Philosophiae Doctor 1 (Nathanaël Fijalkow)

Counting and Randomising in Automata Theory

Compteurs et Aléas en Théorie des Automates

Liczenie i Losowość w Teorii Automatów

Structure of this Document

The purpose of this document is threefold:

- investigate the Finite-memory Determinacy for Boundedness Games,
- study the Value 1 Problem for Probabilistic Automata,
- provide its author with the title of Doctor in Computer Science.

Hence this document tells two stories; although independent, they are quite related, were developed in parallel, and influenced each other in various ways.

We first give a short Summary of the results presented in this document. The Introduction gives a historical perspective and motivates the two questions we consider.

The main part of this document is divided into two chapters of approximately equal length. The first chapter is "Finite-memory Determinacy for Boundedness Games" and the second chapter "The Value 1 Problem for Probabilistic Automata"; they can be read independently from one another.

What can be found in these two chapters? A rough classification of research in theoretical computer science distinguishes between "problem solving" and "theory building".

The first activity, "problem solving", consists in giving a definitive answer to a well defined and clearly identified question, either by constructing an algorithm, or by proving a mathematical property. To some extent, the first chapter can be classified as "problem solving".

On the other hand, the second activity, "theory building", does not focus on answers, but rather on questions: starting from an idea, a paradigm or a phenomenon it is trying to explain, it identifies mathematical objects to model and abstract it. It then draws from mathematical theories to reason about these objects, giving rise to mathematical foundations for defining problems and start answering them. The second chapter follows this path.

As a Conclusion, we argue that one contribution of this document is to bring together two questions a priori unrelated, and to transfer ideas and techniques from one to the other.

Summary

This thesis is a contribution to the study of quantitative models of automata, and more specifically of automata with counters and probabilistic automata. They have been independently studied for decades, leading to deep theoretical insights of practical value.

We investigate here two seemingly unrelated questions, the first about finite-memory determinacy for boundedness games, and the second about the value 1 problem for probabilistic automata. Some of the results are obtained by transferring techniques and ideas from one model to the other, revealing some similarities between them.

The first part of this document investigates finite-memory determinacy for boundedness games, which is motivated by, and belongs to, a research program launched ten years ago by Bojańczyk and Colcombet, aiming at understanding boundedness logics; the so-called MSO + $\mathbb{U}$ and cost-MSO. These two logics induce two related models of games with counters, which we call uniform and non-uniform boundedness games.

We first consider non-uniform boundedness games, which appear in the study of MSO + $\mathbb{U}$, and present results aiming at characterising the memory requirements of a given condition, first for special cases of boundedness conditions, and then for general conditions under topological assumptions.

We then consider uniform boundedness games, which appear in the study of cost-MSO. In 2010, Colcombet and Löding stated a conjecture about finite-memory determinacy for such games, that we call the LoCo conjecture. They proved that this conjecture implies the decidability of cost-MSO over infinite trees, which is the main open problem of this research program. We show that unfortunately the LoCo conjecture does not hold. On the positive side, we show a weaker statement: the LoCo conjecture holds for all thin tree arenas.

The second part of this document is about the value 1 problem for probabilistic automata. The starting point is the undecidability of this problem, which was proved in 2010 by Gimbert and Oualhadj. The aim of the results presented here is to understand to what extent is the value 1 problem undecidable, by constructing an algorithm that partially solves it and arguing that it is *in some sense* optimal.

The first step is the construction of the Markov Monoid algorithm, inspired by the notion of stabilisation monoids as introduced by Colcombet in the study of cost-MSO. We first prove that it is correct for a subclass of probabilistic automata that we define, called leaktight automata, relying on the notion of factorisation trees as developed by Imre Simon. We then show that all subclasses for which the value 1 problem was shown to be decidable are leaktight, implying that this algorithm is *so far* the best known algorithm.

The second step aims at better understanding the Markov Monoid algorithm, using a new framework that we introduce, called the prostochastic theory. The prostochastic theory is a topological approach for probabilistic automata inspired by the profinite theory as developed by Almeida, Pin, Weil and others for classical automata, and used by Toruńczyk for the study of $\mathrm{MSO} + \mathbb{U}$. In this context, the value 1 problem reformulates at an emptiness problem, providing theoretical foundations for reasoning about the Markov Monoid algorithm. Our main result is to give a characterisation of the Markov Monoid algorithm using the prostochastic theory. It says that the Markov Monoid algorithm captures exactly all polynomial behaviours; the undecidability result shows that combining polynomial and exponential behaviours leads to undecidability. The combination of these two results supports the claim that the Markov Monoid algorithm is optimal.

Introduction

This document presents a number of contributions to automata theory, which is the study of abstract machines, appearing in various fields such as formal language theory, complexity theory, compiler design, artificial intelligence, parsing and formal verification.

More specifically, we consider two questions, the first related to memory in boundedness games, and the second to probabilistic automata. In this introduction, we explain where do these questions come from. We start from the very beginning, and discuss in Section 1 how the fundamental concepts emerged: first the notion of regularity in automata theory, then games as a tool for understanding automata, and in parallel extensions to quantitative aspects. We then narrow down to our two topics of interests, and give an account on their developments over the last decades:

- In Section 2, we consider memory requirements in games and quantitative games. At the crossing of these two lies the LoCo conjecture, which is about finite-memory determinacy for boundedness games, a class of quantitative games.

  The first chapter of this document investigates further the LoCo conjecture.

- In Section 3, we consider the algorithmic properties of probabilistic automata. In the last fifty years of research, a wide range of quantitative problems have been investigated, and all proved undecidable. This sequence of negative results justifies considering problems of different nature: we discuss the recent developments on the value 1 problem, which is a qualitative (as opposed to quantitative) problem.

  The second chapter of this document investigates further the value 1 problem.

1 The origins

In the beginning was logics. A perfect match between automata and logics was discovered, leading to the notion of *regularity*. Automata theory expanded, and further investigations led to new theoretical tools; *games* is one of them. Among the various natural extensions of automata theory, adding *quantitative* aspects is a prominent one. We put a focus on one such quantitative aspect, namely *boundedness questions*. In this section, we discuss these three concepts: regularity, games, and quantitative extensions.

1.1 The Notion of Regularity in Automata Theory

Automata theory takes its roots in the early fifties, with two main ideas: *rational expressions*, introduced by Kleene [Kle56], and *non-determinism*, introduced by Rabin and Scott [RS59]. In both cases, the results are about two formalisms being expressively equivalent: rational expressions, deterministic automata and non-deterministic automata. Later, Büchi and Elgot (together [BE58], and separately [Büc60; Elg61]), and independently Trakhtenbrot [Tra62] added a logical perspective, showing that monadic second-order logic and automata define the same class of languages. A fourth equivalence, due to Schützenberger [Sch56], laid the foundations of the algebraic approach for automata theory, stating that the languages recognizable by monoids are exactly the regular languages.

Thus started automata theory, based on the "virtuous circle", *i.e.* the equivalence between different formalisms to define *regular languages over finite words*: automata, rational expressions, monoids and logic. Thanks to this diversity, automata theory has developed in various directions, for instance it has been extended to more complicated structures. This culminates with the works of Rabin [Rab69] defining *regular languages over infinite trees*, implying the decidability of monadic-second order logic.

Today, automata theory can be thought of as a toolbox for solving various problems. Most importantly, "automata theory is a tool to make logics effective" [VW08]. Indeed, the effective translations from logics to automata allow to use the latter to algorithmically solve problems on the former. A canonical example of this principle is the research area called *Model Checking*, a branch of automata theory interested in constructing powerful algorithms and specification languages for the verification of programs.

1.2 The Ubiquity of Games

A milestone in the development of automata theory is Church's problem [Chu57], also known as the circuit synthesis problem. It has been solved by Büchi and Landweber [BL69], following automata-theoretic ideas from McNaughton [McN66], but mostly by setting this problem in the context of games.

An instance of Church's problem is given by a specification, described for instance by a logical formula, and asks for the construction of a system satisfying this specification. The idea of Büchi and Landweber is to cast this as a game, in which two players have antagonistic goals. The game is played over a graph, called an arena, which represents the system; the first player, we call her Eve, represents the controller of the system, and her opponent, we call him Adam, represents the environment. The specification gives rise to a winning condition that Eve tries to ensure. Thus, the Church problem is equivalent to determining whether Eve has a winning strategy in the game, and to construct such a winning strategy.

In other words, an important conceptual contribution of Büchi and Landweber is to advocate the use of games as algorithmic back-ends to solve computational problems, such as Church's problem.

A second observation motivates the study of games for its applications to automata theory, this time not as an algorithmic back-end but as a theoretical tool. Indeed, studying the shape of winning strategies, in order to prove the existence of simple winning strategies, gives a deep combinatorial insight into automata-theoretic problems.

The first achievement in this direction is the long list of simplifications of the complex and celebrated Rabin's Theorem [Rab69], stating the decidability of monadic second-order logic over infinite trees.

The use of determinacy and positional determinacy for automata constructions was first noticed by Gurevich and Harrington [GH82]. Emerson and Jutla found a strong relation between the parity conditions and the modal μ-calculus, allowing to a give a simple proof of the positionality of parity games [EJ88; EJ91]. Muller and Schupp gave a conceptually different proof of this result, which in particular gives a new proof of the determinisation for automata over infinite words [MS87; MS95].

An important contribution of these works is to pinpoint the use of *positional determinacy of parity games* in the proof of Rabin's theorem. This result states the existence of positional winning strategies: if there exists a winning strategy, then there exists a very simple one, namely a positional winning strategy. We say that parity conditions are positionally determined; this type of results, called positionality results, or more generally finite-memory determinacy results, is central in many automata-theoretic constructions that came later, for instance for the equivalence between two-way and one-way automata [Var98], or for solving pushdown games [Wal01].

1.3 Quantitative Aspects, with a focus on Boundedness Questions

The central object in automata theory is *languages*, *i.e.* *qualitative* properties: a structure (a word, a tree) is either in the language, or not. In order to extend the toolbox to a *quantitative* setting, different attempts have been made, to give automata the ability – for instance – to count or to randomise.

A general and widely studied approach is given by weighted automata as defined by Schützenberger [Sch61]. In this framework fits a number of important classes of quantitative models of automata, such as probabilistic automata (introduced independently by Rabin [Rab63]) and distance automata.

The literature on quantitative extensions of automata theory is vast and still growing; We will now focus on one branch of this research area, specifically boundedness questions.

About ten years ago, Bojańczyk introduced the logic MSO + $\mathbb{U}$ [Boj04], to capture boundedness questions. A typical example of a boundedness question is: given a regular language $L \subseteq \{a, b\}^*$, does there exist a bound N such that all words from L contain at most N occurrences of a?

The motivations for studying boundedness questions gets back to the 80s, when Hashiguchi, and then later Leung, Simon and Kirsten solved the *star-height problem* by reducing it to boundedness questions [Has90; Sim94; Leu91; Kir05].

Over the last ten years, a lot of results have been obtained about the logic MSO + $\mathbb{U}$ and its close parent cost-MSO, introduced as part of the theory of regular cost functions by Colcombet [Col09; Col13b].

The results of this line of work come in two flavours: the first is reducing various problems to boundedness questions, and the second is obtaining decidability results for boundedness questions as formulated by variants of the logics MSO + $\mathbb{U}$ and cost-MSO.

For the first point, many problems have been reduced to boundedness questions. The first example is the star-height problem over words [Has90; Sim94; Leu91; Kir05] and over trees [CL08a]. A year later, Kirsten solved the finite substitution problem [Kir06]. When Bojańczyk introduced MSO + $\mathbb{U}$, one of his motivation was for solving the finite satisfiability of the modal μ-calculus with backward modalities. Recently, the boundedness question for fixed points of monadic formulae over finite and infinite words and trees have been cast as boundedness problems by Blumensath, Otto and Weyer [BOW14]. Last but not least, the most important problem that has been reduced is to a boundedness problem is to decide the Mostowski hierarchy for infinite trees [CL08b]. The decidability of this problem remains open, as the corresponding boundedness question, namely boundedness of B-parity automata over infinite trees, is not known to be decidable.

For the second point, we review some of the most important results. The first paper by Bojańczyk introduces MSO + $\mathbb{U}$ and solves a fragment over infinite trees, not closed under negations [Boj04]. Two years later, Bojańczyk and Colcombet show the decidability of a bigger fragment closed under negations, over infinite words [BC06]. From there, two paths emerge: the study of the weak variant of MSO + $\mathbb{U}$ led by Bojańczyk on one side, and the theory of regular cost functions and cost-MSO led by Colcombet on the other.

The logic weak MSO + $\mathbb{U}$ has been shown decidable over infinite words [Boj11], infinite trees [BT12], and even with path quantifiers [Boj14]. Quite recently, the non weak version had a tragic end, when Bojańczyk, Toruńczyk and Parys proved the undecidability of MSO + $\mathbb{U}$ over infinite words [BPT15].

The theory of regular cost functions, built around the logic cost-MSO, has been successfully developed over finite words [Col09; Col13b] and finite trees [CL10], yielding notions of regular expressions, automata, monoids and logics that all have the same expressive power, and that extend the standard notions. This led to decidability results for boundedness questions over finite words and trees.

2 Memory in Quantitative Games

The study of games in automata theory is motivated by two different objectives: the first is to use games as algorithmic back-ends, constructing powerful algorithms to determine the winner and synthesise winning strategies, the second is to use games as proof objects, proving the existence of positional or finite-memory winning strategies.

They are often related; in the first chapter of this document, we will focus on the second objective, namely finite-memory determinacy issues, and study the class of quantitative games given by boundedness games. We discuss on this section these two lines of work: first, results characterising memory requirements, and second, the study of quantitative games.

This leads us to the LoCo conjecture, which is the main motivation for the first chapter of this document.

2.1 Characterising Memory Requirements

As explained in the previous section, the first to notice the use of games for automata-theoretic constructions were Gurevich and Harrington [GH82], followed by Muller and Schupp [MS87; MS95], and by Emerson and Jutla [EJ88; EJ91].

This set of results motivated further investigations: what are the general techniques to prove positionality or finite-memory determinacy results, and can we characterise the memory requirements of a given winning condition?

A very elegant result by Dziembowski, Jurdziński and Walukiewicz characterised the memory requirements of Muller conditions [DJW97], relying on the structure of Zielonka trees [Zie98]. The literature on characterising memory requirements is very large; for instance, two PhD theses focussed on finding necessary and sufficient conditions for a condition to be half-positionally determined, first by Gimbert [Gim07], and later by Kopczyński [Kop09].

2.2 Quantitative Games

There are two ways to makes games quantitative:

- the first is to add quantitative features to the arenas, for instance randomised edges, inducing stochastic arenas,
- the second is to consider quantitative conditions, for which a play is not winning or losing, but is assigned a value, that one player tries to minimise or maximise.

For the first approach, we just mention one interesting example. Stochastic games were introduced by Shapley [Sha53]; Condon investigated the case of stochastic reachability games and proved several properties about them, in particular their positional determinacy [Con92]. Later, Zwick and Paterson considered mean-payoff and discounted games: the arenas are deterministic (they do not have randomised edges), but the conditions are quantitative, as they assign to a play the limit (or discounted limit) of the average weights along the play. Zwick and Paterson showed that both types of games are positionally determined, and used this result to reduce them to the stochastic games of Condon [ZP96].

This is an interesting example where one approach reduces to the other: considering quantitative conditions can be reduced to qualitative conditions over stochastic arenas.

The second approach, considering quantitative conditions, has been quite popular recently. We distinguish two aims: either the quantitative aspect is used to refine a qualitative specification, or the quantitative aspect is used to specify a quantitative behaviour, combined with an independent qualitative specification.

We give an example of the first aim. Linear temporal logic is a powerful formalism to express qualitative specifications; for instance, one can specify that a request is eventually granted. Three quantitative counterparts have been introduced to further specify the existence of a time bound between requests and their grants. The first are the finitary conditions introduced by Alur and Henzinger [AH98], then parametric linear temporal logic, defined by Alur, Etessami, La Torre and Peled [AELP01], and its fragment prompt linear temporal logic, introduced by Kupferman and Vardi [KPV09]. In these three approaches, the quantitative aspect aims at giving a numerical account of "how well" is the specification satisfied.

Games with finitary conditions have been studied by Henzinger, Chatterjee and Horn, leading to surprisingly effective algorithms [CHH09]. The PhD thesis of Zimmermann studies games with parametric linear temporal logic conditions [Zim12], both by proving finite-memory determinacy and by constructing algorithms to determine the winner.

There are several examples of conditions combining an independent qualitative aspect and a quantitative aspect in the literature. We mention for instance energy parity games [CD12] and mean-payoff parity games [CDGO14]. In these two examples, the parity condition serves as an abstraction for a qualitative specification, and either the mean-payoff or the energy condition serves an abstraction for a quantitative specification.

2.3 The LoCo Conjecture

The theory of regular cost functions has been successfully developed for finite and infinite words, and for finite trees, leading to decidability results for the logic cost-MSO over these structures.

However, extending this theory to infinite trees in order to prove the decidability of the logic cost-MSO seems to be much harder. One motivation for this extension is a result of Colcombet and Löding, which reduces the index problem of the *Mostowski hierarchy* to the decidability of cost-MSO over infinite trees [CL08b]. This problem is open for over forty years; the deterministic case was solved, and recently extended to the case of game automata [FMS13]. The approach through regular cost functions allowed to solve the special case of weak definability for Büchi languages [CKLV13].

Colcombet and Löding pointed out that the only missing point to obtain the decidability of cost-MSO over infinite trees is a finite-memory determinacy result for boundedness games. Boundedness games are quantitative games involving a finite set of counters; their conditions are conjunctions of a parity condition and a condition requiring that the counter values remain bounded along the play.

Colcombet and Löding conjectured that there exists a trade-off between the size of the memory and the bound achieved on the counters [Col13a], which we call the *LoCo conjecture*.

So far, this conjecture resisted both proofs and refutations, and the only non-trivial positive case known is due to Vanden Boom [Van11], which implied the decidability of the weak variant of cost-MSO over infinite trees, later generalised to quasi-weak cost-MSO in [BCKPV14]. Unfortunately, quasi-weak cost-MSO is strictly weaker than cost-MSO, and this leaves open the question whether cost-MSO is decidable.

The first chapter of this document, "Finite-memory Determinacy for Boundedness Games", investigates further the LoCo conjecture.

3 Probabilistic Automata

Probabilistic automata have been introduced by Rabin [Rab63], as a very simple randomised computation model. It has been broadly studied for both its theoretical and practical values; in particular, it is used in various fields such as computational biology, linguistics and image processing.

In this section, we are interested in the algorithmic properties of probabilistic automata: given a probabilistic automaton, what can be said about its behaviour, in an effective way?

We first discuss some algorithmic questions that appeared in the seminal paper of Rabin. These are *quantitative* problems: they ask for the behaviours of a given probabilistic automaton, comparing the acceptance probabilities to numerical thresholds. This set of questions led to several developments over the last forty years, which unfortunately are almost all undecidability results, supporting the claim that quantitative problems for probabilistic automata are undecidable.

Hence we take a different path and discuss a recent work of Gimbert and Oualhadj, who considered a *qualitative* problem, the value 1 problem. The second chapter of this document continues this line of work, aiming at a better understanding of the value 1 problem.

3.1 Algorithmic Properties for Probabilistic Automata

The first result about algorithmic properties of probabilistic automata precedes the introduction of probabilistic automata. Indeed, while studying weighted automata, Schützenberger proved that the equivalence between two weighted automata is decidable, provided the underlying semiring is a field [Sch61]. This applies to probabilistic automata, as the underlying semiring is the reals with addition and multiplication. Later, a different polynomial time algorithm was given by Tzeng [Tze92]. Recently, this problem has been further analysed, leading to very efficient randomised algorithms, with applications to software verification [KMOWW11; KMOWW12; KMOWW13].

When Rabin introduced probabilistic automata, he showed how they can be used to define languages: a probabilistic automaton defines the language consisting of the set of words accepted with probability at least one half. The threshold one half is not special, any rational threshold between 0 and 1 leads to an equivalent notion of probabilistic languages. This motivated the first algorithmic question:

> (1) can we decide if a probabilistic language is empty, *i.e.* if there exists a word accepted with probability at least one half?

Rabin first observed that probabilistic languages strictly subsume regular languages, and gave a sufficient condition for a probabilistic automaton to define a regular language, using the notion of isolated thresholds.

The threshold one half is said isolated if there exists an interval around it such that for all words, their probability to be accepted lie outside this interval. In other words, Rabin showed that if no complicated convergence phenomenon occurs around the threshold one half, then the language is regular. This motivated the second and third algorithmic questions:

(2) can we decide if a probabilistic automaton defines a regular language?

(3) can we decide if for a probabilistic automaton, the threshold one half is isolated, *i.e.* if there exists a sequence of words accepted with probability arbitrarily close to one half?

3.2 Undecidability of Quantitative Problems

The first question was answered by Paz: the emptiness problem for probabilistic languages is undecidable [Paz71].

This undecidability result says that asking to precisely compare to a threshold is too much to ask. A probabilistic automaton may exhibit complicated behaviours that happen very close to the threshold, and this implies the undecidability.

Bertoni answered the second question, using a similar reduction: the regularity problem is undecidable [Ber74a].

