 1-player Streett arena with states of Eve $(\mathcal{A}, \mathcal{S}), \mathrm{Win}_E^{\mathtt{fs}}(\mathcal{A}, \mathcal{S}) = \mathrm{Win}_E^{\mathtt{St}}(\mathcal{A}, \mathcal{S})$. As the winner problem of 1-player Streett games is polynomial, so is the winner problem of 1-player finitary Streett games. The algorithm for 1-player arenas with states of Adam can easily be adapted for the finitary Streett condition. □

Likewise, the lower bounds in memory derive from the ones for Request-Response games:

Lemma 5.26 *For any k, there is a 2-player Streett arena $(\mathcal{A}, \mathcal{S})$ of order $2k$ such that Eve wins, but has no winning strategy with less than 2^k memory states.*

Proof. The arena of Figure 5.6 can also be adapted for finitary games, by adding a transition from the last state to the first. □

5.3 Perspectives

Our contribution to the study of finitary games unearthed a quite complete set of complexity bounds for the various problems induced by finitary games.

The polynomial algorithm for finitary parity is especially pleasing: from a verification point of view, it offers a much cheaper alternative to classical parity games, while removing only pathological behaviours that are often unsatisfactory to begin with.

It would be nice, of course, to get the exact complexity of finitary Streett games, through either a `PSPACE` algorithm or a proof of `EXPTIME`-hardness. Our most promising prospect, however, is to refine the analysis of finitary games, by taking in account not only the mere satisfaction of the winning condition but also the quantitative aspect of minimising the delay between the requests and the subsequent responses, in the spirit of [HTW08].

Chapter 6

Conclusion

"School's out for summer
School's out forever"

School's Out
Alice Cooper

In Chapter 2, we have described a new approach to the fundamental problem of reachability games, linking the complexity to an intuitive parameter, the number of random vertices. Furthermore, the complexity of our permutation algorithm is comparable to the best known deterministic algorithms, and the "permutation improvement" scheme makes it a candidate for polynomiality. The obvious problem now would be to find a polynomial algorithm computing the values of $2\frac{1}{2}$-player reachability games. However, this problem is harder than the winner problem in 2-player parity games. An interesting, yet more reasonable objective would be to prove that our permutation improvement algorithm is polynomial on $1\frac{1}{2}$-player games.

We considered then in Chapter 3 the general case of prefix-independent conditions, and proved their optimal determinacy. We also adapt our permutation algorithm to compute the values of any prefix-independent games with a single non-deterministic guess and a qualitative algorithm. It is well known that Borel games in general are not optimally determined , but it does not mean that quantitative determinacy is the best we can do: we do not know any counter-examples for qualitative determinacy.

We came back in Chapter 4 to the origins of infinite games with the venerable case of Muller games. A first result was the membership of the winner

problem in explicit games to `PTIME`, and we would like to check whether this result can be adapted for other tree automata problems. Our main result, however, is the tight bound on the necessary memory for randomised strategies. We also got smaller witness arenas for the lower bounds, leading naturally to the question of the resilience of these bounds in the case of arenas of polynomial size. Another logical extension would be to get memory bounds for any ω-regular winning condition: even Muller games are a normal form, whose cost may be reduced in some cases.

A question that arose during this study was the problem of the correct definition of a randomised strategy with memory: by contrast with the case of pure strategy, there is not an obvious "standard" notion, and we have shown that semi-randomised strategies and strategies with random memory really are two different models. This gives perspectives in two directions: first, does our upper bound for Muller games hold for semi-randomised strategies; second, how do these two models relate together, and with other models of randomised strategies with memory, *e.g.* strategies where the move and the update are independent.

Lastly, the finitary games we studied in Chapter 5 came more from the model-checking tradition: quite often, a really infinitary controller with unbounded delays is unacceptable. Our polynomial algorithm for parity games yields an efficient approach for any finitary ω-regular game, through the Zielonka tree reduction. We also studied the finitary version of strong fairness, with the case of finitary Streett games. Our reduction to request-response games suggests a new way to consider these games, where a play yields a reward instead of a winner, in the spirit of [HTW08].

List of Figures

List of Algorithms

Bibliography

[AH98] Rajeev Alur and Thomas A. Henzinger. Finitary Fairness. *ACM Transactions on Programming Languages and Systems*, 20(6):1171–1194, 1998.