The third question asks whether there is a convergence phenomenon towards the threshold, without trying to determine whether it goes beyond the threshold. Unfortunately, Bertoni showed that the isolation problem is also undecidable [Ber74b; BMT77], answering the third question. This means that not only determining if a word exceeds the threshold is undecidable, but also determining whether it is approached at all is also undecidable.

A simpler question is the following bounded-error emptiness problem, which asks for the construction of an algorithm with the following behaviour; given a probabilistic automaton as input:

- if there exists a word accepted with probability at least two thirds, then accept,
- if all words are accepted with probability at most one third, then reject,
- in any other case, any answer is fine, including no answer at all.

In this problem, there is no convergence phenomena involved: what happens between the thresholds one third and two thirds is not relevant. Condon and Lipton gave the *coup de grâce*: there exists no algorithm solving the bounded-error emptiness problem [CL89].

We comment on the term *quantitative problem*: it refers here to the idea that the emptiness problem, the isolation problem and the bounded-error emptiness problem all involve numerical thresholds, like one half, one third or two thirds, and comparisons with these thresholds. A priori, an algorithm solving a quantitative problem needs to deal with numerical values, and maybe approximations of them, which may be intrinsically hard.

The bounded-error emptiness seems to be the “easiest” quantitative problem, and its undecidability hints at the sad conclusion that all quantitative problems for probabilistic automata are undecidable.

On the opposite, a *qualitative problem* does not involve any numerical thresholds. Following the above intuition, qualitative problems should be easier than quantitative problems.

3.3 The Value 1 Problem

Gimbert and Oualhadj considered in 2010 the following qualitative problem, which they called the value 1 problem: given a probabilistic automaton, does there exist a sequence of words accepted with probability arbitrarily close to 1? (Similar problems have been considered for stochastic games, see for instance [AH00], where it is called “limit winning”.) Their first contribution was to prove that this problem is undecidable [GO10].

There are several reasons that make this problem interesting and worth further investigations.

First of all, it is very simple, and asks about a convergence phenomenon in a qualitative way: it does not involve comparisons with any precise threshold other than 1. As such, it is a good stepping stone to construct partial algorithms to determine properties of probabilistic automata. Indeed, the undecidability proof involves probabilistic automata with a complex structure, leaving some hope that these behaviours are rare and that it is possible to construct algorithms that solves the value 1 problem in most cases.

As it turned out later, the value 1 problem appeared in a different context, as the emptiness of probabilistic Büchi automata with positive semantics [BBG12]. The two problems are Turing-equivalent [CSV13].

Besides proving that the value 1 problem is undecidable, Gimbert and Oualhadj also constructed a subclass of probabilistic automata, called $\sharp$-acyclic, and showed that the value 1 problem is decidable when restricted to this subclass.

This led to further developments, and in particular to the second chapter of this document, “The Value 1 Problem for Probabilistic Automata”. This question has also been considered for more expressive models, such as probabilistic timed automata [BBG14].

Last but not least, the value 1 problem can be cast as an (un)boundedness question for probabilistic automata; this informal observation is the missing link between the two chapters of this document.

1 Finite-memory Determinacy for Boundedness Games

Contents

This first chapter deals with boundedness games, which are games with counters equipped with conditions requiring that the values of the counters remain bounded. The problem we tackle here is the finite-memory determinacy for such games:

"Does there exist finite-memory winning strategies for boundedness games?"

This question comes in several variants, depending on the conditions and the structural properties of the underlying arenas. In 2010, Colcombet and Löding stated a conjecture, that we call the LoCo conjecture, asserting the existence of finite-memory strategies in uniform boundedness games, and proved that this conjecture implies the decidability of cost-MSO over infinite trees. This conjecture has been the main motivation for all the results presented in this chapter.

In Section 1, we define boundedness games, which can be uniform and non-uniform. The non-uniform variant originates from the study of MSO + $\mathbb{U}$, and the uniform variant from the study of cost-MSO. We discuss further the relation between the two definitions: we show that for games played over pushdown arenas, both definitions are equivalent, a result obtained in collaboration with Krishnendu Chatterjee in [CF13].

We define finite-memory strategies in Section 2, and state finite-memory determinacy questions for boundedness games. We present in this section two sets of results about these questions.

In the first set of results, we take as a starting point the finitary parity games studied by Chatterjee, Henzinger and Horn [CHH09], which are a special case of non-uniform boundedness games. We extend their results in two directions: first by considering the cost-parity games, which generalise at the same time parity games and finitary parity games, in collaboration with Martin Zimmermann [FZ12; FZ14], and second by considering finitary parity games played over infinite arenas, in collaboration with Krishnendu Chatterjee in [CF13].

The second set of results is a characterisation of the memory requirements for topologically closed conditions, obtained in collaboration with Thomas Colcombet and Florian Horn [CFH14]. This allows to obtain tight memory bounds for several conditions, and in particular applies to a special case of uniform boundedness conditions.

We state the LoCo conjecture in Section 3. We show that it implies the equivalence between alternating and non-deterministic B-parity automata, as introduced in the theory of regular cost functions by Colcombet [Col09; Col13b]. A closer analysis at how the LoCo conjecture is used to prove this result allows to state a refined LoCo conjecture taking into account the structural properties of the arenas.

In Section 4, we give a state of the art on this conjecture and show new examples. We then explain how the slicing technique developed by Vanden Boom in [Van11] to prove a special case of the LoCo conjecture allowed to obtain a positionality for stochastic games, which was the main contribution of [FPS13], in collaboration with Sophie Pinchinat and Olivier Serre.

In Section 5, we show that the LoCo conjecture does not hold, by constructing a counter example. The counter example is defined over a sequence of finite arenas, and is extended to show that non-uniform boundedness games are not finite-memory determined. On the positive side, we show in Section 6 that the LoCo conjecture holds for games played over thin tree arenas. The proof involves structural decompositions of thin tree and word arenas.

Both the counter example and the case of thin tree arenas are joint work with Florian Horn, Denis Kuperberg and Michał Skrzypczak [FHKS15].

We conclude in Section 7.

1 Boundedness Games

This section gives the definitions used throughout this chapter. We define games in Subsection 1.1, and boundedness conditions in Subsection 1.2.

As we will see, a fine point in this definition is whether the bound on these counters is *uniform* over all plays, as in the study of cost-MSO, or *non-uniform*, as it is the case for MSO + $\mathbb{U}$. We discuss this further in Subsection 1.3.

We then report on our first contribution in Subsection 1.4, which an equivalence result between the uniform and non-uniform variants for games played over pushdown arenas, obtained in collaboration with Krishnendu Chatterjee and published in [CF13].

1.1 Games

The model of games we consider is closely connected to the study of automata. Most of our definitions are standard, and taken from the book [GTW02].

The games are played by two players, Eve and Adam, over potentially infinite graphs called *arenas*.

Definition 1 (Arena)

An arena $\mathcal{G}$ is given by a directed graph (V, E) whose vertex set is divided into vertices controlled by Eve (V_E) and vertices controlled by Adam (V_A), and a colouring function $c : E \to A$, where A is a finite alphabet.

The colouring function $c : E \to A$ will be used by the winning condition. Note that sometimes, the colouring function is $c : V \to A$. When the arena is given without the colouring function, we talk about the underlying graph.

In Figure 1 we present a finite arena. The vertices controlled by Eve are drawn as circles, the vertices controlled by Adam are drawn as squares.

For the following definitions, fix an arena $\mathcal{G}$.

A token is initially placed on a given initial vertex v_0, and the player who controls this vertex pushes the token along an edge, reaching a new vertex; the player who controls this new vertex takes over, and this interaction goes on forever, describing an infinite path called a *play*. We assume that the arenas contain no dead ends: for every vertex $v \in V$, there exists at least one edge $(v, v') \in E$.

Definition 2 (Play)

A play is a finite or infinite path, seen as a sequence of edges.

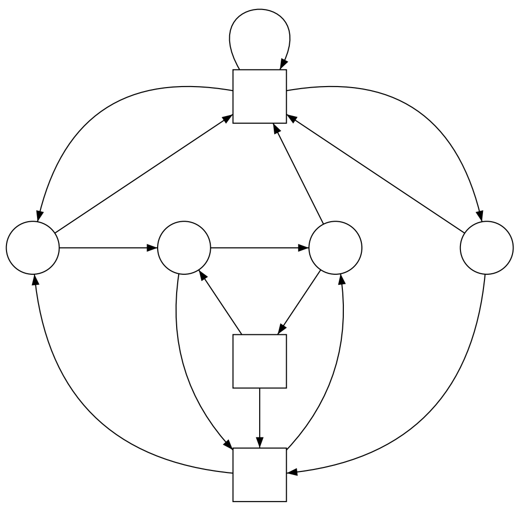


Figure 1: A finite arena (without any colouring function).

Plays are typically denoted π. Note that plays are sequences of edges, but sometimes it is technically convenient to consider only the induced sequences of vertices. As a convention, a play denoted $\pi = e_0 \cdot e_1 \cdots$ is a sequence of edges, and a play denoted $\pi = v_0 \cdot v_1 \cdots$ is a sequence of vertices.

Definition 3 (Strategy)

A strategy for Eve is a mapping $\sigma : E^* \cdot V_E \to E$.

A strategy takes as input the history played so far and the current vertex, and outputs the next edge. We say that a play $\pi = e_0 \cdot e_1 \ldots$ is consistent with σ if $e_{n+1} = \sigma(e_0 \cdots e_n \cdot v_n)$ for every n with $v_n \in V_E$.

From now on, the definitions that we give of winning conditions and games are *qualitative*, *i.e.* a play is either won or lost by Eve. In the next subsection, we will define *quantitative* winning conditions and games.

Informally speaking, a winning condition (or simply: condition) is a set of plays. The plays in the condition are called the winning plays for Eve, the other plays are winning for Adam. A strategy is winning for a condition, or ensures this condition, if all plays consistent with the strategy belong to the condition. The formal definition is slightly more complicated, in order to reason about conditions abstractly.

Definition 4 (Winning condition)

A condition over an alphabet A is a subset $W \subseteq A^\omega$.

Given an arena $\mathcal{G}$ and a condition W, we define the following set of plays:

$$\{\pi = e_0 \cdot e_1 \cdots \mid c(e_0) \cdot c(e_1) \cdots \in W\},$$

to be the set of winning plays for Eve.

Although conditions are not formally sets of plays, we will often abuse notations and say that a play $\pi = e_0 \cdot e_1 \cdots$ satisfies a condition $W \subseteq A^\omega$, meaning that $c(e_0) \cdot c(e_1) \cdots \in W$.

Definition 5 (Game)

A game is given by an arena $\mathcal{G}$ and a condition W. Such a game is denoted $(\mathcal{G}, W)$.

For a game $(\mathcal{G}, W)$, we denote $\text{Win}_E(\mathcal{G}, W)$ the winning region of Eve, *i.e.* the set of vertices from which Eve has a winning strategy. When $\mathcal{G}$ is clear from the context, we write $\text{Win}_E(W)$.

All these definitions can be transposed for Adam. A strategy for Adam is usually denoted τ, and $\text{Win}_A(\mathcal{G}, W)$ is the winning region of Adam.

Note that $\text{Win}_E(\mathcal{G}, W) \cap \text{Win}_A(\mathcal{G}, W) = \emptyset$. If $\text{Win}_E(\mathcal{G}, W) \cup \text{Win}_A(\mathcal{G}, W) = V$, we say that the game is determined: from any vertex, either Eve has a winning strategy, or Adam has a winning strategy. All the games we will consider in this document are determined, as we will only consider Borel conditions [Mar75]. When stating results about general conditions, we will always (implicitly) assume that the condition is Borel, to use the fact that the underlying games are determined.

We now define the classical ω-regular conditions.

Definition 6 (ω-regular condition)

- The parity condition with $d+1$ colors is defined over the alphabet $A = \{0, \ldots, d\}$, by:
$$\text{Parity} = \{c_0 \cdot c_1 \cdot c_2 \cdots \mid \max(\inf\{c_i \mid i \in \mathbb{N}\}) \text{ is even}\},$$
in words, it requires that the maximum color seen infinitely often is even.

- The Büchi condition is the special case of the parity condition over the alphabet $A = \{1, 2\}$, denoted Büchi.

- The CoBüchi condition is the special case of the parity condition over the alphabet $A = \{0, 1\}$, denoted CoBüchi.

- The reachability condition is defined over the alphabet $A = \{0, 1\}$, by:
$$\text{Reach} = \{c_0 \cdot c_1 \cdot c_2 \cdots \mid \exists i \in \mathbb{N}, c_i = 1\}.$$

- The safety condition is defined over the alphabet $A = \{0, 1\}$, by:
$$\text{Safe} = \{c_0 \cdot c_1 \cdot c_2 \cdots \mid \forall i \in \mathbb{N}, c_i = 0\}.$$

As a convention, the colouring function of the parity condition is denoted $\Omega : V \to \{0, \ldots, d\}$. We will sometimes use partial colouring functions for the parity condition; what matters is that every infinite play contains infinitely many coloured vertices, which will always be the case.

For a Büchi condition, we define $F = \{v \in V \mid c(v) = 2\}$, and call F the set of Büchi vertices; the Büchi condition requires to see F infinitely many times. Similarly, for a CoBüchi condition, we define $F = \{v \in V \mid c(v) = 1\}$, and call F the set of CoBüchi vertices; the Büchi condition requires to see F finitely many times.

For both reachability and safety conditions, we define $F = \{v \in V \mid c(v) = 1\}$. The reachability condition requires to see F at least once, and the safety condition never to see F at all.

1.2 Boundedness Games

We now introduce counters and the boundedness conditions, which are *quantitative conditions*, *i.e.* functions $f : A^\omega \to \mathbb{N} \cup \{\infty\}$.

Definition 7 (B and B-parity conditions)

The quantitative B condition with k counters is defined over the alphabet $A = \{\varepsilon, i, r\}^k$. The counters take integer values, initialized at 0. The actions are:

- increment by 1, denoted i,
- reset to 0, denoted r,
- leave unchanged, denoted ε.

The function $B : A^\omega \to \mathbb{N} \cup \{\infty\}$ assigns to a sequence s the supremum of the values of all counters along the sequence.

The quantitative B-parity condition with $d + 1$ colors and k counters is defined over the alphabet $A = \{0, \ldots, d\} \times \{\varepsilon, i, r\}^k$. The function $B \cap \text{Parity} : A^\omega \to \mathbb{N} \cup \{\infty\}$ assigns to a sequence $s = (c_0, a_0) \cdot (c_1, a_1) \cdot (c_2, a_2) \cdots$:

$$\begin{cases} \infty & \text{if } c_1 \cdot c_2 \cdots \notin \text{Parity} \\ B(a_0 \cdot a_1 \cdot a_2 \cdots) & \text{otherwise.} \end{cases}$$

A quantitative condition $f : A^\omega \to \mathbb{N} \cup \{\infty\}$ gives rise to two conditions:

- the *uniform* condition $W(N) = \{s \in A^\omega \mid f(s) \leqslant N\}$,
- the *non-uniform* condition $W = \{s \in A^\omega \mid f(s) < \infty\}$.

We say that the condition $W(N)$ is *uniform*, because a strategy ensuring this condition ensures the same bound N for all plays. On the opposite, the condition W is *non-uniform*, because a strategy ensuring this condition ensures a different bound for each play.

The two conditions induced by the quantitative B-parity conditions are denoted $B(N) \cap \text{Parity}$ (uniform) and $B \cap \text{Parity}$ (non-uniform).

We consider particular cases of B-parity conditions, as for instance B-Büchi conditions or B-CoBüchi conditions. In the special case of B-reachability conditions, denoted B Until Reach, we assume that the game stops when it reaches F.

A colouring function for the B-parity condition is usually given by two functions $\Omega : V \to \{0, \dots, d\}$ and $\mathbf{c} : E \to \{\varepsilon, i, r\}^k$.

Definition 8 (Boundedness game)

The B-parity games are called *boundedness games*.

Boundedness games have been given a lot of different names: the uniform variants have been called cost-games, cost-parity games, B-parity games or B games, and the non-uniform variants ωB-games.

1.3 Uniform and Non-Uniform

We explain in this subsection the distinction between uniform and non-uniform bounds, which comes from the study of MSO + $\mathbb{U}$ and cost-MSO. We here define both logics over infinite trees, but the definition also applies to infinite words, and to finite or infinite trees for cost-MSO.

We fix an alphabet A. An A-labelled (complete binary) *tree* is a function $t : \{0,1\}^* \to A$. The elements of t are called *nodes*: for $n \in \{0,1\}^*$, the node $n \cdot \ell$ is a *child* of n if $\ell \in \{0,1\}$, and a *descendant* of n if $\ell \in \{0,1\}^*$.

The logic MSO + $\mathbb{U}$ was introduced by Bojańczyk in 2004 [Boj04]. It is an extension of MSO, with a $\mathbb{B}$ quantifier (we will define $\mathbb{B}$ rather than $\mathbb{U}$, which is equivalent as they are dual); we assume the reader to be familiar with MSO, see for instance [GTW02].

Consider $\phi(X)$ an MSO formula using a free second-order variable X, ranging over sets of nodes. We define the syntactic construction $\mathbb{B}(X)\ \phi(X)$, whose semantics is as follows:

$$\mathbb{B}(X)\ \phi(X) \quad = \quad \exists N \in \mathbb{N},\ \forall X, \phi(X) \implies |X| \leqslant N\ .$$

In words, the formula $\mathbb{B}(X)\ \phi(X)$ holds for a tree t if there exists a bound $N \in \mathbb{N}$ such that if t satisfies $\phi(X)$, then X has size at most N.

A typical example is the formula

$$\mathbb{B}(X) \left\{ \begin{array}{cc} & \forall x, y, z, x < y < z \wedge x \in X \wedge z \in X \implies y \in X \\ \wedge & \forall x, a(x). \end{array} \right.$$

Here the symbol $<$ is interpreted as the descendant relation in the tree, so $x < y$ holds if y is a descendant of x. This formula states that there exists a bound N, such that all connected sets X that contain only a's have size at most N.

The semantics is Boolean: a tree t either satisfies or does not satisfy a given MSO + $\mathbb{U}$ formula. Hence the following decision problem:

Problem 1 (Satisfiability of MSO + $\mathbb{U}$)

Given an MSO + $\mathbb{U}$ formula ϕ, does there exist a tree t that satisfies ϕ?

Note that a formula in MSO+$\mathbb{U}$ can using several nestings of this $\mathbb{B}$ quantifier, together with first- and second-order quantifications. Consider for instance a formula of the form $\forall X,\ \mathbb{B}(Y)\ \phi(X,Y)$. Unravelling the semantics, we obtain:

$$\forall X,\ \exists N \in \mathbb{N},\ \forall Y, \phi(X,Y) \implies |Y| \leqslant N\ .$$

Here, if a tree t satisfies this formula, the bound N (a priori) depends on X: for every X, there exists a different N. This *non-uniform* behaviour makes the problem very hard.

Bojańczyk conjectured in 2004 that the satisfiability problem for MSO + $\mathbb{U}$ was decidable, even over infinite trees [Boj04]. Quite recently (February 2015), this conjecture has been disproved by Bojańczyk, Parys and Toruńczyk:

Theorem 1 (Undecidability of MSO + $\mathbb{U}$ over infinite words [BPT15])

The satisfiability problem for MSO + $\mathbb{U}$ over infinite words is undecidable.

The logic cost-MSO was introduced by Colcombet in 2009 [Col09]. It also extends MSO, with the syntactic construction $|X| \leqslant N$.

The atomic formula $|X| \leqslant N$ can appear in different places in a cost-MSO formula, but always positively, *i.e.* under the scope of an even number of negations, and with the same N.

The semantics of a cost-MSO formula assigns a value in $\mathbb{N} \cup \{\infty\}$ to a tree, which is the smallest $N \in \mathbb{N}$ such that t satisfies ϕ where the variable N is substituted by the value N. So instead of defining a language, as MSO and MSO + $\mathbb{U}$, a cost-MSO formula defines a function from the set of trees to $\mathbb{N} \cup \{\infty\}$.

The main decision problem for cost-MSO is the boundedness problem:

Problem 2 (Boundedness of cost-MSO)

Given a cost-MSO formula ϕ, does there exist $N \in \mathbb{N}$ such that all trees t satisfy ϕ with the value N?

Unlike MSO + $\mathbb{U}$, the same bound N applies to all trees. This *uniform* behaviour makes the problem easier. In particular, the undecidability result mentioned above does not imply anything for cost-MSO.

1.4 From Non-Uniform to Uniform for Pushdown Arenas

This subsection reports on a result obtained in collaboration with Krishnendu Chatterjee, published in [CF13]. The aim is to understand when are the uniform and non-uniform B-parity conditions equivalent.