[AS85] Bowen Alpern and Fred B. Schneider. Defining Liveness. *Information Processing Letters*, 21(4):181–185, 1985.

[BBBM08] Nathalie Bertrand, Patricia Bouyer, Thomas Brihaye, and Nicolas Markey. Quantitative Model-Checking of One-Clock Timed Automata under Probabilistic Semantics. In *Proceedings of the 5th International Conference on Quantitative Evaluation of Systems, QEST'08*, pages 55–64. IEEE Computer Society, 2008.

[BBJ+08] Patricia Bouyer, Thomas Brihaye, Marcin Jurdziński, Ranko Lazić, and Michał Rutkowski. Average-Price and Reachability-Price Games on Hybrid Automata with Strong Resets. In *Proceedings of the 6th International Conference on Formal Modelling and Analysis of Timed Systems, FORMATS'08*, volume 5215 of *Lecture Notes in Computer Science*, pages 63–77. Springer-Verlag, 2008.

[BCHG+97] Christel Baier, Edmund M. Clarke, Vassili Hartonas-Garmhausen, Marta Z. Kwiatkowska, and Mark Ryan. Symbolic Model Checking for Probabilistic Processes. In *Proceedings of the 24th International Colloquium on Automata, Languages and Programming, ICALP'97*, volume 1256 of *Lecture Notes in Computer Science*, pages 430–440. Springer-Verlag, 1997.

[BDM+06] Mikolaj Bojanczyk, Claire David, Anca Muscholl, Thomas Schwentick, and Luc Segoufin. Two-variable logic on data trees

and XML reasoning. In *Proceedings of the 25th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, PODS'06*, pages 10–19. ACM Press, 2006.

[BFLP03] Sébastien Bardin, Alain Finkel, Jérôme Leroux, and Laure Petrucci. FAST: Fast Acceleration of Symbolikc Transition Systems. In *Proceedings of the 15th International Conference on Computer Aided Verification, CAV'03*, volume 2725 of *Lecture Notes in Computer Science*, pages 118–121. Springer-Verlag, 2003.

[BJNT00] Ahmed Bouajjani, Bengt Jonsson, Marcus Nilsson, and Tayssir Touili. Regular Model Checking. In *Proceedings of the 12th International Conference on Computer Aided Verification, CAV'00*, volume 1855 of *Lecture Notes in Computer Science*, pages 403–418. Springer-Verlag, 2000.

[BKH99] Christel Baier, Joost-Pieter Katoen, and Holger Hermanns. Approximate Symbolic Model Checking of Continuous-Time Markov Chains. In *Proceedings of the 10th International Conference on Concurrency Theory, CONCUR'99*, volume 1664 of *Lecture Notes in Computer Science*, pages 146–161. Springer-Verlag, 1999.

[BL69] J. Richard Büchi and Lawrence H. Landweber. Solving Sequential Conditions by Finite-State Strategies. *Transactions of the American Mathematical Society*, 138:295–311, 1969.

[Bla92] Andreas Blass. A Game Semantics for Linear Logic. *Annals of Pure and Applied Logic*, 56(1–3):183–220, 1992.

[BLV96] Nils Buhrke, Helmut Lescow, and Jens Vöge. Strategy Construction in Infinite Ganes with Streett and Rabin Chain Winning Conditions. In *Proceedings of the 2nd International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS'96*, volume 1055 of *Lecture Notes in Computer Science*, pages 207–224. Springer-Verlag, 1996.

[Bor21] Émile Borel. La théorie du Jeu et les équations intégrales 'a noyau symétrique. *Comptes-rendus de l'Académie des Sciences*, 1923:1304–1308, 1921.

[Büc62] J. Richard Büchi. On a decision method in restricted second-order arithmetic. In *Proceedings of the 1st International Congress of Logic, Methodology, and Philosophy of Science, CLMPS'60*, pages 1–11. Stanford University Press, 1962.

[CdAH04] Krishnendu Chatterjee, Luca de Alfaro, and Thomas A. Henzinger. Trading Memory for Randomness. In *Proceedings of the 1st International Conference on Quantitative Evaluation of Systems, QEST'04*, pages 206–217. IEEE Computer Society, 2004.

[CdAH05] Krishnendu Chatterjee, Luca de Alfaro, and Thomas A. Henzinger. The Complexity of Stochastic Rabin and Streett Games. In *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming, ICALP'05*, volume 3580 of *Lecture Notes in Computer Science*, pages 878–890. Springer-Verlag, 2005.