We have:

$$B(0) \cap \text{Parity}\ \subseteq\ B(1) \cap \text{Parity}\ \subseteq\ B(2) \cap \text{Parity}\ \subseteq \cdots \subseteq\ B \cap \text{Parity}.$$

Consider an arena $\mathcal{G}$ with the colouring function given as two functions $\Omega : V \to \{0, \ldots, d\}$ and $\mathbf{c} : E \to \{\varepsilon, i, r\}^k$. Denoting $\text{Win}_E(N)$ for $\text{Win}_E(\mathcal{G}, B(N) \cap \text{Parity})$ and Win_E for

$\text{Win}_E(\mathcal{G}, B) \cap \text{Parity})$, it follows that:

$$\text{Win}_E(0) \subseteq \text{Win}_E(1) \subseteq \text{Win}_E(2) \subseteq \cdots \subseteq \text{Win}_E.$$

Hence the question: does there exist N such that $\text{Win}_E(N) = \text{Win}_E$, *i.e.* does this hierarchy collapse?

We denote $\lim B(N)$ the condition requiring that a suffix of the play satisfies the condition $B(N)$. The complement of $\lim B(N)$ is closed under suffixes, which implies interesting properties for strategies ensuring this condition.

Lemma 1 (Winning regions of suffix-closed conditions are traps)

Consider a game $(\mathcal{G}, W)$ where W is closed under suffixes: for π a finite play and π' an infinite play, if $\pi \cdot \pi' \in W$, then $\pi' \in W$.

Let σ be a winning strategy, then all plays consistent with σ remain in $\text{Win}_E(\mathcal{G}, W)$.

Proof. Assume towards contradiction that there exists a strategy σ ensuring W, and a play consistent with this strategy that at some point leaves $\text{Win}_E(\mathcal{G}, W)$, and reaches a vertex $v \in V \backslash \text{Win}_E(\mathcal{G}, W) = \text{Win}_A(\mathcal{G}, W)$. Denote by π this finite play. By definition, there exists a strategy τ of Adam that from v ensures that W is not satisfied. Consider the play from there consistent with both σ and τ, denote it π'. Since σ is winning, we have $\pi \cdot \pi' \in W$, and since τ ensures the complement of W, we have $\pi' \notin W$, contradicting that W is closed under suffixes. ∎

We first consider the case of finite arenas.

Theorem 2 (Collapse over finite arenas)

For all boundedness games played over a finite arena, for all initial vertices, the following are equivalent:

- $\exists \sigma$ strategy for Eve, $\forall \pi$ plays, $\exists N \in \mathbb{N}$, $\pi \in B(N) \cap \text{Parity}$,
- $\exists \sigma$ strategy for Eve, $\exists N \in \mathbb{N}$, $\forall \pi$ plays, $\pi \in \lim B(N) \cap \text{Parity}$.

Proof. We argue that the following two properties hold:

1. there exists N such that $\text{Win}_E(N) = \text{Win}_E(N+1) = \cdots$,
2. for such N, we have $\text{Win}_A(N) \subseteq \text{Win}_A$, hence $\text{Win}_E = \text{Win}_E(N)$.

The first property follows from the observation that $(\text{Win}_E(N))_{n \in \mathbb{N}}$ is a non-decreasing family of *finite* sets.

We now argue that the second property holds. Observe that $\lim B(N) \cap \text{Parity}$ is closed under prefixes, or equivalently its complement is closed under suffixes. From $\text{Win}_A(N)$, Adam has a strategy τ_N that ensures that either the parity condition is not satisfied, or that some counter value exceeds N. It follows from Lemma 1 that all plays consistent with this strategy remain in $\text{Win}_A(N)$.

We construct a strategy from $\text{Win}_A(N)$ that ensures the complement of $B \cap \text{Parity}$. The argument above yields a sequence of strategies $(\tau_{N'})_{N' \geqslant N}$. The strategy simulates those strategies in turn, switching from the strategy $\tau_{N'}$ to the strategy $\tau_{N'+1}$ if some counter value exceeds N'. Note that since each strategy $\tau_{N'}$ ensures to remain in $\text{Win}_A(N') = \text{Win}_A(N)$, the general strategy can indeed switch from one strategy to the next. It follows that $\text{Win}_A(N) \subseteq \text{Win}_A$, implying $\text{Win}_E = \text{Win}_E(N)$. ∎

***Remark* 1.** *The above proof does not give a bound on N; the sequence $(\text{Win}_E(N))_{N \in \mathbb{N}}$ is ultimately constant, but the index from which it is constant can be a priori arbitrarily large.*

Indeed, it is not true that $\text{Win}_E(N) = \text{Win}_E(N+1)$ implies that $\text{Win}_E(N+1) = \text{Win}_E(N+2)$, so this sequence might be constant (say, equal to the empty set) for N up to a very large value, and then contain all vertices for N larger than this value.

The main contribution of [CF13] is to extend this collapse result to pushdown games. Such games are played over arenas that are induced by a pushdown process; they are in general infinite.

A pushdown process is a finite-state machine which features a stack: it is given by a finite set of control states Q, a stack alphabet Γ and a transition relation Δ. There is a special stack symbol denoted $\perp$ which does not belong to Γ; we denote by $\Gamma_\perp$ the alphabet $\Gamma \cup \{\perp\}$. A configuration is a pair $(q, u\perp)$ (the top stack symbol is the leftmost symbol of u). There are three kinds of transitions in Δ:

- $(p, a, \text{push}(b), q)$: allowed if the top stack element is $a \in \Gamma_\perp$, the symbol $b \in \Gamma$ is pushed onto the stack.
- $(p, \text{pop}(a), q)$: allowed if the top stack element is $a \in \Gamma$, the top stack symbol a is popped from the stack.
- (p, a, skip, q): allowed if the top stack element is $a \in \Gamma_\perp$, the stack remains unchanged.

The symbol $\perp$ is never pushed onto, nor popped from the stack.

Definition 9 (Pushdown graph)

A pushdown process and two subsets Q_E, Q_A such that $Q = Q_E \uplus Q_A$ induce a pushdown graph, whose set of vertices is $Q \times \Gamma^* \perp$, and set of edges is E induced by the transition relation Δ.

For instance, if $(p, a, \text{push}(b), q) \in \Delta$, then $((p, aw\perp), (q, baw\perp)) \in E$, for all words w in Γ^*.

A colouring function for a pushdown arena is given by $c : \Delta \to A$, and naturally extended to $c : E \to A$.

Following our convention, a colouring function for the B-parity condition over a pushdown graph is given by two functions $\Omega : Q \to \{0, \ldots, d\}$ and $\mathbf{c} : \Delta \to \{\varepsilon, i, r\}^k$.

We give an example of a game played over a pushdown arena in Figure 2.

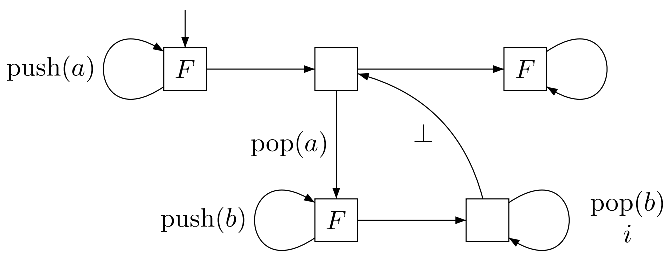


Figure 2: A B-Büchi game over a pushdown arena where Eve wins the non-uniform condition but no uniform ones.

Example 1 (Stack as credit in a boundedness game played over a pushdown arena)

Figure 2 presents a B-Büchi game played over a pushdown arena where Eve wins for the non-uniform condition B, but for no uniform condition $B(N)$.

Let us first look at the two bottom states: in the left-hand state at the bottom, Adam can push as many b's as he wishes, and moves the token to the state to its right, where all those b's are popped one at a time, incrementing the counter each time. In other words, each visit of the two bottom states allows Adam to announce a number N and to increment the counter by N.

We now look at the states on the top line: the initial state is the leftmost one, where Adam can push an arbitrary number of a's. We see those a's as credits: from the central state, Adam can use one credit (*i.e* pop an a) to pay a visit to the two bottom states. When he runs out of credit, which will eventually happen, he moves the token to the rightmost state, where nothing happens anymore.

The following theorem is the main contribution of [CF13]. It was called the "forgetful property", following the intuition that even if a configuration carries an unbounded amount of information (since the stack may be arbitrarily large), this information cannot be forever transmitted along a play. Indeed, to increase the counter values significantly, Adam has to use the stack, consuming or forgetting its original information.

Theorem 3 (Collapse over pushdown arenas)

For all boundedness games played over a pushdown arena, for all initial configurations, the following are equivalent:

- $\exists \sigma$ strategy for Eve, $\forall \pi$ plays, $\exists N \in \mathbb{N}$, $\pi \in B(N) \cap \text{Parity}$,
- $\exists \sigma$ strategy for Eve, $\exists N \in \mathbb{N}$, $\forall \pi$ plays, $\pi \in \lim B(N) \cap \text{Parity}$.

This result is in some sense maximal: it is easy to construct an example of a game played over a higher order pushdown system of level 2, where the above equivalence does not hold.

The proof follows the same lines as for the case of finite arenas, except that for the case of pushdown arenas, the sequence $(\text{Win}_E(N))_{N \in \mathbb{N}}$ now consists of infinite sets of configurations. To prove that this sequence is ultimately constant, we show that these sets are regular of bounded size, relying on general results for pushdown games due to [Ser03; Ser06].

We will use alternating $\mathcal{P}$-automata to recognize sets of configurations: an alternating $\mathcal{P}$-automaton $\mathcal{B}$ for a given pushdown process is a classical alternating automaton over finite words: it has a finite set of control states S, a transition function $\delta : S \times \Gamma \to \mathcal{B}^+(S)$ (the notation $\mathcal{B}^+(S)$ denotes the set of positive boolean formulae over S) and a subset F of S of final states. We assume that the set of states S contains Q. A configuration $(q, u\bot)$ is accepted by $\mathcal{B}$ if it is accepted with $q \in Q \subseteq S$ as initial state and the classical alternating semantics. A set of configurations is called *regular* if it is accepted by an alternating $\mathcal{P}$-automaton. The size of an alternating $\mathcal{P}$-automaton is its number of states.

Theorem 4 (Regularity of winning regions [Ser06])

For all prefix-independent winning conditions $W \subseteq A$ for all games played over a pushdown arena, the set $\text{Win}_E(W)$ is a regular set of configurations recognized by an alternating $\mathcal{P}$-automaton of size $|Q|$.

Relying on Theorem 4, it is easy to see that the sequence $(\text{Win}_E(N))_{N\in\mathbb{N}}$ is ultimately constant, so the proof of Theorem 2 can be repeated to prove Theorem 3.

As a corollary of the forgetful property, we obtain the following decidability result, relying on decidability results obtained in the theory of regular cost functions [BCKPV14].

Corollary 1 (Decidability of boundedness games played over pushdown arenas)

Determining the winner in a non-uniform boundedness game played over a pushdown arena is decidable.

Indeed, it is easy and classical to reduce the problem of determining the winner in a game played over a pushdown arena to the membership problem of a given regular tree by a corresponding two-way automaton [KV00]. This problem was shown decidable for two-way cost-automata in [BCKPV14].

2 Finite-Memory Determinacy

In this section, we define finite-memory strategies, and investigate finite-memory determinacy questions: roughly speaking, is it true that if Eve has a winning strategy, then she has a finite-memory winning strategy?

In Subsection 2.1 we introduce finite- and bounded-memory determinacy, and state four questions about finite-memory determinacy for boundedness games.

We then present some results we obtained in this direction.

In Subsection 2.2, we discuss the results obtained in collaboration with Krishnendu Chatterjee, published in [CF13], and with Martin Zimmermann, published in [FZ12; FZ14]. These results are about finite-memory determinacy for special classes of non-uniform boundedness games, and specifically extensions of finitary conditions. For the sake of keeping this document to a reasonable length, we do not detail the proofs, which can be found in the corresponding papers.

In Subsection 2.3, we report on results obtained in collaboration with Thomas Colcombet and Florian Horn, published in [CFH14]. In this work, we focus on topologically closed conditions, and give an exact characterisation of the memory requirements.

2.1 Finite-Memory Strategies

Definition 10 (Memory structure)

A memory structure for the arena $\mathcal{G}$ consists of a set M of memory states, an initial memory state $m_0 \in M$ and an update function $\mu : M \times E \to M$.

The update function takes as input the current memory state and the chosen edge to compute the next memory state, in a deterministic way. It can be extended to a function $\mu : E^* \cdot V \to M$ by defining $\mu^*(v) = m_0$ and $\mu^*(\pi \cdot (v,v')) = \mu(\mu^*(\pi \cdot v),(v,v'))$.

A strategy using the memory structure $\mathcal{M}$ is induced by a next-move function $\sigma : V_E \times M \to E$, by $\sigma(\pi \cdot v) = \sigma(v, \mu^*(\pi \cdot v))$. Note that we denote both the next-move function and the induced strategy σ. The size of such a strategy is the cardinal of M and denoted $\mathrm{mem}(\sigma)$. A strategy is *memoryless*, or *positional* if M is a singleton: it only depends on the current vertex. Note that a memoryless strategy can be described as a function $\sigma : V_E \to E$.

An arena $\mathcal{G}$ and a memory structure $\mathcal{M}$ for $\mathcal{G}$ induce the expanded graph $\mathcal{G} \times \mathcal{M}$ where the memory state is computed online: the vertex set is $V \times M$, the edge set is $E \times \mu$, defined by: $((v,m),(v',m')) \in E'$ if $(v,v') \in E$ and $\mu(m,(v,v')) = m'$. There is a natural one-to-one correspondence between memoryless strategies in $\mathcal{G} \times \mathcal{M}$ and strategies in $\mathcal{G}$ using $\mathcal{M}$ as memory structure, and similarly between strategies in $\mathcal{G} \times \mathcal{M}$ using $\mathcal{M}'$ as memory structure and strategies in $\mathcal{G}$ using $\mathcal{M} \times \mathcal{M}'$ as memory structure. We often implicitly reason in the expanded graph $G \times \mathcal{M}$ when consider a strategy using $\mathcal{M}$ as memory structure.

Definition 11 (Positional and finite-memory determinacy)

Consider a condition $W \subseteq A^\omega$.

- We say that W is *positionally determined* if the following holds: for all games with condition W, for all initial vertices, if Eve has a winning strategy, then she has a positional winning strategy.
- We say that W is *finite-memory determined* if the following holds: for all games with condition W, for all initial vertices, if Eve has a winning strategy, then she has a finite-memory winning strategy.
- We say that W is *bounded-memory determined* if the following holds: there exists a bound $\text{mem} \in \mathbb{N}$ such that for all games with condition W, for all initial vertices, if Eve has a winning strategy, then she has a finite-memory winning strategy using mem memory states.

Note that here, we only consider the case of Eve, so the classical terminology talks about *half*-positionally determined conditions in this case, see for instance [Gim07; Kop09]. As we will be only interested in the case of Eve, we use "positionally determined" for "half-positionally determined".

The following positionality result has far reaching consequences in automata theory.

Theorem 5 (Positionality of parity games [EJ88])

Parity games are positionally determined.

Problem 3 (Finite- and bounded-memory determinacy for boundedness games)

Are uniform and non-uniform boundedness games finite-memory determined? Bounded-memory determined?

We instantiate the definitions of finite- and bounded-memory determinacy for boundedness games.

For a boundedness game and v_0 an initial vertex, we denote $\text{val}(v_0)$ the smallest N such that Eve has a strategy ensuring $B(N) \cap \text{Parity}$.

(i) **Bounded-memory determinacy for uniform boundedness games**

For all $k, d \in \mathbb{N}$, there exists $\text{mem} \in \mathbb{N}$ such that for all boundedness games with k counters, $d + 1$ colors and initial vertices v_0, there exists a strategy using mem memory states ensuring $B(\text{val}(v_0)) \cap \text{Parity}$.

(ii) **Finite-memory determinacy for uniform boundedness games**

For all $k, d \in \mathbb{N}$, for all boundedness games with k counters, $d+1$ colors and initial vertices v_0, there exists a strategy using finitely many memory states ensuring $B(\text{val}(v_0)) \cap \text{Parity}$.

(iii) **Bounded-memory determinacy for non-uniform boundedness games**

For all $k, d \in \mathbb{N}$, there exists mem $\in \mathbb{N}$ such that for all boundedness games with k counters, $d+1$ colors and initial vertices, if Eve has a strategy ensuring $B \cap \text{Parity}$, then she has one using mem memory states.

(iv) **Finite-memory determinacy for non-uniform boundedness games**

For all $k, d \in \mathbb{N}$, for all boundedness games with k counters, $d+1$ colors and initial vertices, if Eve has a strategy ensuring $B \cap \text{Parity}$, then she has one using finite memory.

We will answer those four questions:

- we show in Subsection 3.1 that (ii) holds and (i) does not hold,
- we show in Section 5 that (iii) and (iv) do not hold.

As we will see, the LoCo conjecture introduces another dimension to these questions: it allows distortion of the value, see Section 3.

2.2 Some results for Non-Uniform Boundedness Games

This subsection discusses the results obtained in collaboration with Krishnendu Chatterjee [CF13] and Martin Zimmermann [FZ12; FZ14].

Both papers are related to the finitary conditions, which have been introduced in verification by Alur and Henzinger [AH98] as a strengthening of liveness conditions. Intuitively, a liveness condition requires that some event occurs *eventually*, its finitary counterpart strengthens this by requiring an overall bound on the waiting time.

We now define the finitary parity condition with $d+1$ colors over the alphabet $A = \{0, \ldots, d\}$. Given a sequence $s = c_0 \cdot c_1 \cdots$ and $k \in \mathbb{N}$, we define:

$$\text{dist}_k(s) = \inf\{k' - k \mid k' \geqslant k,\ c_{k'} \text{ is even and } c_{k'} \leqslant c_k\};$$

i.e $\text{dist}_k(s)$ is the "waiting time" by means of number of transitions followed from the kth position before reaching a preferable color (that is, even and lower or equal than in the kth position). The finitary parity condition was defined as follows in [CHH09]:

$$\text{FinParity} = \{s \mid \limsup_k \text{dist}_k(s) < \infty\};$$

i.e, it requires that the supremum limit of the distance sequence is bounded.

The finitary parity condition can be seen up to encoding as a special case of the non-uniform B-parity condition. The specificity here is that the counters are used to quantify how well is the parity condition satisfied. This makes the situation very different from the general case of B-parity conditions, where the parity condition and the counters are completely independent.

Chatterjee, Henzinger and Horn introduced and studied games with finitary parity conditions [CHH09]. Their main result is that determining the winner in a finitary parity game can be achieved in cubic time, which should be contrasted with the situation for parity games, where determining the winner is not known to be doable in polynomial time. Furthermore, they showed that finitary parity games over finite arenas are, as parity games, positionally determined:

Theorem 6 (Positionality of finitary parity games over finite arenas [CHH09])

Finitary parity games over finite arenas are positionally determined.

Inspired by these results, we introduced in collaboration with Martin Zimmermann a condition called cost-parity condition, generalizing both parity conditions and finitary parity conditions. We proved that the corresponding games, called parity games with costs, have the best expected properties, *i.e.* that they do not get worse than any of the two conditions they generalise:

Theorem 7 (Parity games with costs)

- Parity games with costs played over finite arenas are positionally determined.
- Given an algorithm that solves parity games played over finite arenas in time $T(n, m, d)$, there is an algorithm that solves parity games with costs played over finite arenas in time $O(n \cdot T(dn, dm, d + 2))$, where n is the number of vertices, m is the number of edges and d the number of colors.

We refer to [FZ14] for the definitions and proofs.

The second generalization we considered was to extend the positionality result for finitary parity conditions to infinite arenas. Relying on different ideas and techniques to tackle infinite arenas, we proved the following result in collaboration with Krishnendu Chatterjee [CF13]:

Theorem 8 (Bounded-memory determinacy of finitary parity games)

Finitary parity games with $d+1$ colors are bounded-memory determined, with strategies using $d/2$ memory states.

We refer to [CF13] for the proof.

A natural question is whether this result can be extended to the general case of non-uniform boundedness games. The answer is no, as we will show in Section 5 that non-uniform boundedness games are not bounded-memory determined.

2.3 Memory for Topologically Closed Conditions

This subsection reports on some results obtained in collaboration with Thomas Colcombet and Florian Horn, published in [CFH14].

The initial observation is the following simple positionality result:

Fact 1 (Positionality of uniform B games with one counter)

Uniform B games with one counter are positionally determined, *i.e.* for all B games with one counter and initial vertices v_0, Eve has a positional strategy ensuring $B(\mathrm{val}(v_0))$.

Proof. Consider a B game $(\mathcal{G}, B)$ with one counter, and a strategy σ ensuring $B(\mathrm{val}(v_0))$. Denote $N = \mathrm{val}(v_0)$. Observe that the condition $B(N)$ is closed under suffix, as defined in Lemma 1, hence all plays consistent with σ remain in $\mathrm{Win}_E(\mathcal{G}, B(N))$. For every vertex $v \in \mathrm{Win}_E(\mathcal{G}, B(N))$, there exists a maximal value $N(v) \leqslant N$ such that σ ensures $B(N)$ from v, starting with $N(v)$ as counter value.

Informally speaking, this value $N(v)$ is the worst-case scenario from v, so in this case the strategy σ has to produce its best move. Indeed, the strategy can play some non-optimal moves as long as the counter value remains small, and start playing optimal moves when the situation gets dangerous, *i.e.* if the counter value gets closer to N. Looking at the move that σ prescribes from v with counter value $N(v)$ guarantees the best outcome. Formally, this means that the strategy σ ensures that the chosen edge leads to a vertex v' with a counter value N' such that σ ensures $B(N)$ from v', starting with N' as counter value. Consequently, $N' \leqslant N(v')$.