[CE81] Edmund M. Clarke and E. Allen Emerson. Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic. In *Logic of Programs*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer-Verlag, 1981.

[CH06a] Krishnendu Chatterjee and Thomas A. Henzinger. Finitary Winning in ω-Regular Games. In *Proceedings of the 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS'06*, volume 3920 of *Lecture Notes in Computer Science*, pages 257–271. Springer-Verlag, 2006.

[CH06b] Krishnendu Chatterjee and Thomas A. Henzinger. Strategy Improvement for Stochastic Rabin and Streett Games. In *Proceedings of the 17th International Conference on Concurrency Theory, CONCUR'06*, volume 4137 of *Lecture Notes in Computer Science*, pages 375–389. Springer-Verlag, 2006.

[CH08a] Julien Cristau and Florian Horn. Graph Games on Ordinals. In *Proceedings of the 28th International Conference on the Foundations of Software Technology and Theoretical Computer Science, FSTTCS'08*, volume 2 of *Leibniz International Proceedings in Informatics*, pages 143–154. Schloß Dagstuhl, 2008.

[CH08b] Julien Cristau and Florian Horn. On Reachability Games of Ordinal Length. In *Proceedings of the 34th Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM'08*, volume 4910 of *Lecture Notes in Computer Science*, pages 211 – 221. Springer-Verlag, 2008.

[Cha06] Krishnendu Chatterjee. Linear Time Algorithm for Weak Parity Games. Technical Report UCB/EECS-2006-153, University of California at Berkeley, 2006.

[Cha07a] Krishnendu Chatterjee. Concurrent Games with Tail Objectives. *Theoretical Computer Science*, 388(1–2):181–198, 2007.

[Cha07b] Krishnendu Chatterjee. Optimal Strategy Synthesis in Stochastic Muller Games. In *Proceedings of the 10th International Conference on the Foundations of Software Science and Computational Structures, FoSSaCS'07*, volume 4423 of *Lecture Notes in Computer Science*, pages 138–152. Springer-Verlag, 2007.

[Cha07c] Krishnendu Chatterjee. Stochastic Müller Games are `PSPACE`-Complete. In *Proceedings of the 27th International Conference on the Foundations of Software Technology and Theoretical Computer Science, FSTTCS'07*, volume 4855 of *Lecture Notes in Computer Science*, pages 436–448. Springer-Verlag, 2007.

[Cha07d] Krishnendu Chatterjee. *Stochastic ω-regular Games*. PhD thesis, University of California at Berkeley, 2007.

[CHH08] Krishnendu Chatterjee, Thomas A. Henzinger, and Florian Horn. Stochastic Finitary Games. Technical Report 2008–002, Laboratoire d'Informatique Algorithmique: Fondements et Applications, CNRS UMR 7089, 2008.

[CHH09] Krishnendu Chatterjee, Thomas A. Henzinger, and Florian Horn. Finitary Winning in ω-regular Games. To appear in ACM Transactions on Computational Logic, 2009.

[CHM+08] Arnaud Carayol, Matthew Hague, Antoine Meyer, C.-H. Luke Ong, and Olivier Serre. Winning Regions of Higher-Order Pushdown Games. In *Proceedings of the 23rd Annual IEEE Sympo-*

sium on Logic in Computer Science, LICS'08, pages 193–204. IEEE Computer Society, 2008.

[Chr03] Juliusz Chroboczek. *Game Semantics and Subtyping*. PhD thesis, University of Edinburgh, 2003.

[Chu62] Alonzo Church. Logic, arithmetic, and automata. In *Proceedings of the International Congress of Mathematicians*, pages 23–35, 1962.

[CJH03] Krishnendu Chatterjee, Marcin Jurdziński, and Thomas A. Henzinger. Simple Stochastic Parity Games. In *Proceedings of the 12th EACSL Annual Conference on Computer Science and Logic, CSL'03*, volume 2803 of *Lecture Notes in Computer Science*, pages 100–113. Springer-Verlag, 2003.

[CJH04] Krishnendu Chatterjee, Marcin Jurdziński, and Thomas A. Henzinger. Quantitative Stochastic Parity Games. In *Proceedings of the 15th Symposium on Discrete Algorithms, SODA'04*, pages 121–130. Society for Industrial and Applied Mathematics, 2004.