We construct a positional strategy σ' ensuring $B(N)$ by playing from v as σ would have played from v with counter value $N(v)$. The reasoning above ensures that this strategy ensures to remain forever in the winning region $\mathrm{Win}_E(\mathcal{G}, B(N))$, implying that the strategy σ' is winning. ∎

Our objective is to generalise this reasoning. The key observation here is that the condition $B(N)$ is topologically closed, as we will define now. We will show that the reasoning sketched above extends to all topologically closed conditions.

Problem 4 (Characterising memory requirement)

Given $W \subseteq A^\omega$, characterise the following quantity:

$$\mathrm{mem}(W) \quad = \quad \sup_{(\mathcal{G},W) \text{ game}} \ \inf_{\substack{\sigma \text{ winning} \\ \text{strategy}}} \ \mathrm{mem}(\sigma).$$

In words, $\mathrm{mem}(W)$ is the necessary and sufficient number of memory states for constructing a winning strategy in games with condition W. Equivalently:

- *upper bound:* for all games $(\mathcal{G}, W)$, for all initial vertices, if Eve has a winning strategy, then she has a winning strategy using at most $\mathrm{mem}(W)$ memory states,
- *lower bound:* there exists a game $(\mathcal{G}, W)$ and an initial vertex such that Eve has a winning strategy, but no winning strategy using less than $\mathrm{mem}(W)$ memory states.

The quantity $\mathrm{mem}(W)$ has been considered in two different works before. First, Dziembowski, Jurdziński and Walukiewicz characterised $\mathrm{mem}(W)$ for all Muller conditions W, *i.e.* for Boolean combinations of the conditions "color c appears infinitely many times" [DJW97].

In this subsection, we characterise mem(W) for all conditions W that are closed (with respect to the Cantor topology over infinite words). We will use the following property as a definition: a condition W over an alphabet A is closed if, and only if, it is given by a subset $P \subseteq A^*$ of forbidden prefixes of colors, inducing the condition

$$W = \{c_0 \cdot c_1 \cdots \mid \forall i \in \mathbb{N}, c_0 \cdot c_1 \cdots c_i \notin P\}.$$

Note that closed conditions are incomparable with ω-regular conditions, so our results are incomparable with [DJW97; Kop06; Kop09].

For the remainder of this section, we fix a closed condition W induced by $P \subseteq A^*$.

***Remark* 2.** *We restrict ourselves to arenas with finite out-degree: for every vertex $v \in V$, the set $\{v' \mid (v, v') \in E\}$ is finite. In particular, in the definition of mem(W), the supremum ranges over all arenas of finite out-degree. This assumption will be used only in the proof of Lemma 6, but we will show that the result fails without this assumption.*

A First Upper Bound

We first give an upper bound on mem(W). Let $w \in A^*$, define its left quotient as:

$$w^{-1}W = \{s \in A^\omega \mid w \cdot s \in W\}.$$

We denote Res(W) the set of left quotients of W. We mention some special left quotients: the initial one, $\varepsilon^{-1}W$ (equal to W), and the empty one, obtained as $w^{-1}W$ for any $w \in P$. Recall that Res(W) is finite if, and only if, W is regular, and in such case it can be used to describe the set of states of the minimal deterministic automaton recognizing W.

This remark allows to reduce games with closed conditions W to safety games, and to use the following folklore result:

Lemma 2 (Positionality of safety games)

Safety games are positionally determined.

We will often use a slightly more general result in this chapter:

Lemma 3 (Positionality of safety games, take 2)

Consider a condition $W \subseteq A$, an arena $\mathcal{G}$ equipped with a colouring function $c : E \to A \times \{0,1\}$, *i.e.* by two colouring functions $c_A : E \to A$ and $c_S : E \to \{0,1\}$, and an initial vertex. This gives rise to the game $(\mathcal{G}, W \cap \text{Safe})$, where W uses the colouring function c_A and Safe the colouring function c_S.

If there exists a winning strategy in $(\mathcal{G}, W \cap \text{Safe})$ using mem memory states, then there exists a winning strategy in $(\mathcal{G}, W)$ using mem memory states.

The following lemma reduces games with closed conditions to safety games, and constructs the most important tool in the proofs to follow:

Lemma 4 (First upper bound)

For all games $(\mathcal{G}, W)$ with a closed condition W, for all initial vertices, if Eve has a winning strategy, then she has a winning strategy using at most $|\mathrm{Res}(W)|$ memory states.

Consequently,

$$\mathrm{mem}(W) \leqslant |\mathrm{Res}(W)| .$$

Proof. We construct a memory structure $\mathcal{M}$, as follows: the set of memory states is $\mathrm{Res}(W)$, the initial memory state is W and the update function is μ, where $\mu(w^{-1}W, a) = (w \cdot a)^{-1}W$. (It is easy to check that this is well defined, *i.e.* independent of the representant w chosen).

At any point in the game, the memory state computed by $\mathcal{M}$ is the current left quotient.

Let $\mathcal{G}$ be an arena with a colouring function $c : E \to A$. We construct the expanded arena $\mathcal{G} \times \mathcal{M}$ equipped with the colouring function $c' : E \times \mathrm{Res}(W) \to \{0,1\}$ defined by:

$$c'(\_, w^{-1}W) = \begin{cases} 1 & \text{if } w^{-1}W = \varnothing \text{ (equivalently, } w \in P), \\ 0 & \text{otherwise.} \end{cases}$$

We equip $\mathcal{G} \times \mathcal{M}$ with the safety condition, giving rise to the game $(\mathcal{G} \times \mathcal{M}, \mathrm{Safe})$. First observe that by construction, the plays in $\mathcal{G} \times \mathcal{M}$ are of the form $(e_0, c(e_0)^{-1}W) \cdot (e_1, c(e_0 \cdot e_1)^{-1}W) \cdots (e_k, c(e_0 \cdot e_1 \cdots e_k)^{-1}W)$, so by definition of c' a play is winning in $(\mathcal{G} \times \mathcal{M}, \mathrm{Safe})$ if, and only if, its projection (on the first component) is winning in $(\mathcal{G}, W)$.

It follows that a winning strategy for Eve in $(\mathcal{G}, W)$ from v_0 induces a winning strategy in $(\mathcal{G} \times \mathcal{M}, \mathrm{Safe})$ from (v_0, W). Now, thanks to Lemma 2, since Eve wins in $(\mathcal{G} \times \mathcal{M}, \mathrm{Safe})$, she has a positional winning strategy. This induces a winning strategy in $(\mathcal{G}, W)$ using $\mathcal{M}$ as memory structure, concluding the proof of Lemma 4. ∎

The game $(\mathcal{G} \times \mathcal{M}, \mathrm{Safe})$ defined above will be an important tool in the proofs to follow. We will also rely on the following remark: assume we want to prove that a strategy σ is winning. Then it is enough to show that for all plays π consistent with σ, for all k, $c(\pi_k)^{-1}W \neq \varnothing$, where π_k is the prefix of π of length k. This simple observation follows from the definition of safety conditions.

A Tighter Upper Bound

The memory structure $\mathcal{M}$ is not optimal. A first remark is that the empty left quotient (which exists if $W \neq A^\omega$) can be removed from the memory states as the game is lost. From now on by "left quotient" we mean "non-empty left quotient of W", and in particular $\mathrm{Res}(W)$ denotes the set of non-empty left quotients.

The second remark is the following: let L_1 and L_2 two left quotients, such that $L_1 \subseteq L_2$. With the same notations as above, consider a vertex v in the arena $\mathcal{G}$. If Eve wins from (v, L_1) in $(\mathcal{G} \times \mathcal{M}, \mathrm{Safe})$, then she also wins from (v, L_2): indeed, she can play as she would have played from (v, L_1). Since this ensures from v that all plays are winning for L_1, then a fortiori they are winning for L_2.

This suggests to restrict the memory states only to *minimally winning* left quotients with respect to inclusion. Two issues arise:

- which left quotients are winning depends on the current vertex, so the semantics of a memory state can no longer be *one* left quotient, but rather a left quotient for each possible vertex,

- there may not exist *minimally winning* left quotients.

For the sake of presentation, we first show how to deal with the first issue, assuming the second issue does not appear. Specifically, in the following lemma, we assume that $\mathrm{Res}(W)$ is finite (*i.e.* W is regular), implying the existence of minimally winning left quotients. We will later drop this assumption.

We define the width of an ordered set $(E, \leqslant)$ as the cardinal of the maximal antichain of E with respect to $\leqslant$, *i.e.* the cardinal of the largest set of pairwise incomparable elements.

Lemma 5 (Upper bound in the regular case)

Assume that $\mathrm{Res}(W)$ is finite.

For all games $(\mathcal{G}, W)$ with a closed condition W, for all initial vertices, if Eve has a winning strategy, then she has a winning strategy using at most K memory states, where K is the width of $(\mathrm{Res}(W), \subseteq)$.

Proof. We use the same notations as for the proof of Lemma 4, and construct a smaller memory structure together with a winning strategy using this memory structure. In this proof, by winning we mean winning in the game $(\mathcal{G} \times \mathcal{M}, \mathrm{Safe})$.

Let K be the cardinal of the maximal antichain of left quotients of W. We construct the memory structure $\mathcal{M}^*$ whose set of memory sates is $\{1, \ldots, K\}$ initial memory state is 1 and update function is μ, and the next-move function σ inducing the strategy also denoted σ.

Let v be a vertex in $\mathcal{G}$. We consider the set of minimal left quotients L such that (v, L) is winning. (Here we use the finiteness of $\mathrm{Res}(W)$ to guarantee the existence of such left quotients.) This is an antichain, so there are at most K of them, we denote them by $L_1(v), \ldots, L_p(v)$, for some $p \leqslant K$. The *key* property is that for every left quotient L such that (v, L) is winning, there exists i such that $L_i(v) \subseteq L$. Furthermore, we choose $L_1(v_0)$ such that $L_1(v_0) \subseteq W$. (Indeed, by assumption (v_0, W) is winning.)

We define the update function: $\mu(i, (v, v'))$ is a j such that $L_j(v') \subseteq L_i(v) \cdot c(v, v')$. Note that in general, such a j may not exist; it does exist if $(v', L_i(v) \cdot c(v, v'))$ is winning, and we will prove that this will always be the case when playing the strategy σ.

We define the next-move function σ. Let $v \in V_E$, and consider $(v, L_i(v))$: since Eve wins from there, there exists an edge $(v, v') \in E$ such that $(v', L_i(v) \cdot c(v, v'))$ is winning. Define $\sigma(v, i)$ to be this v'.

We show that the strategy σ is winning. Consider a play $\pi = (v_0, v_1) \cdot (v_1, v_2) \cdots$ consistent with σ, and $i_0 \cdot i_1 \cdots$ the sequence of memory states assumed along this play. Denote π_k the prefix of π of length k, we prove that for all k, $L_{i_k}(v_k) \subseteq c(\pi_k)^{-1}W$. Note that by definition, $(v_k, L_{i_k}(v_k))$ is winning, so $L_{i_k}(v_k) \neq \emptyset$, implying that $c(\pi_k)^{-1}W \neq \emptyset$.

We proceed by induction. For $k = 0$, it follows from $L_1(v_0) \subseteq W$. Let $k > 0$, the induction hypothesis is $L_{i_{k-1}}(v_{k-1}) \subseteq c(\pi_{k-1})^{-1}W$. We distinguish two cases.

- Either v_{k-1} belongs to Eve, then by construction of σ we have that $(v_k, L_{i_{k-1}}(v_{k-1}) \cdot c(v_{k-1}, v_k))$ is winning. It follows that the update function is well defined, and $L_{i_k}(v_k) \subseteq L_{i_{k-1}}(v_{k-1}) \cdot c(v_{k-1}, v_k)$, which together with the induction hypothesis implies $L_{i_k}(v_k) \subseteq c(\pi_k)^{-1}W$.

- Or v_{k-1} belongs to Adam. Since Adam cannot escape $\mathrm{Win}_E(\mathcal{G} \times \mathcal{M}, \mathrm{Safe})$, we have that $(v_k, L_{i_{k-1}}(v_{k-1}) \cdot c(v_{k-1}, v_k))$ is winning, and the same reasoning concludes.

It follows that the strategy σ is winning, concluding the proof of Lemma 5. ∎

We now get rid of the regularity assumption. This means that for a vertex v, there may not be a minimal left quotient L such that (v, L) is winning. To get around this difficulty, the semantics of a memory state is a not anymore a left quotient for each vertex, but rather a decreasing sequence of left quotients for each vertex.

In the following proof of Lemma 6, we will use the assumption that the out-degree of every vertex is finite, *i.e.* for every vertex $v \in V$, the set $\{v' \mid (v, v') \in E\}$ is finite.

Lemma 6 (Upper bound)

For all games $(\mathcal{G}, W)$ with a closed condition W over an arena with finite out-degree, for all initial vertices, if Eve has a winning strategy, then she has a winning strategy using at most K memory states, where K is the width of $(\mathrm{Res}(W), \subseteq)$.

Consequently, $\mathrm{mem}(W)$ is smaller than or equal to the width of $(\mathrm{Res}(W), \subseteq)$.

Proof. We use the same notations as for the proof of Lemma 5, and construct a memory structure together with a winning strategy using this memory structure.

Let K be the cardinal of the maximal antichain of left quotients of W. We construct the memory structure $\mathcal{M}^*$ whose set of memory states is $\{1, \ldots, K\}$ initial memory state is 1 and update function is μ, and the next-move function σ inducing the strategy also denoted σ.

Let v be a vertex in $\mathcal{G}$. We consider the set $\mathrm{Win}(v)$ of left quotients L such that (v, L) is winning. We split $\mathrm{Win}(v)$ into maximal decreasing (finite or infinite) sequences of left quotients, denoted $\ell_1(v), \ldots, \ell_p(v)$, for some $p \leqslant K$. Furthermore, we choose $\ell_1(v_0)$ such that $W \in \ell_1(v_0)$. (Indeed, by assumption (v_0, W) is winning.)

We say that (v, ℓ) is winning if for all $L \in \ell$, we have that (v, L) is winning. For ℓ a sequence of left quotients and $a \in A$, we define $\ell \cdot a$ component-wise. Note that even if ℓ is infinite, it may be that $\ell \cdot a$ is finite.

We define the update function: $\mu(i, (v, v'))$ is a j as follows.

- If $\ell_i(v) \cdot c(v, v')$ is finite, denote $L \cdot c(v, v')$ its last element. Choose j such that $L \cdot c(v, v') \in \ell_j(v')$. Note that in general, such a j may not exist; it does exist if $(v', \ell_i(v) \cdot c(v, v'))$ is winning, and we will prove that this will always be the case when playing the strategy σ.

- If $\ell_i(v) \cdot c(v, v')$ is infinite, then choose j such that $\ell_j(v')$ has an infinite intersection with $\ell_i(v) \cdot c(v, v')$. Such a j exists without any assumption.

We define the next-move function σ. Let $v \in V_E$, and consider $(v, \ell_i(v))$. Let $L \in \ell_i(v)$, Eve wins from (v, L), so there exists an edge $(v, v') \in E$ such that $(v', L \cdot c(v, v'))$ is winning, we say that $(v, v') \in E$ is good for L. Since $\mathrm{Win}(v')$ is upward closed, if $(v, v') \in E$ is good for L, then it is good for every L' such that $L \subseteq L'$. We argue that there exists an edge $(v, v') \in E$ that is good for all $L \in \ell_i(v)$, *i.e.* such that $(v', \ell_i(v) \cdot c(v, v'))$ is winning; define $\mu(v, i)$ to be this v'. There are two cases:

- Either $\ell_i(v)$ is finite, denote L its last element. Since $\ell_i(v)$ is decreasing, an edge good for L is good for all $L' \in \ell_i(v)$.

- Or $\ell_i(v)$ is infinite. The vertex v has finite degree, so there exists an edge which is good for infinitely many $L \in \ell_i(v)$. Since $\ell_i(v)$ is decreasing, it is good for all $L' \in \ell_i(v)$.

We show that the strategy σ is winning. Consider a play $\pi = (v_0, v_1) \cdot (v_1, v_2) \cdots$ consistent with σ, and $i_0 \cdot i_1 \cdots$ the sequence of memory states assumed along this play. Denote π_k the prefix of π of length k, we prove that for all k, there exists $L \in \ell_{i_k}(v_k)$ such that $L \subseteq c(\pi_k)^{-1}W$. Note that by definition, $(v_k, \ell_{i_k}(v_k))$ is winning, so $c(\pi_k)^{-1}W \neq \emptyset$.

We proceed by induction. For $k = 0$, it follows from $W \in \ell_1(v_0)$. Let $k > 0$, the induction hypothesis implies the existence of $L \in \ell_{i_{k-1}}(v_{k-1})$ such that $L \subseteq c(\pi_{k-1})^{-1}W$. We distinguish two cases, and denote $c_k = c(v_{k-1}, v_k)$.

- Either v_{k-1} belongs to Eve, then by construction of σ we have that $(v_k, \ell_{i_{k-1}}(v_{k-1}) \cdot c_k)$ is winning. It follows that the update function is well defined, and:

  1. If $\ell_{i_{k-1}}(v_{k-1}) \cdot c_k$ is finite, denote $L' \cdot c_k$ its last element, we have $L' \cdot c_k \in \ell_{i_k}(v_k)$. Since $L' \cdot c_k$ is the last element of $\ell_{i_{k-1}}(v_{k-1}) \cdot c_k$, it follows that $L' \subseteq L$. We have thus $L' \cdot c_k \subseteq L \cdot c_k$, so $L' \subseteq c(\pi_k)^{-1}W$, and $L' \cdot c_k \in \ell_{i_k}(v_k)$.
  2. If $\ell_{i_{k-1}}(v_{k-1}) \cdot c_k$ is infinite, $\ell_{i_k}(v_k)$ has an infinite intersection with $\ell_{i_{k-1}}(v_{k-1}) \cdot c_k$. So there exists $L' \subseteq L$ with $L' \in \ell_{i_{k-1}}(v_{k-1})$ such that $L' \cdot c_k$ is in this intersection. We have $L' \cdot c_k \in \ell_{i_k}(v_k)$ and $L' \subseteq c(\pi_k)^{-1}W$.

- Or v_{k-1} belongs to Adam. Since Adam cannot escape $\mathrm{Win}_E(\mathcal{G} \times \mathcal{M}, \mathrm{Safe})$, we have that $(v_k, \ell_{i_{k-1}}(v_{k-1}) \cdot c_k)$ is winning, and the same reasoning concludes.

It follows that the strategy σ is winning, concluding the proof of Lemma 6. ∎

A Matching Lower Bound

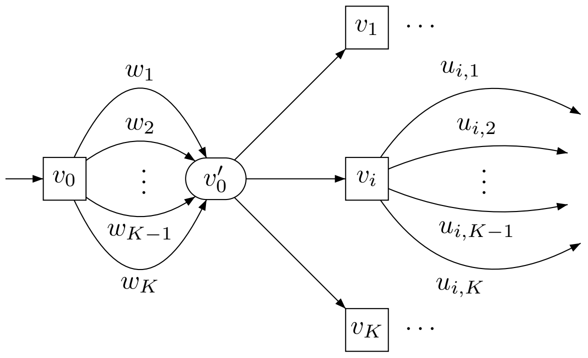


Figure 3: The lower bound.

Lemma 7 (Lower bound)

For all closed conditions W, there exists a game $(\mathcal{G}, W)$ and an initial vertex such that Eve has a winning strategy, but no winning strategy using less than K memory states where K is the width of $(\mathrm{Res}(W), \subseteq)$.

Consequently, $\mathrm{mem}(W)$ is greater than or equal to the width of $(\mathrm{Res}(W), \subseteq)$.

Proof. Consider $\{w_1^{-1}W, \dots, w_K^{-1}W\}$ an antichain of left quotients of W. For $i \neq j$, there exists $u_{i,j} \in A^\omega$ such that $u_{i,j} \in w_i^{-1}W$ and $u_{i,j} \notin w_j^{-1}W$.

We describe the game, illustrated in Figure 3. A play consists in three steps:

1. From v_0 to v_0': Adam chooses a word in $\{w_1, \dots, w_K\}$;
2. Eve chooses between K options;
3. say Eve chose the i^{th} option, then Adam chooses between the $K-1$ words $u_{i,j}$ for $j \neq i$.

We first show that Eve has a winning strategy from v_0, using K memory states. It consists in choosing the i^{th} option whenever Adam chooses the word u_i: whatever Adam chooses at the third step, $w_i \cdot u_{i,j} \in W$.

We now show that there exists no winning strategy using less than K memory states. Indeed, such a strategy will not comply with the above strategy and for some $i \neq j$, choose the j^{th} option if Adam chooses w_i. Then Adam wins by playing $u_{i,j}$, since $w_i \cdot u_{i,j} \notin W$. ∎

Tight Bounds

Putting together upper and lower bounds, we proved the following result:

Theorem 9 (Memory for closed conditions)

For all closed conditions W, $\text{mem}(W)$ is the width of $(\text{Res}(W), \subseteq)$.

Theorem 9 allows to uniformly derive memory requirements for different conditions, such as the $B(N)$-condition with one counter, which was the original motivation, but also the generalised reachability condition, that we introduced and studied in collaboration with Florian Horn [FH13], and the energy condition, see [CFH14] for more details about these applications.

We conclude by giving an example showing that the result would fail without the finite out-degree assumption.

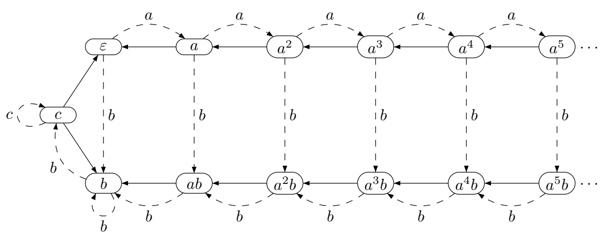


Figure 4: The outbidding condition: more b's than a's.

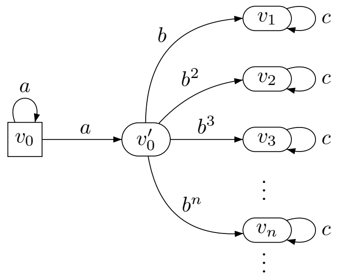


Figure 5: An outbidding game with infinite out-degree where Eve needs infinite memory to win.

Let $A = \{a, b, c\}$ and $W = \{a^n \cdot b^p \cdot c^\omega \mid n \leqslant p\} \cup \{a^\omega\} \cup a^* \cdot b^\omega$. It is a non-regular closed condition, called the outbidding condition. Figure 4 represents the partial order $(\text{Res}(W), \subseteq)$: the solid edges represent inclusions of left quotients, and the dashed edges figure the minimal (yet infinite) deterministic automaton recognizing W. Its width is 3: there are two incomparable infinite increasing sequences of left quotients, $((a^n)^{-1}W)_{n\in\mathbb{N}}$ and $((a^n \cdot b)^{-1}W)_{n\in\mathbb{N}}$, and $c^{-1}W$.

Hence thanks to Theorem 9, $\mathrm{mem}(W) = 3$. However, there exists an outbidding game where Eve wins but needs infinite memory for this. This does not contradict Theorem 9, as this game, represented in Figure 5, has a vertex of infinite out-degree. It goes as follows: first Adam picks a number n, and then Eve takes over: she has to pick a number p, higher than or equal to n. A strategy using finitely many memory states can only choose from finitely many options, hence cannot win against all strategies of Adam.

3 The LoCo Conjecture

In this section, we state the LoCo conjecture.

We give a first statement of the conjecture in Subsection 3.1, and show what it implies in the theory of regular cost functions in Subsection 3.2. We then refine the conjecture by considering structural properties of the arenas in Subsection 3.3.

3.1 A First Statement of the Conjecture

The LoCo conjecture is a bounded-memory determinacy statement, that introduces a trade-off between memory and bounds:

Conjecture 1 (LoCo conjecture)

For all $k, d \in \mathbb{N}$, there exists mem $\in \mathbb{N}$ and $\alpha : \mathbb{N} \to \mathbb{N}$ such that for all boundedness games with k counters, $d+1$ colors and initial vertices v_0, there exists a strategy σ using mem memory states ensuring $B(\alpha(\mathrm{val}(v_0))) \cap \mathrm{Parity}$.

The function α is called a trade-off function: at the price of increasing the bound from $\mathrm{val}(v_0)$ to $\alpha(\mathrm{val}(v_0))$, one can use a bounded-memory strategy.

Note that we state the LoCo conjecture here only for Eve. Throughout this chapter, we will only consider strategies for Eve. As we will see in Subsection 3.2, this allows to prove the equivalence between alternating and non-deterministic B-parity automata over infinite trees. To obtain the corresponding equivalence for S-parity automata, and the decidability of cost-MSO over infinite trees, one needs to also prove the LoCo conjecture for Adam. We do not consider this here.

$$\begin{tikzcd}
 & \text{finite uniform} & & \\
\text{bounded uniform} \arrow[ur] \arrow[dr] & & & \text{finite non-uniform} \\
 & \text{LoCo} \arrow[r, "\text{fixpoint}"] & \text{bounded non-uniform} \arrow[ur] &
\end{tikzcd}$$

Figure 6: Implications.

To get a better understanding of this conjecture, we make a series of observations.

1. We show the implications between the LoCo conjecture and finite- and bounded-memory determinacy for boundedness games, as illustrated in Figure 6.

2. We show that the uniform boundedness games are not bounded-memory determined.

3. We show that the non-uniform boundedness games are finite-memory determined.

We start by the implications illustrated in Figure 6. Three of them are simple logical implications, so we focus on the last one, overlined "fixpoint".

Lemma 8 (Fixpoint)

Consider a prefix-independent condition W, *i.e.* closed under prefixes and suffixes: for π a finite play and π' an infinite play, $\pi \cdot \pi' \in W$ if, and only if, $\pi' \in W$.

Assume that there exists mem $\in \mathbb{N}$ such that for all games $(\mathcal{G}, W)$, if $\text{Win}_E(\mathcal{G}, W)$ is non-empty, then there exists a vertex v_0 such that Eve has a winning strategy from v_0 using mem memory states. Then W is bounded-memory determined.

Before proving this lemma, let us see how it helps to show the implication from Figure 6. Assume that the LoCo conjecture holds, we prove that non-uniform boundedness games are bounded-memory determined.

Let $(\mathcal{G}, B \cap \text{Parity})$ be a non-uniform boundedness game. We show that the assumptions of Lemma 8 hold; first of all the condition $B \cap \text{Parity}$ is prefix-independent. We argue that the following holds:

$$\text{If } \text{Win}_E(\mathcal{G}, B \cap \text{Parity}) \text{ is non-empty, then there exists } N \in \mathbb{N} \text{ such that } \text{Win}_E(\mathcal{G}, B(N) \cap \text{Parity}) \text{ is non-empty.}$$

We show the contrapositive. Assume that for all $N \in \mathbb{N}$, $\text{Win}_E(\mathcal{G}, B(N) \cap \text{Parity})$ is empty, it means that Adam has a strategy τ_N ensuring the complement of $B(N) \cap \text{Parity}$: in a play consistent with τ_N, either at some point some counter reaches value $N+1$, or the infinite play does not satisfy the parity condition. We construct a strategy τ for Adam that ensures the complement of $B \cap \text{Parity}$, implying that $\text{Win}_E(\mathcal{G}, B(N) \cap \text{Parity})$ is empty. The strategy τ simulates the strategies $(\tau_N)_{N \in \mathbb{N}}$, starting from τ_1 and switching from τ_N to τ_{N+1} if at some point some counter reaches value $N+1$.

Let $v_0 \in \text{Win}_E(\mathcal{G}, B(N) \cap \text{Parity})$, thanks to the LoCo conjecture (which we assume here holds), this yields a strategy ensuring $B(\alpha(N)) \cap \text{Parity}$ using mem memory states. A fortiori, this strategy ensures $B \cap \text{Parity}$.

It follows from Lemma 8 that under the assumption that the LoCo conjecture holds, the non-uniform condition $B \cap \text{Parity}$ is bounded-memory determined. As we will see in Section 5, the LoCo conjecture does not hold.

We now prove Lemma 8. It is a rather classical technique, which was for instance heavily used in [Kop06; Kop09; CF13]. To prove it, we need the following lemma, which raises the question of *uniformity* for winning strategies. So far, all finite-memory determinacy statements were for a given initial vertex. A stronger finite-memory determinacy statement is obtained by asserting the existence of *one* strategy, which would be winning from all vertices of the winning region; such a strategy is called *uniform*. Note that this notion of uniformity is not related at all to the uniformity in boundedness games.

Lemma 9 (Uniform strategies)

Consider a condition W closed under suffixes: for π a finite play and π' an infinite play, if $\pi \cdot \pi' \in W$, then $\pi' \in W$.

Let Y be a subset of vertices, and for all vertices $v \in Y$, a strategy σ_v using mem memory states winning from v. Then there exists a uniform strategy using mem memory states, winning from Y.

Proof. Without loss of generality, we assume that the strategies σ_v use the same memory structure $\mathcal{M}$. We construct a uniform strategy σ using the memory structure $\mathcal{M}$, which is winning from all vertices of Y.

Consider $\preceq$ a well order on V (the existence of such a well order is a consequence of the axiom of choice). For a vertex $v \in V$ and a memory state $m \in M$, we define $\sigma(v, m)$ as $\sigma_{v_0}(v, m)$, where v_0 is the least element with respect to the well-order $\preceq$ such that $\sigma_{v_0}(v, m)$ is defined (*i.e.* there exists a play from (v_0, m_0) to (v, m)).

We argue that σ is winning. Consider a play consistent with σ from some vertex of Y. By definition, it is ultimately consistent with some strategy σ_v, so a suffix of the play belongs to W. By closure under suffixes, this implies that the play itself belongs to W. ∎

We can now proceed to the proof of Lemma 8.

Proof. Consider a game $(\mathcal{G}, W)$. We define the following sequences indexed by ordinals:

$$\begin{cases} X_0 = \emptyset \\ Y_0 = \left\{ v \in V \;\middle|\; \begin{array}{l} \text{Eve has a winning strategy in } \mathcal{G} \text{ from } v \\ \text{using mem memory states} \end{array} \right\} \\ X_{\alpha+1} = X_\alpha \cup Y_\alpha \\ Y_{\alpha+1} = \left\{ v \in V \;\middle|\; \begin{array}{l} \text{Eve has a winning strategy in } \mathcal{G}[V \backslash X_{\alpha+1}] \text{ from } v \\ \text{using mem memory states} \end{array} \right\} \\ X_\beta = \bigcup_{\alpha<\beta} X_\alpha \cup Y_\alpha \\ Y_\beta = \left\{ v \in V \;\middle|\; \begin{array}{l} \text{Eve has a winning strategy in } \mathcal{G}[V \backslash X_\beta] \text{ from } v \\ \text{using mem memory states} \end{array} \right\} \end{cases}$$

Here β denotes a limit ordinal, and $\mathcal{G}[X]$ is the arena induced by the subset of vertices X.

By assumption $\bigcup_\alpha X_\alpha = \text{Win}_E(\mathcal{G}, W)$. Thanks to Lemma 9, for every ordinal α there exists a strategy σ_α using mem memory states which is winning from Y_α. The same reasoning as in Lemma 1 shows that all plays consistent with σ_α in $\mathcal{G}[V \backslash X_\alpha]$ remain in Y_α. However, this is not true anymore when considering plays consistent with σ_α in $\mathcal{G}$: a play can go from Y_α to Y_β for $\beta < \alpha$.

Consider the strategy σ which is the disjoint sum of the strategies σ_α. It uses mem memory states. The above observation implies that all plays consistent with σ are ultimately consistent with one σ_α, so a suffix of the play belongs to W. By closure under suffixes, this implies that the play itself belongs to W, and σ is winning from $\text{Win}_E(\mathcal{G}, W)$, which concludes. ∎

We now show that the uniform boundedness games are not bounded-memory determined:

Lemma 10 (Uniform boundedness games are not bounded-memory determined)

For all $N \in \mathbb{N}$, there exists a B-reachability game and an initial vertex v_0 such that:

- $\text{val}(v_0) = N$, so Eve has a strategy ensuring $B(N)$ Until Reach using $N+1$ memory states,
- no strategy using less than $N+1$ memory states ensures $B(N)$ Until Reach.

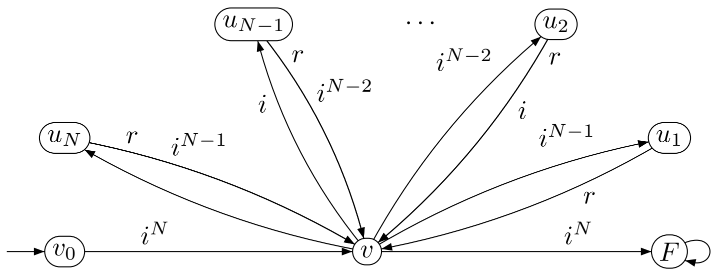


Figure 7: Eve needs $N+1$ memory states to ensure $B(N)$ Until Reach, but has a positional strategy for $B(2N)$ Until Reach.

Proof. The game is represented in Figure 7. It involves one counter and the condition B Until Reach. Starting from v_0, the game moves to v and sets the value of the counter to N. The objective of Eve is to take the edge to the right to F. However, this costs N increments, so if she wants the counter value to remain smaller than N she has to set its value to 0 before taking this edge. She has N options: for $\ell \in \{1, \dots, N\}$, the ℓ^{th} option consists in going to u_ℓ, involving the following actions:

- first, take $N - \ell$ increments,
- then, reset the counter,
- then, take $\ell - 1$ increments, setting the value to $\ell - 1$.

It follows that there is a strategy for Eve to ensure $B(N)$ Until Reach, which consists in going successively through u_N, u_{N-1}, and so on, until u_1, and finally to F. Hence to ensure that the bound is always smaller than N, Eve needs $N+1$ memory states. ∎

Observe that in Figure 7, if we consider the bound $2N$ rather than N, then Eve has a very simple strategy, which consists in going directly to F, using no memory at all. This is a simple example of a trade-off: to ensure the bound N, Eve needs $N+1$ memory states, but to ensure the worse bound $2N$, she has a positional strategy.

We conclude by showing that the uniform boundedness games are finite-memory determined.

Lemma 11 (Finite-memory determinacy for uniform boundedness games)

For all $k \in \mathbb{N}$, for all boundedness games with k counters and initial vertices v_0, there exists a strategy σ ensuring $B(\text{val}(v_0)) \cap \text{Parity}$ using $(\text{val}(v_0) + 1)^k$ memory states.

Proof. Denote $N = \text{val}(v_0)$, and consider the memory structure $\mathcal{M}$ which keeps track of the counter values: the set of memory states is $M = \{0, \ldots, N\}^k \cup \{\perp\}$, the initial memory state 0^k, and the update function μ mimics the counter actions. If a counter value exceeds N, the memory state is updated to $\perp$. We construct the arena $\mathcal{G} \times \mathcal{M}$ equipped with the colouring function $c : E \times M \to \{0, 1\}$ defined by:

$$c(\_, m) = \begin{cases} 1 & \text{if } m = \perp, \\ 0 & \text{otherwise.} \end{cases}$$

The condition $B(N) \cap \text{Parity}$ in $\mathcal{G}$ is equivalent to the condition $\text{Safe} \cap \text{Parity}$ in $\mathcal{G} \times \mathcal{M}$. Since parity games are positionally determined and thanks to Lemma 3 (to handle the additional safety condition), there exists a positional winning strategy in $(\mathcal{G} \times \mathcal{M}, \text{Safe} \cap \text{Parity})$, which induces a winning strategy in $(\mathcal{G}, B(N) \cap \text{Parity})$ using $\mathcal{M}$ as memory structure. Note that since this strategy is winning, it does not make use of the memory state $\perp$, so it uses $(N + 1)^k$ memory states. This concludes. ∎

3.2 Applications to the Theory of Regular Cost Functions

In this section, we show that the LoCo conjecture is the key property for proving that alternating and non-deterministic B-parity automata over infinite trees are equivalent.

We first recall the classical construction for alternating automata without counters, which from an alternating automaton constructs an equivalent non-deterministic automaton. This construction is due to Muller and Schupp [MS87; MS95].

Then, we explain how to extend this construction to the case of alternating B-automata, *i.e.* when adding counters. The LoCo conjecture was introduced to make this construction work.

Recall that a (complete binary) A-labelled tree is a function $t : \{0, 1\}^* \to A$. A branch of t is an infinite word $\pi \in \{0, 1\}^\omega$.

Definition 12 (Alternating automaton over infinite trees)

An *alternating (parity tree) automaton* $\mathcal{A}$ is given by a finite set of states Q divided into existential states (Q_E) and universal state (Q_A), a transition function $\Delta \subseteq Q \times A \times Q \times Q$, an initial state q_0 and a colouring function $\Omega : Q \to \{0, \ldots, d\}$.

We define the semantics of alternating automata using games. An alternating automaton $\mathcal{A}$ and an A-labelled tree t induce the parity game $(\mathcal{G}_{\mathcal{A},t}, \text{Parity})$, that we define now, and we say that t is accepted by $\mathcal{A}$ if, and only if, Eve has a winning strategy in $\mathcal{G}_{\mathcal{A},t}$.

In the acceptance game $(\mathcal{G}_{\mathcal{A},t}, \text{Parity})$, Eve is trying to prove that t is accepted, and Adam tries to disprove her. She is in charge of the existential states, and Adam in charge of the universal states. Together, they build a run of the automaton, which can be seen as a Q-labelled tree. In addition, Adam also chooses the directions; this corresponds to the fact that a run is accepting if all its branches are accepting. Here the acceptance condition is given by a parity condition.

Formally:

$$V = \begin{cases} V_E = \{(q,n) \mid q \in Q_E, n \in \{0,1\}^*\} \\ V_A = \{(q,n) \mid q \in Q_A, n \in \{0,1\}^*\} \cup \{(q_0,q_1,n) \mid q_0,q_1 \in Q, n \in \{0,1\}^*\} \end{cases}$$

$$E = \begin{cases} (q,n) \longrightarrow (q_0,q_1,n) & (q,t(n),q_0,q_1) \in \Delta \\ (q_0,q_1,n) \longrightarrow (q_\ell, n \cdot \ell) & n \in \{0,1\}^*, \ell \in \{0,1\} \end{cases}$$

$$\Omega(q,n) = \Omega(q).$$

The definition of non-deterministic automata can be obtained as the special case of alternating automata where all states are existential. Note that in this case, the games semantics given above reduces to the following: a tree is accepted if there exists a run over this tree such that all branches of the run are accepting.

Theorem 10 (Alternating and non-deterministic automata [MS87; MS95])

For every alternating automaton, there exists an equivalent non-deterministic automaton, *i.e.* accepting the same set of trees.

We sketch the proof of this theorem.

Since parity games are positionally determined, a tree t is accepted by $\mathcal{A}$ if, and only if, Eve has a positional winning strategy in $\mathcal{G}_{\mathcal{A},t}$. Observe that a positional strategy is a function $\sigma : Q \times \{0,1\}^* \to \Delta$, so it can be seen as a $(Q \times \Delta)$-labelled tree. We construct a non-deterministic automaton $\mathcal{C}$ that accepts $(A \times Q \times \Delta)$-labelled trees; a tree (t,σ) is accepted by $\mathcal{C}$ if, and only if, σ is a winning strategy in $\mathcal{G}_{\mathcal{A},t}$.

This set of infinite trees has a special property: it is *branch-wise*, *i.e.* it is the form $\{t \mid \text{for all branches } \pi, t[\pi] \in L\}$, for some set L of infinite words, where $t[\pi]$ is the infinite word obtained by restricting the tree t to the branch π.

In the present case, the language L is ω-regular, so it recognized by some deterministic automaton over infinite words. This allows to construct $\mathcal{C}$ as a deterministic automaton over infinite trees.

Finally, we construct $\mathcal{B}$ as the projection of $\mathcal{C}$ on the second component: a tree t is accepted by $\mathcal{B}$ if, and only if, there exists a tree (t,σ) accepted by $\mathcal{C}$. The automaton $\mathcal{B}$ makes at every transition non-deterministic guesses about the second component σ.

To summarize, the key properties used in the above construction are:

- parity games are positionally determined,
- every ω-regular language is recognized by a deterministic automaton, allowing to construct a deterministic automaton for the branch-wise language over infinite trees.

We now consider B-parity automata, as introduced by Colcombet in the theory of regular cost functions [Col09; Col13b]. An alternating B-parity automaton is an alternating automaton additionally equipped with a set of counters, which are manipulated through the colouring function $\mathbf{c} : \Delta \to \{\varepsilon, i, r\}^k$.

The semantics of a B-parity automaton is given as before through an acceptance game; an alternating B-parity automaton $\mathcal{A}$ and a tree t induce the boundedness game $(\mathcal{G}_{\mathcal{A},t}, B \cap \text{Parity})$, which is defined as before, extended with the colouring function $\mathbf{c}$ for the counters. It follows that $\mathcal{A}$ defines a function from the set of trees to $\mathbb{N} \cup \{\infty\}$: $[\mathcal{A}]\,(t)$ is the smallest N such that Eve wins has a strategy in $\mathcal{G}_{\mathcal{A},t}$ ensuring $B(N) \cap \text{Parity}$.

Recall that the theory of regular cost functions aims at defining boundedness properties; hence functions f, g from the set of trees to $\mathbb{N} \cup \{\infty\}$ are considered up to an equivalence relation, which only preserves boundedness properties:

$$f \approx g \qquad \text{if for all sets of trees } X, f(X) \text{ is bounded } \iff g(X) \text{ is bounded.}$$

Equivalently, $f \approx g$ if there exists a non-decreasing function $\alpha : \mathbb{N} \to \mathbb{N}$ such that $f \leqslant \alpha(g)$ and $g \leqslant \alpha(f)$.

We say that two B-parity automata $\mathcal{A}, \mathcal{B}$ are equivalent if $[\mathcal{A}] \approx [\mathcal{B}]$.