[CL08] Thomas Colcombet and Christof Löding. The Nesting-Depth of Disjunctive μ-calculus for Tree Languages and the Limitedness Problem. In *Proceedings of the 17th EACSL Annual Conference on Computer Science and Logic, CSL'08*, volume 5213 of *Lecture Notes in Computer Science*, pages 416–430. Springer-Verlag, 2008.

[Cla08] Edmund M. Clarke. The Birth of Model Checking. In *25 Years of Model Checking*, volume 5000 of *Lecture Notes in Computer Science*, pages 1–26. Springer-Verlag, 2008.

[CN06] Thomas Colcombet and Damian Niwinski. On the Positional Determinacy of Edge-Labeled Games. *Theoretical Computer Science*, 352(1–3):190–196, 2006.

[Con92] Anne Condon. The Complexity of Stochastic Games. *Information and Computation*, 96(2):203–224, 1992.

[Con93] Anne Condon. On Algorithms for Simple Stochastic Games. In *Advances in Computational Complexity Theory*, volume 13 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 51–73. American Mathematical Society, 1993.

[Cou38] A. Augustin Cournot. *Recherches sur les principes mathématiques de la théorie des richesses.* 1838.

[CY95] Costas Courcoubetis and Mihalis Yannakakis. The Complexity of Probabilistic Verification. *Journal of the ACM*, 42(4):857–907, 1995.

[dA97] Luca de Alfaro. *Formal Verification of Probabilistic Systems.* PhD thesis, Stanford University, 1997.

[dAFH+03] Luca de Alfaro, Marco Faella, Thomas A. Henzinger, Rupak Majumdar, and Marielle Stoelinga. The Element of Surprise in Timed Games. In *Proceedings of the 14th International Conference on Concurrency Theory, CONCUR'03*, volume 2761 of *Lecture Notes in Computer Science*, pages 142–156. Springer-Verlag, 2003.

[dAH00] Luca de Alfaro and Thomas A. Henzinger. Concurrent ω-regular Games. In *Proceedings of the 16th Annual IEEE Symposium on Logic in Computer Science, LICS'00*, pages 141–154. IEEE Computer Society, 2000.

[dAHK98] Luca de Alfaro, Thomas A. Henzinger, and Orna Kupferman. Concurrent Reachability Games. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science, FoCS'98*, pages 564–575. IEEE Computer Society, 1998.

[Der62] Cyrus Derman. On Sequential Decisions and Markov Chains. *Management Science*, 9(1):16–24, 1962.

[Dic55] Philip K. Dick. *Solar Lottery.* Ace Books, 1955.

[Dix82] John D. Dixon. Exact Solution of Linear Equations Using p-adic Expansions. *Numerische Mathematik*, 40:137–141, 1982.

[DJP03] Nachum Dershowitz, D. N. Jayasimha, and Seungjoon Park. Bounded Fairness. In *Verification: Theory and Practice, Essays dedicated to Zohar Manna on the occasion of his 64th birthday*, volume 2772 of *Lecture Notes in Computer Science*, pages 304–317. Springer-Verlag, 2003.

[DJW97] Stefan Dziembowski, Marcin Jurdziński, and Igor Walukiewicz. How Much Memory is Needed to Win Infinite Games? In *Proceedings of the 12th Annual IEEE Symposium on Logic in Computer Science, LICS'97*, pages 99–110. IEEE Computer Society, 1997.

[DK00] Michael J. Dinneen and Bakhadyr Khoussainov. Update Networks and Their Routing Strategies. In *Proceedings of the 26th International Workshop on Graph-Theoretic Concepts in Computer Science, WG'00*, volume 1928 of *Lecture Notes in Computer Science*, pages 127–136. Springer-Verlag, 2000.

[Dur96] Richard Durett. *Probability Theory and Examples.* Duxbury Press, 1996.

[EJ88] E. Allen Emerson and Charanjit S. Jutla. The Complexity of Tree Automata and Logics of Programs. In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science, FoCS'88*, pages 328–337. IEEE Computer Society, 1988.

[EJ91] E. Allen Emerson and Charanjit S. Jutla. Tree Automata, μ-Calculus and Determinacy. In *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science, FoCS'91*, pages 368–377. IEEE Computer Society, 1991.