Theorem 11 (The LoCo conjecture and alternating B-parity automata [Col13a])

If the LoCo conjecture holds, then for every alternating B-parity automaton, there exists an equivalent non-deterministic B-parity automaton.

In [Col13a], Colcombet explained how to extend the above construction to B-parity automata:

- the positionality of parity games is replaced by the LoCo conjecture,
- the determinisation over infinite words is replaced by history-deterministic automata.

Note that in this document we only consider the LoCo conjecture for Eve. A similar conjecture can be made for Adam, and a similar construction implies that alternating and non-deterministic S-parity automata are equivalent. Under the assumption that both results are true, this implies that for every cost-MSO formula, there exists an equivalent non-deterministic S-parity automaton, implying the decidability of the boundedness problem for cost-MSO. As shown in [CL08b], this would imply the decidability of the Rabin-Mostowski hierarchy, a long-standing open problem in automata theory.

3.3 A Refined Statement: Structural Properties of the Arenas

In the previous subsection, we showed the motivations for introducing the LoCo conjecture. A closer inspection reveals that the LoCo conjecture is not used for general arenas, but only for arenas underlying the acceptance game of a tree by an alternating automaton.

In this subsection, we refine the LoCo conjecture by taking this remark into account.

First of all, the arenas underlying acceptance games are chronological:

Definition 13 (Chronological arena)

An arena is *chronological* if there exists a function $r : V \to \mathbb{N}$ which increases by one on every edge: for all $(v, v') \in E$, $r(v') = r(v) + 1$.

One can be even more precise. The arenas underlying acceptance games for alternating automata over infinite words are word arenas:

Definition 14 (Word arena)

An arena is a *word arena of width* W if it is chronological, and for all $i \in \mathbb{N}$, the set $\{v \in V \mid r(v) = i\}$ has cardinal at most W.

In the previous subsection, we defined complete labelled trees. We here consider partial unlabeled trees, which are given by $T \subseteq \{0,1\}^*$, where T is prefix-closed and non-empty.

A (finite or infinite) branch π is a word in $\{0,1\}^*$ or $\{0,1\}^\omega$. We say that π is a branch of the tree T if $\pi \subseteq T$ (or every prefix of π belongs to T when π is infinite) and π is maximal satisfying this property.

Definition 15 (Thin tree [BIS13])

A tree is called *thin* if it has only countably many branches. Equivalently, a tree is thin if, and only if, there exists a ranking function rank associating a countable ordinal to every node, such that:

1. if n' is a child of n, then $\text{rank}(n') \leqslant \text{rank}(n)$,
2. the set of nodes having the same rank is either a single node or an infinite branch of the tree.

The arenas underlying acceptance games for alternating automata over infinite trees are tree arenas:

Definition 16 (Tree and thin tree arena)

An arena is *a tree arena of width* W if there exists a function $R : V \to \{0,1\}^*$ such that:

1. for all $n \in \{0,1\}^*$, the set $\{v \in V \mid R(v) = n\}$ has cardinal at most W.
2. for all $(v, v') \in E$, we have $R(v') = R(v) \cdot \ell$ for some $\ell \in \{0,1\}$.

It is a thin tree arena if $R(V)$ is a thin tree.

To avoid a possible confusion: in a tree arena, "vertices" refers to the arena and "nodes" to $R(V)$, so if the arena has width W, then a node is a bundle of at most W vertices.

Observe that in both word and tree arenas, the width corresponds to the size of the automaton. The following refined statement of the LoCo conjecture says that both the trade-off function and the number of memory states can depend on the width of the arena.

Conjecture 2 (LoCo conjecture over tree arenas)

For all $k, d, W \in \mathbb{N}$, there exists mem $\in \mathbb{N}$ and $\alpha : \mathbb{N} \to \mathbb{N}$ such that for all boundedness games with k counters, $d + 1$ colors and initial vertex v_0 played over a tree arena of width W, there exists a strategy σ using mem memory states ensuring $B(\alpha(\mathrm{val}(v_0))) \cap \mathrm{Parity}$.

Note that both functions mem and α depend on k, d and W. In Section 6, we will make this dependence explicit, by considering mem $: \mathbb{N}^2 \to \mathbb{N}$ and $\alpha : \mathbb{N}^4 \to \mathbb{N}$.

4 Examples

In this section, we consider special cases of the LoCo conjecture, and gather results from the literature, new results and simple examples.

We start with B games in Subsection 4.1. We then consider two restrictions that make the LoCo conjecture much simpler: finite arenas in Subsection 4.2, and temporal arenas in Subsection 4.3.

We continue with B-reachability games in Subsection 4.4, then B-Büchi games in Subsection 4.5. This case has been solved by Vanden Boom in [Van11] using a slicing technique, and is crucial in the decidability proofs of both weak cost-MSO [Van11] and quasi weak cost-MSO [BCKPV14].

The next step is B-CoBüchi games, considered in Subsection 4.6.

To conclude, we present in Subsection 4.7 a result obtained in collaboration with Sophie Pinchinat and Olivier Serre and published in [FPS13], where we used ideas from the slicing technique to obtain a positionality result for stochastic games over infinite arenas.

4.1 The B games

The case of B games, with only counters, is well understood:

Theorem 12 (The LoCo conjecture for B games [CL08a; Col13a])

For all B games with k counters and initial vertex v_0, Eve has a strategy ensuring $B(k^k \cdot \mathrm{val}(v_0)^{2k})$ using $k!$ memory states.

This result is obtained as the combination of two ideas:

- an adaptation of the Latest Appearance Record (LAR), reducing k counters into k hierarchical counters [Col13a],
- a reduction from k hierarchical counter into one [CL08a],

The LAR construction implies a blow-up of $k!$ memory states, and the bounds the value by $k \cdot \mathrm{val}(v_0)^k$. The second construction also implies a distortion of the value, from $k \cdot \mathrm{val}(v_0)^k$ to $k^k \cdot \mathrm{val}(v_0)^{2k}$.

The LAR construction extends to B-parity conditions *mutatis mutandis*, allowing to assume without loss of generality that the counters are hierarchical. The second construction also has a general flavour, and extends to many cases. This is why when considering examples and counter-examples, we usually have only one counter instead of k: the interactions between the counters are not where the difficulty lies. Indeed, the difficulty in understanding boundedness games is in the interactions between the counters and the parity condition.

4.2 The Boundedness Games over Finite Arenas

The case of boundedness games over finite arenas is the very special, as in this setting boundedness games are somehow equivalent to ω-regular games.

Theorem 13 (The LoCo conjecture for boundedness games over finite arenas)

For all boundedness games with k counters played over a finite arena, Eve has a strategy ensuring $B(n \cdot 2^k) \cap \text{Parity}$ with 2^k memory states, where $n = |V|$.

Furthermore, if $\text{val}(v_0) < \infty$, then $\text{val}(v_0) \leqslant n \cdot 2^k$.

We sketch the proof. The idea is to approximate the condition B by the following ω-regular condition: for every counter, if it is incremented infinitely many times, then it is reset infinitely many times.

This is an over-approximation: if the condition B is satisfied, then the latter is satisfied as well. Furthermore, ω-regular games are bounded-memory determined; more specifically here, there exists a strategy using 2^k memory states, as the condition approximating $B \cap$ Parity is a conjunction of a Streett condition with k pairs together with a parity condition. A simple pumping argument shows that this strategy also ensures $B(n \cdot 2^k) \cap \text{Parity}$.

Note that in this result, the trade-off function depends on the number of vertices in the arena. We will see in Section 5 that this cannot be improved, as the LoCo conjecture does not hold, even over finite arenas.

The same proof yields the following result for the non-uniform case:

Theorem 14 (Non-uniform boundedness games over finite arenas)

Non-uniform boundedness games over finite arenas are bounded-memory determined.

4.3 The Temporal Boundedness Games

The case of temporal boundedness games is a folklore result in the regular cost function community. The temporal regular cost functions have been introduced in [CKL10], and correspond to boundedness games where the counters can be incremented and reset, but not left unchanged.

Formally, in a temporal boundedness game the colouring function for the counter actions is $\mathbf{c} : E \to \{i, r\}^k$.

Theorem 15 (The LoCo conjecture for temporal boundedness games)

For all temporal boundedness games played over a chronological arena, Eve has a strategy ensuring $B(2 \cdot \text{val}(v_0)) \cap \text{Parity}$ using 2^k memory states.

We sketch the proof. Let $N = \text{val}(v_0)$. The idea is to approximate the condition $B(N)$ by a safety condition; using the fact that the arena is chronological, we slice the arena every N steps, and compose the arena with a memory structure of size 2^k checking for each counter whether it has been reset in the current slice.

Consider the following ω-regular condition: each counter is reset in each slice, and the parity condition is satisfied. Observe that it is positionally determined, as a conjunction of a safety and a parity condition. A strategy ensuring $B(N) \cap$ Parity also ensures this condition. Conversely, a strategy ensuring this condition ensures $B(2N) \cap$ Parity.

This idea of slicing does not extend to the general case where counters can be left unchanged. However, the slicing technique has a similar flavour, see Subsection 4.5.

4.4 The B-Reachability Games

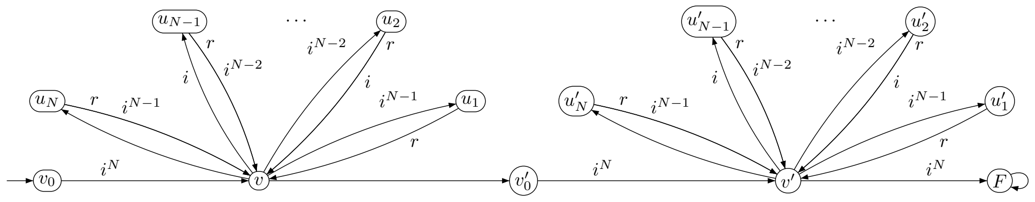


Figure 8: Eve has a 2 memory states strategy ensuring $B(2N)$ Until Reach, but no positional strategy for this.

For all $N \in \mathbb{N}$, we construct a B-reachability game, which is a variant of the one presented in Figure 7 in Subsection 3.1. Recall that in the original game:

- $\text{val}(v_0) = N$, so Eve has a strategy ensuring $B(N)$ Until Reach using $N+1$ memory states,
- Eve has a positional strategy ensuring $B(2N)$ Until Reach, which consists in going directly to F.

The present game repeats the original game twice; it is represented in Figure 8. In this game, the positional strategy cannot be repeated, as it would ensure $B(3N)$ Until Reach. Hence a second memory state is necessary to reset the value. The wisest choice is then to pick the option $u_{N/2}$.

- $\text{val}(v_0) = N$, so Eve has a strategy ensuring $B(N)$ Until Reach using $N+1$ memory states,
- Eve has a strategy ensuring $B(2N)$ Until Reach using 2 memory states,
- no positional strategy ensures $B(2N)$ Until Reach.

The bound $3N$ can be deemed acceptable; a simple extension of this example consists in repeating K times the original game, yielding a game where:

- $\text{val}(v_0) = N$, so Eve has a strategy ensuring $B(N)$ Until Reach using $N+1$ memory states,
- Eve has a strategy ensuring $B(2N)$ Until Reach using 2 memory states,

- no positional strategy ensures $B(K \cdot N)$ Until Reach.

We can go further, and repeat the original game infinitely many time, yielding the B-Büchi game described in the next subsection.

4.5 The B-Büchi Games

We consider now the case of B-Büchi games. The LoCo conjecture holds for B-Büchi games over chronological arenas, and this is so far the best result known about the LoCo conjecture over chronological arenas.

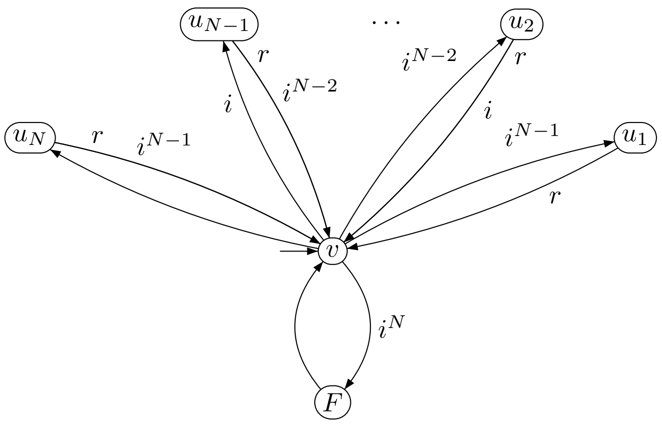


Figure 9: Eve has a 2 memory states strategy ensuring $B(2N) \cap$ Büchi, but no positional strategy for $B \cap$ Büchi.

We start with the game represented in Figure 9, which is a variant of Figure 7 in Subsection 3.1. In this game:

- Eve has a strategy ensuring $B(N) \cap$ Büchi using $N+1$ memory states, which is the diligent strategy that goes through each option $u_N, \dots, u_1$ in between each visit to a Büchi vertex,

- Eve has a strategy ensuring $B(2N) \cap$ Büchi using 2 memory states, which is the careless strategy that alternates between $u_{N/2}$ and the Büchi vertex,

- no positional strategy ensures $B \cap$ Büchi.

The following result uses an interesting slicing technique, which is reminiscent of the breakpoint construction of Miyano and Hayashi [MH84].

Theorem 16 (The LoCo for B-Büchi games over chronological arenas [Van11])

For all B-Büchi games over a chronological arena with k counter and initial vertices v_0, Eve has a strategy ensuring $B(k^k \text{val}(v_0)^{2k}) \cap$ Büchi using $2k!$ memory states.

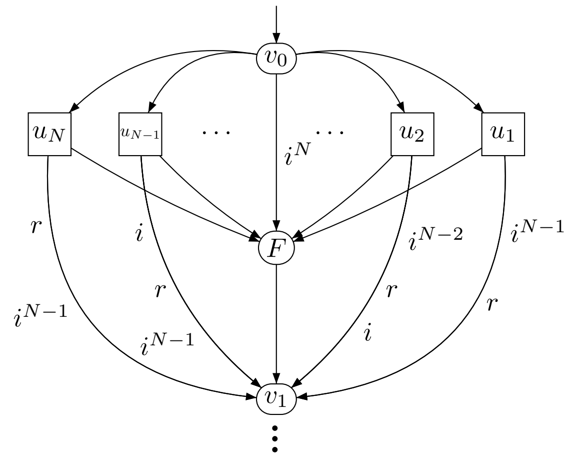


Figure 10: Eve has a 2 memory states strategy ensuring $B(N) \cap$ Büchi.

We illustrate the slicing technique on an example; the proof ideas will be slightly generalised later, in the proof of the LoCo conjecture for boundedness games over word arenas, to remove the least important color when it is odd, see Subsection 6.1.

Consider the game represented in Figure 10, which is yet another variant of Figure 7 in Subsection 3.1. On a first approximation, it is just an unravelling of the game presented in Figure 9: Eve tries to see infinitely many times Büchi vertices, but each time she does so she has to pay N increments. To keep the counter value below the value N, she can use one of the N options $u_N, \ldots, u_1$; the ℓ^{th} option is available to her only if the counter value is at most ℓ, and sets the value to $\ell - 1$.

The new feature of this game is that when Eve chooses such an option, Adam can decide not to grant it, and instead to go directly to the Büchi vertex, without any action on the counter. Although this is fine for Eve as she visits a Büchi vertex, she has to take into account that the counter value has not been decremented.

As before, there is a diligent strategy ensuring $B(N) \cap$ Büchi using $N + 1$ memory states, and a simple strategy ensuring $B(2N) \cap$ Büchi using 2 memory states, which chooses either $u_{N/2}$ or the Büchi vertex. More surprisingly so, there is a strategy ensuring $B(N) \cap$ Büchi using 2 memory states, which is constructed by the slicing technique, and that we describe now.

Fix a strategy ensuring $B(N) \cap$ Büchi, let us say the diligent one. The slicing decomposes the arena horizontally; the slices are chosen so that the strategy ensures to reach a Büchi vertex in each slice. In our case, the slices consist in $N + 1$ consecutive steps, since the diligent strategy chooses in turn $u_N, u_{N-1}, \cdots, u_1$, the Büchi vertex, and then starts over, so it visits a Büchi vertex every $N + 1$ steps.

The slicing techniques constructs two positional strategies for playing in each slice:

- the first is the waiting strategy, that always plays u_N,
- the second is the active strategy, that plays $u_N, u_{N-1}, \cdots, u_1$ and the Büchi vertex. Note that since each slice contains $N + 1$ steps, if the active strategy starts at the beginning of a slice, it will visit a Büchi vertex before the next slice.

The 2 memory states strategy constructed by the slicing technique alternates between the waiting and the active strategy, as follows:

- when reaching a new slice, it switches to the active strategy,
- if at some point Adam does not grant an action and imposes a visit to the Büchi vertex, then it switches to the waiting strategy until a new slice.

This strategy ensures $B(N) \cap$ Büchi using 2 memory states.

4.6 The B-CoBüchi Games

The case of B-CoBüchi games is the first case for which we do not know whether the LoCo conjecture holds over chronological arenas, or even tree arenas. Here we give two examples:

- A B-CoBüchi game over a word arena where Eve has no bounded-memory strategy to ensure $B(\mathrm{val}(v_0)) \cap$ CoBüchi,
- A B-CoBüchi game over a tree arena where Eve has no positional strategy to ensure $B \cap$ CoBüchi.

In other words, the first example shows that a trade-off function is necessary for B-CoBüchi games over word arenas, and the second example shows that 2 memory states is necessary for B-CoBüchi games over tree arenas.

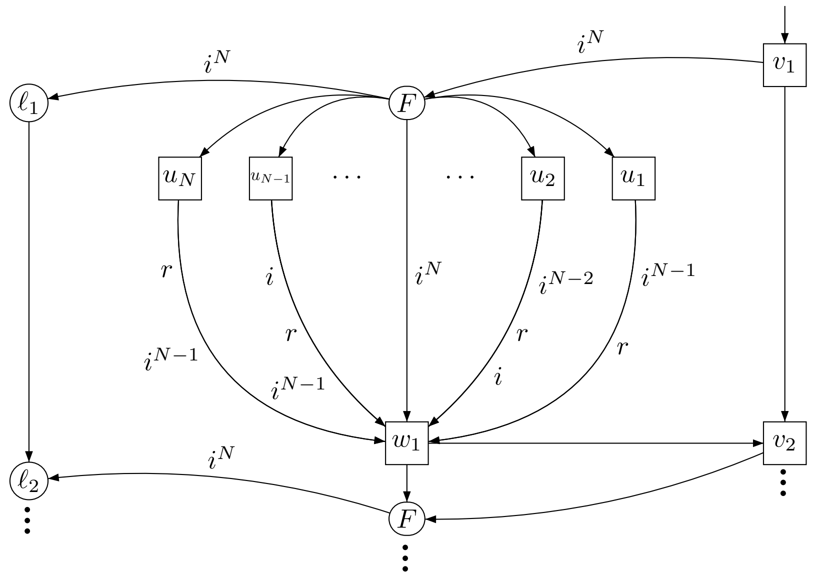


Figure 11: Eve has a 2 memory states strategy ensuring $B(2N) \cap$ CoBüchi(F), but no strategy using less than $N+1$ memory states ensures $B(N) \cap$ CoBüchi(F).

We start with the game represented in Figure 11. In this game:

- $\mathrm{val}(v_0) = N$, so Eve has a strategy ensuring $B(N) \cap$ CoBüchi using $N+1$ memory states,
- Eve has a positional strategy ensuring $B(2 \cdot N) \cap$ CoBüchi,

- no strategy using less than $N + 1$ memory states ensures $B(N) \cap$ CoBüchi.

At a very high level, this game is again a variant of Figure 7 in Subsection 3.1: the counter value is initialized at N, and Eve has to go through the N options $u_N, u_{N-1}, \ldots, u_1$ to decrease its value one by one, until the value reaches 0 and she can pay N increments to go to the left, where nothing happens anymore.

The implementation of this idea is different, as it here uses a CoBüchi condition instead of the Büchi condition in the previous section. Note that although it has not been presented this way, one can construct a word arena simulating Figure 11, of constant width (namely, 5).

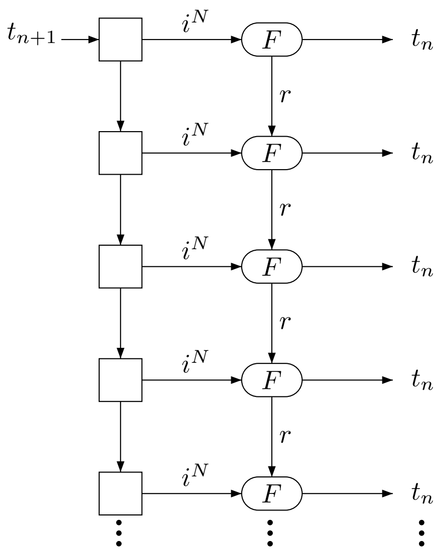


Figure 12: Eve has a 2 memory states strategy ensuring $B(N) \cap$CoBüchi, but no positional strategy ensures $B \cap$ CoBüchi.

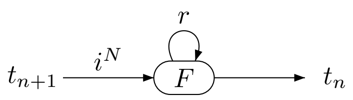


Figure 13: A behaviourally equivalent arena.

We now look at the following game, played over a tree arena. First, Adam chooses a value n and goes to the tree arena t_n. The tree arena t_{n+1} is represented in Figure 12; it leads to t_n. In the tree arena t_0, the play stops. Although this is again not clear from the representation in Figure 12, one can construct a tree arena simulating Figure 12, of constant width (namely, 3).

For the sake of explanation, we refer to Figure 13, which is behaviourally equivalent to Figure 12. What happens between t_{n+1} and t_n is pretty simple: Eve pays N increments. From u_n, she can loop and reset the value, and proceeds to t_n.

The reasonable strategy is of course to first reset, and then proceeds; this requires 2 memory states. A positional strategy would either reset forever, or never, in which case the counter value would grow unbounded.

To summarize:

- $\text{val}(v_0) = N$, and Eve has a strategy ensuring $B(N) \cap$ CoBüchi using 2 memory states,

- no positional strategy ensures $B \cap \text{CoBüchi}$.

4.7 Application of the Slicing Technique for Stochastic Games

In this subsection, we report on a result obtained in collaboration with Sophie Pinchinat and Olivier Serre and published in [FPS13]. Note that the paper [FPS13] actually deals with emptiness checking of alternating automata, and in particular introduces the class of alternating qualitative automata; the corollary of the positionality result that we present here is the decidability of the emptiness problem for alternating qualitative Büchi automata.

We do not develop this further, and focus on the positionality result, as it makes use of the slicing technique, discussed in Subsection 4.5.

Before stating the result, we define stochastic games:

Definition 17 (Stochastic game)

A stochastic arena is an arena whose vertex set is divided into three subsets: vertices controlled by Eve (V_E), vertices controlled by Adam (V_A), and random vertices (V_R). It additionally features a probabilistic transition $\Delta : V_R \to \mathcal{D}(V)$, where $\mathcal{D}(V)$ is the set of distributions over V (see Subsection 1.1 for the definition of distributions).

By convention, the set E of edges is such that for every $v \in V_R$ and $v' \in V$, we have $(v, v') \in E$ if, and only if, $\Delta(v)(v') > 0$.

A pair of strategies σ for Eve and τ for Adam induces a set of infinite plays, which is equipped with the appropriate probability measure. For a measurable event W, we denote $P_{\sigma,\tau}(W)$ its probability.

A strategy σ for Eve is said almost-surely winning for the condition W if for all strategies τ of Adam, we have $P_{\sigma,\tau}(W) = 1$.

Note that we did not extend the definition of strategies to include *randomized strategies*, which are strategies $\sigma : E^* \cdot V_E \to \mathcal{D}(E)$. Since here *pure strategies* are sufficient, *i.e.* strategies $\sigma : E^* \cdot V_E \to E$, we will consider only them.

Theorem 17 (Positionality for Stochastic Büchi Games with Almost-Sure Semantics)

For all stochastic Büchi games played over a chronological arena with finite out-degree, for all initial vertices, if Eve has an almost-surely winning strategy, then she has a positional almost-surely winning strategy.

We will use the following result for finite arenas. We state it here in a rather weak form (with the chronological assumption), as it can be easily proved by a backward induction, whereas the proof for general finite arenas is more involved. We refer to [Con92] for the original proof, and to [Kuč11] for a nice survey.

Lemma 12 (Positionality for Stochastic Reachability Games over Finite Arenas)

For all stochastic reachability games played over a chronological and finite arena, if for every vertex v_0 in a given subset X of vertices Eve has strategy ensuring to win with probability at least $\frac{1}{2}$ from v_0, then she has a uniform positional strategy ensuring to win with probability at least $\frac{1}{2}$ from X.

We first sketch the proof. The main idea is to note that if Eve can ensure to reach a Büchi vertex with probability 1 from some initial vertex, then there exists a bound k such that she can ensure to reach a Büchi vertex with probability at least $\frac{1}{2}$ within k steps against *any* strategy of Adam.

This allows to "slice" the arena into infinitely many disjoint finite arenas: in each slice Eve plays to reach a Büchi vertex with probability at least $\frac{1}{2}$. Since each slice forms a finite subarena, Eve can use positional strategies there thanks to Lemma 12.

The resulting strategy consists in playing in turns the above positional strategies; since each slice gives a probability to reach a Büchi vertex of at least $\frac{1}{2}$ before proceeding to the next, the probability to reach infinitely many Büchi vertices is 1.

Proof. We fix an initial vertex and an almost-surely winning strategy σ from there. Note that σ is almost-surely winning from all reachable vertices; from now on we restrict to such vertices.

We first claim that for every vertex v, there exists $k \in \mathbb{N}$ such that for all strategies τ of Adam we have $P_{\sigma,\tau}(V^{\leqslant k} \cdot F \cdot V^\omega) \geqslant \frac{1}{2}$ from v, *i.e.* σ ensures to reach a Büchi vertex within k steps with probability at least $\frac{1}{2}$ from v.

Towards contradiction, assume that such a k does not exist. Hence, for each k there exists a strategy τ_k such that $P_{\sigma,\tau_k}(V^{\leqslant k} \cdot F \cdot V^\omega) < \frac{1}{2}$. From the sequence $(\tau_k)_{k\in\mathbb{N}}$ we extract a strategy τ_∞ that is consistent, for any $k \in \mathbb{N}$, with infinitely many τ_h on its k first moves. Note that this is possible since the arena has finite out-degree.

Consider the pair of strategies σ and τ_∞. For $k \in \mathbb{N}$, there exists $h \geqslant k$ such that

$$P_{\sigma,\tau_\infty}(V^{\leqslant k} \cdot F \cdot V^\omega) = P_{\sigma,\tau_h}(V^{\leqslant k} \cdot F \cdot V^\omega) \leqslant P_{\sigma,\tau_h}(V^{\leqslant h} \cdot F \cdot V^\omega) \leqslant \frac{1}{2}.$$

Since $V^* \cdot F \cdot V^\omega = \bigcup_{k\in\mathbb{N}} V^{\leqslant k} \cdot F \cdot V^\omega$, and the sequence $(V^{\leqslant k} \cdot F \cdot V^\omega)_{k\in\mathbb{N}}$ is increasing for set inclusion, this implies that $P_{\sigma,\tau_\infty}(V^* \cdot F \cdot V^\omega) = \lim_k P_{\sigma,\tau_\infty}(V^{\leqslant k} \cdot F \cdot V^\omega) \leqslant 1$, which leads a contradiction with σ being almost-surely winning. This proves the claim.

For $k < k'$, define $\mathcal{G}_{[k,k']}$ the arena induced by $\mathcal{G}$ restricted to vertices of rank in $[k, k']$. Since $\mathcal{G}$ has finite out-degree, there are finitely many vertices of rank in $[k, k']$, hence $\mathcal{G}_{[k,k']}$ is finite.

We define inductively a sequence of ranks $(k_i)_{i\in\mathbb{N}}$ together with a sequence of strategies $(\sigma_{[k_i,k_{i+1}[})_{i\in\mathbb{N}}$ such that for all $i \in \mathbb{N}$, $\sigma_{[k_i,k_{i+1}[}$ is a positional strategy, defined on all vertices of rank $[k_i, k_{i+1}[$, such that from all vertices of rank k_i, for all strategies τ of Adam, we have $P_{\sigma_{[k_i,k_{i+1}[},\tau}(V^{\leqslant \ell} \cdot F \cdot V^\omega) \geqslant \frac{1}{2}$, where $\ell = k_{i+1} - k_i$.

Set $k_0 = 0$. Assume the first i ranks and strategies are defined. For each vertex of rank k_i, the above claim implies the existence of a bound; since there are finitely many such vertices, we can consider the maximum of those bounds, and denote it by k_{i+1}. By construction, from all vertices of rank k_i, for all strategies τ of Adam, we have $P_{\sigma_{[k_i,k_{i+1}[},\tau}(V^{\leqslant \ell} \cdot F \cdot V^\omega) \geqslant \frac{1}{2}$, where $\ell = k_{i+1} - k_i$. In other words, Eve has a strategy in the reachability game played over the chronological and finite arena $\mathcal{G}_{[k_i,k_{i+1}]}$ ensuring to win with probability at least $\frac{1}{2}$, so thanks to Lemma 12 there exists a uniform positional strategy ensuring to reach a Büchi vertex with probability at least $\frac{1}{2}$, denote it $\sigma_{[k_i,k_{i+1}[}$. This concludes the inductive construction.

The arenas $\mathcal{G}_{[k_i,k_{i+1}]}$ are called slices.

Now define σ_∞ as the disjoint union of the strategies $\sigma_{[k_i,k_{i+1}[}$. This is a positional strategy; we argue that it is almost-surely winning. Indeed, since σ_∞ ensures that going through any slice, a Büchi vertex will be visited with probability at least $\frac{1}{2}$, the Borel-Cantelli Lemma implies that infinitely many Büchi vertices will be visited with probability 1. ∎

5 Counter Examples

In this section, we show that the LoCo conjecture does not hold, even for finite arenas. This is a joint work with Florian Horn, Denis Kuperberg and Michał Skrzypczak [FHKS15].

Theorem 18 (The LoCo conjecture does not hold)

For all K, for all N, there exists a B-reachability game played over a finite arena with one counter and an initial vertex such that:

- there exists a 3^K memory states strategy ensuring $B(K(K+3))$ Until Reach,
- no $K+1$ memory states strategy ensures $B(N)$ Until Reach.

We proceed in two steps. The first is an example giving a lower bound of 3 explained in Subsection 5.1, and the second is a nesting of this first example, given in Subsection 5.2.

We then show how to adapt the example to prove that non-uniform boundedness games are not finite-memory determined, in Subsection 5.3.

5.1 A first lower bound of 3

We start with a first lower bound of 3. The arena $\mathcal{G}_1$ is represented in Figure 14. The condition is B Until Reach. In this game, Eve is torn between going to the right to reach F, which implies incrementing the counter, and going to the left, to reset the counter. The actions of Eve from the vertex u_n are:

- *increment*, and go one step to the right, to v_{n-1},
- *reset*, and go two steps to the left, to v_{n+2}.

The actions of Adam from the vertex v_n are:

- *play*, and go down to u_n,
- *skip*, and go to v_{n-1}.

Formally (note that the colouring functions $\mathbf{c} : E \to \{\varepsilon, i, r\}$ and $c : V \to \{0,1\}$ are definitely together with the graph here):

$$V = \begin{cases} V_E = \{u_n \mid n \in \mathbb{N}\} \cup \{F\} \\ V_A = \{v_n \mid n \in \mathbb{N}\} \end{cases}$$

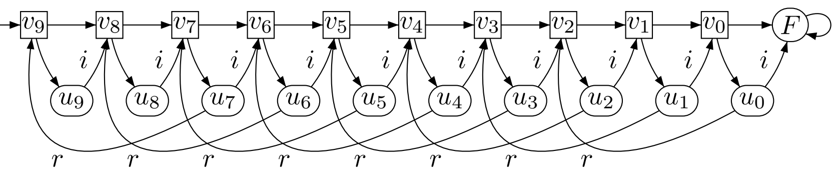


Figure 14: Part of the game $(\mathcal{G}_1, B \text{ Until Reach})$, where Eve needs 3 memory states.

$$E = \begin{cases} & \{v_{n+1} \longrightarrow v_n \mid n \in \mathbb{N}\} \\ \cup & \{v_n \longrightarrow u_n \mid n \in \mathbb{N}\} \\ \cup & \{u_{n+1} \xrightarrow{i} v_n \mid n \in \mathbb{N}\} \cup \{u_0 \xrightarrow{i} F\} \\ \cup & \{u_n \xrightarrow{r} v_{n+2} \mid n \in \mathbb{N}\} \end{cases}$$

Proposition 1 (A lower bound of 3)

In $(\mathcal{G}_1, B \text{ Until Reach})$:

- Eve has a 4 memory states strategy ensuring $B(3)$ Until Reach,
- Eve has a 3 memory states strategy ensuring $B(4)$ Until Reach,
- For all N, no 2 memory states strategy ensures $B(N)$ Until Reach from v_N.

Proof. The first item follows from Lemma 11. However, to illustrate the properties of the game $\mathcal{G}_1$ we will provide a concrete strategy with 4 memory states that ensures $B(3)$ Until Reach. The memory states are i_1, i_2, i_3 and r, linearly ordered by $i_1 < i_2 < i_3 < r$. With the memory states i_1, i_2 and i_3, the strategy chooses to increment, and updates its memory state to the next memory state. With the memory state r, the strategy chooses to reset, and updates its memory state to i_1. This strategy satisfies a simple invariant: it always resets to the right of the previous reset, if any.

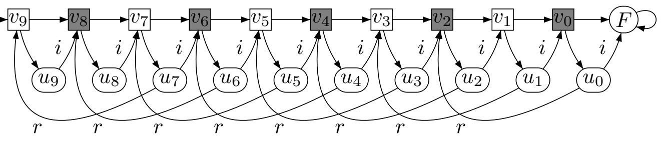


Figure 15: Illustration of the 3 memory states strategy in $\mathcal{G}_1$.

We show how to save one memory state, at the price of increasing the bound by one: we construct a 3 memory states strategy ensuring $B(4)$ Until Reach. The idea, as represented in Figure 15, is to color every second vertex and to use this information to track progress. The 3 memory states are called i, j and r. The update is as follows: the memory state is unchanged in uncoloured (white) states, and switches from i and j and from j to r on gray states. The strategy is as follows: in the two memory states i and j, Eve chooses to increment, and in r she chooses to reset. As for the previous strategy, this strategy ensures that it always resets to the right of the previous reset, if any.

We now show that 2 memory states is not enough. Assume towards contradiction that there exists a 2 memory states strategy ensuring $B(N)$ Until Reach from v_{2N}, for some N, using the memory structure $\mathcal{M}$.

We first argue that without loss of generality we can assume that the strategy σ is normalized, *i.e.* satisfies the following three properties:

1. for all $n \leqslant 2N$, there is at least one memory state that chooses increment from u_n,
2. for all $n \leqslant 2N$ but at most N of them, there is at least one memory state that chooses reset from u_n,
3. no play from (v_{2N}, m_0) consistent with σ comes back to v_{2N}.

Indeed:

1. Assume towards contradiction that this is not the case, then there exists n such that Eve resets from u_n with both memory states; Adam can loop around this u_n, contradicting that σ ensures to reach F.
2. Assume towards contradiction that there are at least $N+1$ vertices u_n from which Eve increments from u_n with both memory states; Adam can force $N+1$ increments without a reset, contradicting that σ ensures $B(N)$.
3. For m and m' two memory states, we say that $m < m'$ if there exists a play from (v_{2N}, m') consistent with σ which reaches (v_{2N}, m). Since σ ensures to reach F, the graph induced by $<$ is acyclic. We can take as initial memory state from v_{2N} the smallest memory state which is smaller than or equal to m_0.

We fix the strategy of Adam which skips if, and only if, both memory states of σ choose to increment. Consider the play from v_{2N} consistent with σ and this strategy of Adam. This means that for all vertices u_n that are reached, there is one memory state that resets, and one that increments. Since σ ensures $B(N)$ and there are at most N positions skipped, at some point Eve chooses to reset. From there two scenarios are possible:

- Either Eve keeps resetting until she reaches v_{2N}, contradicting that σ is normalized,
- Or she starts incrementing again, which means that she uses the same memory state than she did before the reset, implying that there is a loop, contradicting that σ ensures to reach F.

∎

5.2 General lower bound

We now push the example above further.

A first approach is to modify $\mathcal{G}_1$ by increasing the length of the resets, going ℓ steps to the left rather than only 2. However, this does not give a better lower bound: there exists a 3 memory states strategy in this modified game that ensures twice the value, following the same ideas as presented above.

We construct $\mathcal{G}_{K,N}$, a nesting of $\mathcal{G}_1$ with K levels. Unlike $\mathcal{G}_1$, it is finite, as we only keep a long enough “suffix”. In Figure 16, we represented the interaction between two levels.

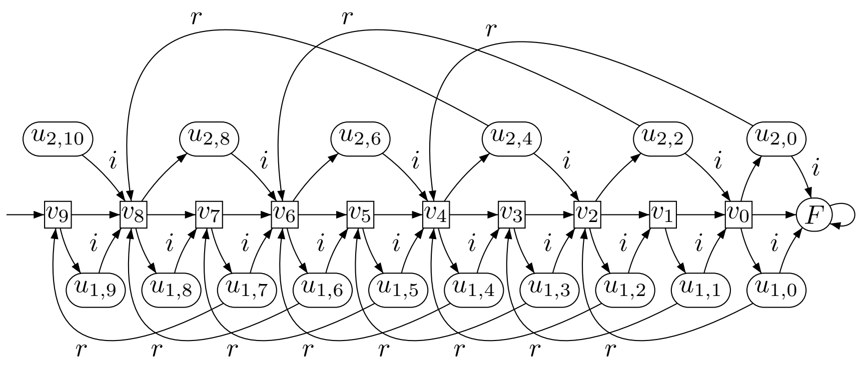


Figure 16: The game with two levels.

Roughly speaking, the two levels are independent, so we play both games at the same time. Those two games use different timelines. For instance, in Figure 16, the bottom level is based on $(+1, -2)$ (an increment goes one step to the right, a reset two steps to the left), and the top level is based on $(+2, -4)$. This difference in timeline ensures that a strategy for Eve needs to take care somehow independently of each level, ensuring that the number of memory states depends on the number of levels.

To give the formal definition of $\mathcal{G}_{K,N}$, we need two functions, $d(K, N) = (N + 1)^{K-1}$ and

$$n(K+1, N) = \begin{cases} 2N & \text{if } K = 0, \\ (N+1)^{K+1} + (N+1) \cdot n(K, N) & \text{otherwise.} \end{cases}$$

We now define $\mathcal{G}_{K,N}$ (note that the colouring functions $\mathbf{c} : E \to \{\varepsilon, i, r\}$ and $c : V \to \{0, 1\}$ are definitely together with the graph here):

$$V = \begin{cases} V_E = \{u_{p,n} \mid p \in \{1, \ldots, K\}, n \in \{0, \ldots, n(K, N)\}\} \cup \{F\} \\ V_A = \{v_n \mid n \in \{0, \ldots, n(K, N)\}\} \end{cases}$$

$$E = \begin{cases} \quad \{v_{n+1} \longrightarrow v_n \mid n\} \\ \cup \quad \{v_n \longrightarrow u_{p,n} \mid p, n\} \\ \cup \quad \{u_{p,n+d(p,N)} \xrightarrow{i} v_n \mid p, n\} \\ \cup \quad \{u_{p,0} \xrightarrow{i} F \mid p\} \\ \cup \quad \{u_{p,n} \xrightarrow{r} v_{n+(p+1) \cdot d(p,N)} \mid p, n\} \end{cases}$$

Observe that $\mathcal{G}_{1,N}$ is the "suffix" of length $n(1, N)$ of $\mathcal{G}_1$, for all N.

Proposition 2 (General lower bound)

In $(\mathcal{G}_{K,N}, B \text{ Until Reach})$:

- Eve has a 3^K memory states strategy ensuring $B(K(K + 3))$ Until Reach,
- No $K + 1$ memory states strategy ensures $B(N)$ Until Reach from $v_{n(K,N)}$.

Proof. We start by constructing a strategy with 3^K memory states ensuring $B(K(K + 3))$ Until Reach. To this end, we construct for the pth level a strategy with 3 memory states ensuring $B(2(p+1))$ Until Reach, using the same ideas as for $\mathcal{G}_1$, colouring every $(p+1)\cdot d(p,N)$ vertices. Now we construct the general strategy by playing independently in each copy, except that when a reset is taken, all memory structures update to the (initial) memory state i. This way, it ensures that it always resets to the right of the previous reset, if any. It uses 3^K memory states, and ensures $B(\sum_{p=1}^{K} 2\cdot(p+1))$ Until Reach, *i.e.* $B(K(K+3))$ Until Reach.

We now show that $K+1$ memory states is not enough. We proceed by induction on K. The case $K=1$ follows from Proposition 1.

Consider a strategy ensuring $B(N)$ Until Reach from $v_{n(K+1,N)}$ in $\mathcal{G}_{K+1,N}$, for some N, using the memory structure $\mathcal{M}$. We will prove that it has at least $K+2$ memory states. To this end, we will show that it induces a strategy ensuring $B(N)$ Until Reach in $\mathcal{G}_{K,N}$, which uses one less memory state. The induction hypothesis will conclude.

First of all, for the sake of readability we will use $n_{K+1} = n(K+1,N)$, $n_K = n(K,N)$ and $d = d(K+1,N)$.

Consider the $N+1$ plays, for $k \in \{0,\dots,N\}$, where from $v_{n_{K+1}-k\cdot(n_K+d)}$ Adam skips n_K times and then plays:

$$v_{n_{K+1}-k\cdot(n_K+d)} \xrightarrow{\text{skip}} v_{n_{K+1}-k\cdot(n_K+d)-n_K} \xrightarrow{\text{play}} u_{K+1,n_{K+1}-k\cdot(n_K+d)-n_K}.$$

(Note that this is well defined because by definition, $n_{K+1}-(N+1)\cdot(n_K+d)=0$.) We argue that there exists $k\in\{0,\dots,N\}$ and a memory state $m \in M$ such that if σ starts the k^{th} play with the memory state m, then σ resets in $u_{K+1,n_{K+1}-k\cdot(n_K+d)-n_K}$. Assume towards contradiction that this is not the case, then by concatenating all plays with the matching memory state we get a play starting from $v_{n_{K+1}}$ with the memory state m_0, which is consistent with σ and contains $N+1$ increments without resets, contradicting that σ ensures $B(N)$.