[EL85] E. Allen Emerson and Chin-Laung Lei. Modalities for Model Checking: Branching Time Strikes Back. In *Proceedings of the 12th Annual ACM Symposium on Principles of Programming Languages, POPL'85*, pages 84–96, 1985.

[EWS01] Kousha Etessami, Thomas Wilke, and Rebecca A. Schuller. Fair Simulation Relations, Parity Games, and State Space Reduction for Büchi Automata. In *Proceedings of the 28th International Colloquium on Automata, Languages and Programming,*

ICALP'01, volume 2076 of *Lecture Notes in Computer Science*, pages 694–707. Springer-Verlag, 2001.

[FGK08] Diana Fischer, Erich Grädel, and Łukasz Kaiser. Model Checking Games for the Quantitative μ-calculus. In *Proceedings of the 25th International Symposium on Theoretical Aspects of Computer Science, STACS'08*, volume 1 of *Leibniz International Proceedings in Informatics*, pages 301–312. Schloß Dagstuhl, 2008.

[Fin07] Irving Finkel. *Ancient Board Games in Perspective*. British Museum Publications, 2007.

[FPT04] Alex Fabrikant, Christos H. Papadimitriou, and Kunal Talwar. The complexity of pure Nash equilibria. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, STOC'04*, pages 604–612. ACM Press, 2004.

[GH82] Yuri Gurevich and Leo Harrington. Trees, Automata, and Games. In *Proceedings of the 14th Annual ACM Symposium on Theory of Computing, STOC'82*, pages 60–65. ACM Press, 1982.

[GH08] Hugo Gimbert and Florian Horn. Simple Stochastic Games with Few Random Vertices are Easy to Solve. In *Proceedings of the 11th International Conference on the Foundations of Software Science and Computational Structures, FoSSaCS'08*, volume 4962 of *Lecture Notes in Computer Science*, pages 5 – 19. Springer-Verlag, 2008.

[GH09] Hugo Gimbert and Florian Horn. Solving Simple Stochastic Games with Few Random Vertices. *Logical Methods in Computer Science*, 5(2):9th, 2009.

[Gil57] Dean Gillette. *Stochastic Games with Zero Stop Probability*, volume 3 of *Contributions to the Theory of Games*, pages 179–187. Princeton University Press, 1957.

[Gim06] Hugo Gimbert. *Jeux Positionels*. PhD thesis, Université Paris VII – Denis Diderot, 2006.

[GLZ04] Paul Gastin, Benjamin Lerman, and Marc Zeitoun. Distributed Games with Causal Memory Are Decidable for Series-Parallel Systems. In *Proceedings of the 24th International Conference on the Foundations of Software Technology and Theoretical Computer Science, FSTTCS'04*, volume 3328 of *Lecture Notes in Computer Science*, pages 275–286. Springer-Verlag, 2004.

[GMSZ02] Blaise Genest, Anca Muscholl, Helmut Seidl, and Marc Zeitoun. Infinite-State High-Level MSCs: Model-Checking and Realizability. In *Proceedings of the 29th International Colloquium on Automata, Languages and Programming, ICALP'02*, volume 2380 of *Lecture Notes in Computer Science*, pages 657–668. Springer-Verlag, 2002.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, STOC'87*, pages 218–229. ACM Press, 1987.

[GTW02] Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001]*, volume 2500 of *Lecture Notes in Computer Science*. Springer-Verlag, 2002.

[GU08] Erich Grädel and Michael Ummels. Solution Concepts and Algorithms for Infinite Multiplayer Games. In *New Perspectives on Games and Interaction*, volume 5 of *Texts in Logic and Games*. Amsterdam University Press, 2008.

[GZ05] Hugo Gimbert and Wieslaw Zielonka. Games Where You Can Play Optimally Without Any Memory. In *Proceedings of the 16th International Conference on Concurrency Theory, CONCUR'05*, volume 3653 of *Lecture Notes in Computer Science*, pages 428–442. Springer-Verlag, 2005.

[GZ07] Hugo Gimbert and Wieslaw Zielonka. Applying Blackwell Optimality: priority mean-payoff games as limits of multi-discounted

games. In *Logic and Automata: History and Perspective*, volume 2 of *Texts in Logic and Games*, pages 331–355. Amsterdam University Press, 2007.

[Hal07] Nir Halman. Simple Stochastic Games, Parity Games, Mean Payoff Games and Discounted Payoff Games are all LP-Type Problems. *Algorithmica*, 49:37–50, 2007.

[HD05] Paul Hunter and Anuj Dawar. Complexity Bounds for Regular Games. In *Proceedings of the 30th International Symposium on Mathematical Foundations of Computer Science, MFCS'05*, volume 3618 of *Lecture Notes in Computer Science*, pages 495–506. Springer-Verlag, 2005.

[HK66] Alan J. Hoffman and Richard M. Karp. On Nonterminating Stochastic Games. *Management Science*, 12(5):359–370, 1966.

[Hor05] Florian Horn. Streett Games on Finite Graphs. In the *2nd Workshop on Games in Design and Verification*, GDV'05, 2005.

[Hor07a] Florian Horn. Dicing on the Streett. *Information Processing Letters*, 104(1):1–9, 2007.

[Hor07b] Florian Horn. Faster Algorithms for Finitary Games. In *Proceedings of the 13th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS'07*, volume 4424 of *Lecture Notes in Computer Science*, pages 472–484. Springer-Verlag, 2007.

[Hor09] Florian Horn. Random Fruits on the Zielonka Tree. In *Proceedings of the 26th International Symposium on Theoretical Aspects of Computer Science, STACS'09*, volume 3 of *Leibniz International Proceedings in Informatics*, pages 541–552. Schloß Dagstuhl, 2009.

[How60] Ronald A. Howard. *Dynamic Programming and Markov Processes*. M.I.T. Press, 1960.

[HTW08] Florian Horn, Wolfgang Thomas, and Nico Wallmeier. Optimal Strategy Synthesis in Request-Response Games. In *Proceedings of the 6th International Symposium on Automated Technology for Verification and Analysis, ATVA'08*, volume 5311 of

Lecture Notes in Computer Science, pages 361–373. Springer-Verlag, 2008.

[IK02] Hajime Ishihara and Bakhadyr Khoussainov. Complexity of Some Infinite Games Played on Finite Graphs. In *Proceedings of the 28th International Workshop on Graph-Theoretic Concepts in Computer Science, WG'02*, volume 2573 of *Lecture Notes in Computer Science*, pages 270–281. Springer-Verlag, 2002.

[JKH02] Marcin Jurdziński, Orna Kupferman, and Thomas A. Henzinger. Trading Probability for Fairness. In *Proceedings of the 11th EACSL Annual Conference on Computer Science and Logic, CSL'02*, volume 2471 of *Lecture Notes in Computer Science*, pages 292–305. Springer-Verlag, 2002.

[JPZ06] Marcin Jurdziński, Mike Paterson, and Uri Zwick. A Deterministic Subexponential Algorithm for Solving Parity Games. In *Proceedings of the 17th Symposium on Discrete Algorithms, SODA'06*, pages 117–123. ACM Press, 2006.

[Jur00] Marcin Jurdziński. Small Progress Measures for Solving Parity Games. In *Proceedings of the 17th International Symposium on Theoretical Aspects of Computer Science, STACS'00*, volume 1770 of *Lecture Notes in Computer Science*, pages 290–301. Springer-Verlag, 2000.

[Jur07] Marcin Jurdziński, 2007. Personal communication.

[Kav86] Gregory S. Kavka. *Hobbesian moral and political theory*. Princeton University Press, 1986.

[Kha79] Leonid G. Khachiyan. A Polynomial Algorithm in Linear Programming. *Soviet Mathematics Doklady*, 20:191–194, 1979.

[Kla97] Nils Klarlund. Mona & Fido: The Logic-Automaton Connection in Practice. In *Proceedings of the 11th EACSL Annual Conference on Computer Science and Logic, CSL'98*, volume 1414 of *Lecture Notes in Computer Science*, pages 311–326. Springer-Verlag, 1997.

[KPV07] Orna Kupferman, Nir Piterman, and Moshe Y. Vardi. From Liveness to Promptness. In *Proceedings of the 19th International Conference on Computer Aided Verification, CAV'07*, volume 4590 of *Lecture Notes in Computer Science*, pages 406–419. Springer-Verlag, 2007.

[LT00] Christof Löding and Wolfgang Thomas. Alternating Automata and Logics over Infinite Words. In *Proceedings of the IFIP International Conference on Theoretical Computer Science, IFIP TCS'00*, volume 1872 of *Lecture Notes in Computer Science*, pages 521–535. Springer-Verlag, 2000.

[Maj03] Rupak Majumdar. *Symbolic Algorithms for Verification and Control*. PhD thesis, University of California at Berkeley, 2003.

[Mar75] Donald A. Martin. Borel Determinacy. *Annals of Mathematics*, 102(2):363–371, 1975.

[Mar98] Donald A. Martin. The Determinacy of Blackwell Games. *Journal of Symbolic Logic*, 63(4):1565–1581, 1998.

[Mar07] George R.R. Martin. Unsound Variations. In *Dreamsongs: A RRetrospective*, volume 2, pages 478–531. Orion Books, 2007.

[Maz75] Antoni W. Mazurkiewicz. Parallel Recursive Program Schemes. In *Proceedings of the 4th International Symposium on Mathematical Foundations of Computer Science, MFCS'75*, volume 32 of *Lecture Notes in Computer Science*, pages 75–87. Springer-Verlag, 1975.

[McM93] Kenneth L. McMillan. *Symbolic Model Checking*. PhD thesis, Carnegie Mellon University, 1993.

[McN65] Robert McNaughton. Finite-state Infinite Games. Technical report, Massachusetts Institute of Technology, 1965. Project MAC.

[McN93] Robert McNaughton. Infinite Games Played on Finite Graphs. *Annals of Pure and Applied Logic*, 65(2):149–184, 1993.

[Mor08] Pierre Moro. *Techniques de vérification basées sur des représentations symboliques par automates et l'abstraction guidée par les contre-exemples.* PhD thesis, Université Paris VII – Denis Diderot, 2008.

[Mos91] Andrzej W. Mostowski. Games with Forbidden Positions. Technical Report 78, University of Gdansk, 1991.

[MP92] Zohar Manna and Amir Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification.* Springer-Verlag, 1992.

[MS96] Ashok P. Maitra and William D. Sudderth. *Discrete Gambling and Stochastic Games.* Springer-Verlag, 1996.

[MTY05] Parthasarathy Madhusudan, P.S. Thiagarajan, and Shaofa Yang. The MSO Theory of Connectedly Communicating Processes. In *Proceedings of the 25th International Conference on the Foundations of Software Technology and Theoretical Computer Science, FSTTCS'05*, volume 3821 of *Lecture Notes in Computer Science*, pages 201–212. Springer-Verlag, 2005.

[Mun07] Randall Munroe. Effect an Effect. *XKCD 326*, 2007.

[MW03] Swarup Mohalik and Igor Walukiewicz. Distributed Games. In *Proceedings of the 23th International Conference on the Foundations of Software Technology and Theoretical Computer Science, FSTTCS'03*, volume 2914 of *Lecture Notes in Computer Science*, pages 338–351. Springer-Verlag, 2003.

[Nas50] John F. Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of Science of the USA*, 36(1):48–49, 1950.

[NRY96] Anil Nerode, Jeffrey B. Remmel, and Alexander Yakhnis. McNaughton Games and Extracting Strategies for Concurrent Programs. *Annals of Pure and Applied Logic*, 78(1–3):203–242, 1996.

[NSW02] Jakub Neumann, Andrzej Szepietowski, and Igor Walukiewicz. Complexity of Weak Acceptance Conditions in Tree Automata. *Information Processing Letters*, 84(4):181–187, 2002.

[OG76] Susan S. Owicki and David Gries. Verifying Properties of Parallel Programs: An Axiomatic Approach. *Communications of the ACM*, 19(5):279–285, 1976.

[Pnu77] Amir Pnueli. The Temporal Logic of Programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science, FoCS'77*, pages 46–57. IEEE Computer Society, 1977.

[PP06] Nir Piterman and Amir Pnueli. Faster Solutions of Rabin and Streett Games. In *Proceedings of the 21st Annual IEEE Symposium on Logic in Computer Science, LICS'06*, pages 275–284. IEEE Computer Society, 2006.

[PR89] Amir Pnueli and Roni Rosner. On the Synthesis of a Reactive Module. In *Proceedings of the 16th Annual ACM Symposium on Principles of Programming Languages, POPL'89*, pages 179–190, 1989.

[QS82] Jean-Pierre Queille and Joseph Sifakis. A Temporal Logic to Deal with Fairness in Transition Systems. In *Proceedings of the 23th Annual Symposium on Foundations of Computer Science, FoCS'82*, pages 217–225. IEEE Computer Society, 1982.

[Rab69] Michael Oser Rabin. Automata on Infinite Objects and Church's Problem. *Transactions of the American Mathematical Society*, 141:1–35, 1969.

[Ren88] James Renegar. A polynomial-time algorithm, based on newton's method, for linear programming. *Mathematical Programming*, 40(1):59–93, 1988.

[Rou05] Tim Roughgarden. *Selfish Routing and the Price of Anarchy*. Massachusetts Institute of Technology, 2005.

[RS08] Alexander Rabinovich and Amit Shomrat. Selection and Uniformization Problems in the Monadic Theory of Ordinals: A Survey. In *Pillars of Computer Science, Essays Dedicated to Boris (Boaz) Trakhtenbrot on the Occasion of his 85th Birthday*, volume 4800 of *Lecture Notes in Computer Science*, pages 571–588. Springer-Verlag, 2008.

[Sch07] Sven Schewe. Solving Parity Games in Big Steps. In *Proceedings of the 27th International Conference on the Foundations of Software Technology and Theoretical Computer Science, FSTTCS'07*, volume 4855 of *Lecture Notes in Computer Science*, pages 449–460. Springer-Verlag, 2007.

[Ser05] Olivier Serre. *Contribution à l'étude des jeux sur des graphes de processus à pile*. PhD thesis, Université Paris VII – Denis Diderot, 2005.

[Sha53] Lloyd S. Shapley. Stochastic Games. In *Proceedings of the National Academy of Science of the USA*, volume 39, pages 1095–1100, 1953.

[Smi82] John Maynard Smith. *Evolution and the Theory of Games*. Cambridge University Press, 1982.

[SW74] Ludwig Staiger and Klaus W. Wagner. Automatentheoretische und automatenfreie Charakterisierungen topologischer Klassen regulärer Folgenmengen. *Elektronische Informationsverarbeitung und Kybernetik*, 10(7):379–392, 1974.

[Tho95] Wolfgang Thomas. On the Synthesis of Strageties in Infinite Games. In *Proceedings of the 12th International Symposium on Theoretical Aspects of Computer Science, STACS'95*, volume 900 of *Lecture Notes in Computer Science*, pages 1–13. Springer-Verlag, 1995.

[Var85] Moshe Y. Vardi. Automatic Verification of Probabilistic Concurrent Finite-State Programs. In *Proceedings of the 26th Annual Symposium on Foundations of Computer Science, FoCS'85*, pages 327–338. IEEE Computer Society, 1985.

[VJ00] Jens Vöge and Marcin Jurdziński. A Discrete Strategy Improvement Algorithm for Solving Parity Games. In *Proceedings of the 12th International Conference on Computer Aided Verification, CAV'00*, volume 1855 of *Lecture Notes in Computer Science*, pages 202–215. Springer-Verlag, 2000.

[vNM44] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1944.

[VW86] Moshe Y. Vardi and Pierre Wolper. An Automata-Theoretic Approach to Automatic Program Verification. In *Proceedings of the 1st Annual IEEE Symposium on Logic in Computer Science, LICS'86*, pages 332–344. IEEE Computer Society, 1986.

[Wal96] Igor Walukiewicz. Pushdown Processes: Games and Model Checking. In *Proceedings of the 8th International Conference on Computer Aided Verification, CAV'96*, volume 1102 of *Lecture Notes in Computer Science*, pages 62–74. Springer-Verlag, 1996.

[WHT03] Nico Wallmeier, Patrick Hütten, and Wolfgang Thomas. Symbolic Synthesis of Finite-State Controllers for Request-Response Specifications. In *Proceedings of the 8th International Conference on Implementation and Application of Automata, CIAA'03*, volume 2759 of *Lecture Notes in Computer Science*, pages 11–22. Springer-Verlag, 2003.

[Zer13] Ernst Zermelo. Uber eine Anwendung der Mengenlehre auf die Theorie des Schachspiels. In *Proceedings of the 5th International Congress of Mathematicians, ICM'13*, volume 2, pages 501–504. Cambridge University Press, 1913.

[Zie98] Wieslaw Zielonka. Infinite Games on Finitely Coloured Graphs with Applications to Automata on Infinite Trees. *Theoretical Computer Science*, 200(1–2):135–183, 1998.

-Bouleī-
Paris 2008
9618

Pour Dieu