Let $k\in\{0,\dots,N\}$ and $m\in M$ satisfying the above property: denote $n = n_{K+1}-k\cdot(n_K+d)$, in the play from (v_n,m) where Adam skips n_K times and then plays, leading to $u_{K+1,n-n_K}$, the strategy σ resets. Up to renaming, we can assume that the memory state assumed along this play is always the same, denoted m. Observe that resetting from $u_{K+1,n-n_K}$ leads to the left of v_n, since by definition $(K+2)\cdot d \geqslant n_K$.

Consider the memory state assumed when starting from $(v_{n_{K+1}},m_0)$, Adam skips until he reaches v_n; denote it m_0 as well, by assumption σ ensures $B(N)$ Until Reach from (v_n,m_0). Furthermore, without loss of generality we can assume that no play from (v_n,m_0) consistent with σ comes back to v_n, following the same proof as for $\mathcal{G}_1$. This implies that no plays from (v_n,m_0) consistent with σ reach $(v_{n'},m)$ for some $n'\in\{n-n_K,\dots,n\}$, as from there Adam would lead to $u_{K+1,n-n_K}$ where σ would reset, leading to the left of v_n, from where Adam would lead to v_n by skipping an appropriate number of times.

The strategy σ induces a strategy σ' using one less memory state, which ensures $B(N)$ Until Reach$'$ from v_n, where $F' = \{v_{n'} \mid n' < n-n_K\}$. The strategy σ' can be seen as a strategy in $\mathcal{G}_{K,N}$ from v_{n_K}, so the induction hypothesis concludes. ∎

5.3 Corollary

Theorem 19 (Non-uniform boundedness games are not finite-memory determined)

There exists a B-Büchi game and an initial vertex such that:

- Eve has a strategy ensuring $B \cap$ Büchi,
- no finite-memory strategy ensures $B \cap$ Büchi.

Proof. We construct a B-Büchi game, relying on the arenas $\mathcal{G}_{K,N}$ from Theorem 18. In this game:

- first, Adam chooses a value K, that will be fixed for the whole play,
- then the game proceeds by rounds (possibly infinitely many). In a round, Adam chooses a value N and the game continues in $\mathcal{G}_{K,N}$; upon reaching F the round ends and the counter is reset.

The condition is $B \cap$ Büchi; the Büchi condition is equivalent to playing infinitely many rounds.

We first show that Eve has a winning strategy. Thanks to Theorem 18 she has a strategy ensuring $B(K(K+3))$ Until Reach in $\mathcal{G}_{K,N}$; note that the strategy is different for different values of N, but the bound is independent of N. Playing this strategy ensures $B \cap$ Büchi.

We now show that there is no finite-memory winning strategy. Assume towards contradiction that there is such a winning strategy, using mem memory states. Consider the strategy of Adam which consists in first choosing $K = \text{mem}$, and then an unbounded sequence for N. Thanks to Theorem 18, the strategy in $\mathcal{G}_{K,N}$ does not ensure $B(N)$ Until Reach, so either F is never reached or the counter value exceeds N. It follows that either the Büchi condition is not satisfied, or the counter values are unbounded, leading to a contradiction. ∎

6 The LoCo Conjecture for Thin Trees

In this section, we prove that the LoCo conjecture holds for the special case of thin tree arenas. This is a joint work with Florian Horn, Denis Kuperberg and Michał Skrzypczak [FHKS15].

Theorem 20 (The LoCo conjecture for thin tree arenas)

There exist two functions $\mathrm{mem} : \mathbb{N}^2 \to \mathbb{N}$ and $\alpha : \mathbb{N}^4 \to \mathbb{N}$ such that for all boundedness games with k counters and $d+1$ colors over thin tree arenas of width W, for all initial vertices v_0, Eve has a strategy to ensure $B(k^k \cdot \mathrm{val}(v_0)^{2k} \cdot \alpha(d, k, W+2, \mathrm{val}(v_0))) \cap \mathrm{Parity}$, with $W \cdot 3^k \cdot 2k! \cdot \mathrm{mem}(d, k)$ memory states.

The functions α and mem are defined inductively. Since the proof will work by induction on the number of colors, removing the least important color, we define four functions:

- α_0 and mem_0 when the least important color is 0, so the set of colors is $\{0, \dots, d\}$,
- α_1 and mem_1 when the least important color is 1, so the set of colors is $\{1, \dots, d\}$:

$$\alpha_0(d, k, W, N) = \alpha_1(d-1, k+1, 3W, W \cdot (N+1)^k),$$

$$\alpha_1(d, k, W, N) = \begin{cases} k^k \cdot N^{2k} & \text{if } d = 2, \\ \alpha_0(d, k, 2W, N) & \text{otherwise,} \end{cases}$$

$$\mathrm{mem}_0(d, k) = 2 \cdot \mathrm{mem}_1(d-1, k+1),$$

$$\mathrm{mem}_1(d, k) = \begin{cases} 2 \cdot k! & \text{if } d = 1, \\ 2 \cdot \mathrm{mem}_0(d, k) & \text{otherwise.} \end{cases}$$

Define $\alpha = \alpha_0$ and $\mathrm{mem} = \mathrm{mem}_0$.

As an intermediate result, in Subsection 6.1 we will prove that the conjecture holds for the special case of word arenas.

6.1 Existence of a trade-off for word arenas

This subsection is devoted to proving the following result:

Proposition 3 (The LoCo conjecture over word arenas)

There exists two functions $\mathrm{mem} : \mathbb{N}^2 \to \mathbb{N}$ and $\alpha : \mathbb{N}^4 \to \mathbb{N}$ such that for all B-parity games over word arenas of width W with initial vertex v_0, Eve has a strategy to ensure $B(\alpha(d, W, k, \mathrm{val}(v_0))) \cap \mathrm{Parity}$, with $\mathrm{mem}(d, k)$ memory states.

We prove Proposition 3 by induction on the colors in the parity condition. Consider a B-parity game $\mathcal{G}$. We examine two cases, depending whether the least important color (*i.e* the smallest) that appears is odd or even:

- if the set of colors is $\{1, \dots, d\}$, then we construct a B-parity game $\mathcal{G}'$ using $\{2, \dots, d\}$ as colors,
- if the set of colors is $\{0, \dots, d\}$, then we construct a B-parity game $\mathcal{G}'$ using $\{1, \dots, d\}$ as colors.

In both cases, we obtain from the induction hypothesis a winning strategy using few memory states in $\mathcal{G}'$, which we use to construct a winning strategy using few memory states in $\mathcal{G}$. The base case is given by Büchi conditions, and follows from Theorem 16.

Let k be the number of counters, W the width of the word arena $\mathcal{G}$, and v_0 the initial vertex. The ranking function is denoted $r : V \to \mathbb{N}$, and the two colouring functions $\mathbf{c} : E \to \{\varepsilon, i, r\}^k$ and $\Omega : V \to \{0, \dots, d\}$ (or $\{1, \dots, d\}$). Denote $N = \text{val}(v_0)$.

6.1.1 Removing the least important color: the odd case

The first case we consider is when the least important color is 1. The proof that follows is an extension of the slicing technique developed by Vanden Boom [Van11] and illustrated in Subsection 4.5. Note that if 1 is the only color in the arena, then Eve cannot win and the result is true; we now assume that the color 2 also appears in the arena.

Let σ be a strategy ensuring $B(N) \cap \text{Parity}$ from v_0. Without loss of generality we restrict ourselves to vertices reachable with σ from v_0.

Consider a vertex v and T_v the tree of plays consistent with σ. The strategy σ ensures the parity condition, so in particular every branch in T_v contains a vertex of color greater than 1. We prune the tree T_v by cutting paths when they first meet a vertex of color greater than 1. Since the arena has finite out-degree, so is T_v and by König's Lemma the tree obtained is finite. Thus, to every vertex v, we can associate a rank r_v such that the strategy σ ensures that all paths from v contain a vertex of color greater than 1 before reaching the rank r_v.

For $k < k'$, define $\mathcal{G}_{[k,k']}$ the arena induced by $\mathcal{G}$ restricted to vertices of rank in $[k, k']$. Since $\mathcal{G}$ has finite out-degree, there are finitely many vertices of rank in $[k, k']$, hence $\mathcal{G}_{[k,k']}$ is finite.

We define inductively a sequence of ranks $(k_i)_{i \in \mathbb{N}}$, such that for all $i \in \mathbb{N}$, the strategy σ ensures that from all vertices of rank k_i, a vertex of color greater than 1 is seen before reaching the rank k_{i+1}. We first set $k_0 = r_{v_0}$. Assume k_i has been defined, we define k_{i+1} as $\max_{v \in V} \{r_v \mid r(v) = k_i\}$.

The arenas $\mathcal{G}_{[k_i,k_{i+1}]}$ are called slices.

Now we equip $\mathcal{G}$ with a memory structure $\mathcal{M}$ of size 3 which keeps track whether or not a vertex of color greater than 1 has been reached in the current slice. Formally, $\mathcal{M}$ has three memory states:

- the memory state $\forall_1$ means that all vertices visited in the current slice have color 1,
- the memory state $\exists_{>1}$ means that some vertex visited in the current slice has color greater than 1,
- the memory state $\bot$ means that a whole slice has been crossed visiting only vertices of color 1.

The initial state is $\forall_1$ if $\Omega(v_0) = 1$, and $\exists_{>1}$ otherwise. The update function is defined as follows:

$$\mu(m,(v,v')) = \begin{cases} \exists_{>1} & \text{if } \Omega(v') > 1, \\ \exists_{>1} & \text{if } m = \exists_{>1} \text{ and } r(v') \notin \{k_i \mid i \in \mathbb{N}\}, \\ \forall_1 & \text{if } \Omega(v') = 1, m = \forall_1 \text{ and } r(v') \notin \{k_i \mid i \in \mathbb{N}\}, \\ \forall_1 & \text{if } \Omega(v') = 1 \text{ and } r(v') \in \{k_i \mid i \in \mathbb{N}\}, \\ \bot & \text{otherwise.} \end{cases}$$

We construct the expanded arena $\mathcal{G} \times \mathcal{M}$ equipped with the colouring function given by the three colouring functions $\mathbf{c}' : E \times M \to \{\varepsilon, i, r\}^k$, $\Omega' : V \times M \to \{2, \ldots, d\}$ and $c_S : V \times M \to \{0, 1\}$ defined by:

$$\mathbf{c}'(e, \_) = \mathbf{c}(e).$$

$$\Omega'(v, \_) = \begin{cases} \Omega(v) & \text{if } \Omega(v) \neq 1, \\ 2 & \text{otherwise.} \end{cases}$$

$$c_S(\_, m) = \begin{cases} 1 & \text{if } m = \bot, \\ 0 & \text{otherwise.} \end{cases}$$

We equip $\mathcal{G} \times \mathcal{M}$ with the condition $B \cap \text{Parity} \cap \text{Safe}$, where B uses the colouring function $\mathbf{c}$, Parity uses the colouring function Ω', and Safe uses the colouring function c_S.

The correctness of the construction is stated in the following lemma:

Lemma 13 (Correctness of the construction)

1. There exists a strategy σ' in $\mathcal{G} \times \mathcal{M}$ ensuring $B(N) \cap \text{Parity} \cap \text{Safe}$.

2. Assume there exists a strategy σ' in $\mathcal{G} \times \mathcal{M}$ ensuring $B(N') \cap \text{Parity} \cap \text{Safe}$ using K memory states, then there exists a strategy σ in $\mathcal{G}$ ensuring $B(N') \cap \text{Parity}$ using $2K$ memory states.

Proof.

1. The strategy σ' that simulates σ, ignoring the memory structure $\mathcal{M}$, ensures $B(N) \cap \text{Parity} \cap \text{Safe}$ by construction.

2. Let σ' be a strategy in $\mathcal{G} \times \mathcal{M}$ ensuring $B(N') \cap \text{Parity} \cap \text{Safe}$ using $\mathcal{M}'$ as memory structure. This induces σ using $\mathcal{M} \times \mathcal{M}'$ as memory structure. Since plays of σ and of σ' are in one-to-one correspondence, σ ensures $B(N') \cap \text{Parity}$. Further, the safety condition satisfied by σ' ensures that infinitely often a vertex of color greater than 1 is seen (specifically, in each slices), so σ satisfies Parity.

   Note that since σ is winning, it does not make use of the memory state $\bot$, so it uses only $2K$ memory states.

∎

We conclude thanks to the induction hypothesis and Lemma 3 (to handle the additional safety condition).

6.1.2 Removing the least important color: the even case

The second case we consider is when the least important color is 0.

We explain the intuition for the case of CoBüchi conditions, *i.e* if there are only colors 0 and 1. Recall that $F = \{v \mid \Omega(v) = 1\}$. Define $S_0 = A_0 = \varnothing$, and for $i \geqslant 1$:

$$\begin{cases} S_{i+1} = \{v \in V \mid \text{there exists } \sigma \text{ that ensures not to visit } F \text{ before } A_i \text{ from } v, \text{ if at all}\} \\ A_{i+1} = \{v \in V \mid \text{there exists } \sigma \text{ that ensures to reach } S_{i+1} \text{ from } v\}. \end{cases}$$

We have $\bigcup_{i\in\mathbb{N}} S_i = \mathrm{Win}_E(\text{CoBüchi})$. A winning strategy based on these sets has two aims: in S_i it avoids F ("Safe" mode) and in A_i it attracts to the next S_i ("Attractor" mode). The key property is that since the arena is a word arena of width W where Eve can bound the counters by N, she only needs to alternate between modes a number of times bounded by a function of N and W. In other words, the sequence $(Y_i)_{i\in\mathbb{N}}$ stabilises after a number of steps bounded by a function of N and W. (A remote variant of this bounded-alternation fact can be found in [KV01].) Hence the CoBüchi condition can be checked using a new counter and a Büchi condition, as follows.

There are two modes: "Safe" and "Attractor". The Büchi condition ensures that the "Safe" mode is visited infinitely often. In the "Safe" mode, only vertices of colors 0 are accepted; visiting a vertex of color 1 leads to the "Attractor" mode and increments the new counter. At any time, she can reset the mode to "Safe". The counter is never reset, so to ensure that it is bounded, Eve must change modes finitely often. Furthermore, the Büchi condition ensures that the final mode is "Safe", implying that the CoBüchi condition is satisfied.

For the more general case of parity conditions, the same idea is used, but as soon as a vertex of color greater than 1 is visited, then the counter is reset.

Define $\mathcal{G}'$:

$$V' = \begin{cases} V'_E = V_E \times \{A, S\} \ \cup \ \overline{V} \\ V'_A = V_A \times \{A, S\}. \end{cases}$$

After each edge followed, Eve is left the choice to set the mode to S. The set of choice vertices is denoted $\overline{V}$. We define E' together with $\mathbf{c}' : E \to \{\varepsilon, i, r\}^{k+1}$:

$$E' = \begin{cases} (v, A) \xrightarrow{\mathbf{c}(v,v'),\varepsilon} \overline{v'} & \text{if } (v, v') \in E, \\ (v, S) \xrightarrow{\mathbf{c}(v,v'),\varepsilon} (v', S) & \text{if } (v, v') \in E \text{ and } \Omega(v') = 0, \\ (v, S) \xrightarrow{\mathbf{c}(v,v'),i} (v', A) & \text{if } (v, v') \in E \text{ and } \Omega(v') = 1, \\ (v, S) \xrightarrow{\mathbf{c}(v,v'),r} (v', S) & \text{if } (v, v') \in E \text{ and } \Omega(v') > 1, \\ \overline{v} \xrightarrow{\varepsilon} (v, A) \text{ and } \overline{v} \xrightarrow{\varepsilon} (v, S) \end{cases}$$

Equip the arena $\mathcal{G}'$ with the colouring function Ω' defined by

$$\Omega'(v, m) = \begin{cases} 1 & \text{if } m = A, \\ 2 & \text{if } \Omega(v) = 0 \text{ and } m = S, \\ \Omega(v) & \text{otherwise.} \end{cases}$$

Before stating and proving the equivalence between $\mathcal{G}$ and $\mathcal{G}'$, we formalise the property mentioned above, that in word arenas Eve does not need to alternate an unbounded number of times between the modes "Safe" and "Attractor".

Lemma 14 (Bounded alternation in word arenas)

Let $\mathcal{G}$ be a word arena of width W, and a subset F of vertices such that every path in $\mathcal{G}$ contains finitely many vertices in F. Define the following sequence of subsets of vertices: $X_0 = \emptyset$, and for $i \geqslant 0$:

$$\begin{cases} X_{2i+1} = \left\{ v \;\middle|\; \begin{array}{c} \text{all paths from } v \text{ contain no vertices in } F \\ \text{before the first vertex in } X_{2i}, \text{ if any} \end{array} \right\}, \\ \\ X_{2i+2} = \left\{ v \;\middle|\; \text{all paths from } v \text{ are finite or lead to } X_{2i+1} \right\}. \end{cases}$$

We have $X_0 \subseteq X_1 \subseteq X_2 \subseteq \cdots$, and X_{2W} covers the whole arena.

Proof. We first argue that the following property, denoted (†), holds: "for all $i \geqslant 0$, if X_{2i} does not cover the whole arena, then $X_{2i+1} \backslash X_{2i-1}$ contains an infinite path". (For technical convenience $X_{-1} = \emptyset$.)

Let $v \notin X_{2i}$. We consider $\mathcal{G}_v$ where v is the initial vertex, and prune it by removing the vertices from X_{2i-1}, as well as vertices which do not have an infinite path after removing X_{2i-1}; denote by $\mathcal{G}'_v$ the arena we get. Note that for any $u \notin X_{2i}$, the vertex u belongs to $\mathcal{G}'_v$, so $\mathcal{G}'_v$ contains an infinite path. We claim that there exists a vertex v' in $\mathcal{G}'_v$ such that all paths from v' contain no vertices F. Indeed, assume towards contradiction that from every node in $\mathcal{G}'_v$, there exists a path to a vertex in F. Then there exists a path that visits infinitely many vertices in F, contradicting the assumption on $\mathcal{G}$. Any infinite path from v' is included into $X_{2i+1} \backslash X_{2i-1}$, hence the latter contains an infinite path.

We conclude using (†): assume towards contradiction that X_{2W} does not cover the whole arena. Then $\mathcal{G}$ contains $W + 1$ pairwise disjoint paths, contradicting that it has width W. ∎

The correctness of the construction is stated in the following lemma:

Lemma 15 (Correctness of the construction)

1. There exists a strategy σ' in $\mathcal{G}'$ that ensures $B(W \cdot (N+1)^k) \cap \text{Parity}$.
2. Assume there exists a strategy σ' in $\mathcal{G}'$ ensuring $B(N') \cap \text{Parity}$ using K memory states, then there exists a strategy σ in $\mathcal{G}$ ensuring $B(N') \cap \text{Parity}$ using $2K$ memory states.

Proof.

1. Thanks to Lemma 11, there exists a strategy σ in $\mathcal{G}$ ensuring $B(N) \cap \text{Parity}$ using a memory structure $\mathcal{M}$ of size $(N+1)^k$. We construct a strategy σ' in $\mathcal{G}'$ simulating σ. We now explain when does σ' chooses to set the "Safe" mode.

   We consider the arena $\mathcal{G} \times \mathcal{M}$, it is a word arena of width $W \cdot (N+1)^k$, and restrict it to the moves prescribed by σ, obtaining the word arena $\mathcal{G}_\sigma$ of width $W \cdot (N+1)^k$. Without loss of generality we restrict $\mathcal{G}_\sigma$ to vertices reachable with σ from the initial vertex $(v_0, 0)$. Consider a vertex v of color 0 or 1, and $\mathcal{G}^v_\sigma$ the word arena obtained by considering v as initial vertex and pruned by cutting paths when they first meet a vertex of color greater than 1. Since the strategy σ ensures that the parity condition is satisfied, every infinite path in $\mathcal{G}_\sigma^v$ contains finitely many vertices of color 1. Relying on Lemma 14 for the word arena G_σ^v and F the set of vertices of color 1, we associate to each vertex v' in $\mathcal{G}_\sigma^v$ a rank with respect to v, which is a number between 1 and $2W \cdot (N+1)^k$, the minimal i such that $v' \in X_i(\mathcal{G}_\sigma^v)$.

Now consider a play consistent with σ, and a suffix of this play starting in a vertex v of color 0 or 1. By definition, from this position on, the rank (with respect to v) is non-increasing until a vertex of color greater than 1 is visited, if any. Furthermore, if the rank is even then no vertices of color 1 are visited, and the rank does not remain forever odd.

The strategy σ' in $\mathcal{G}'$ simulates σ, and at any point of a play remembers the first vertex v that has not been followed by a vertex of color greater than 1. As observed above, the rank with respect to v is non-increasing; the strategy σ' switches to the "Safe" mode when the rank goes from even to odd. By definition, the new counter is incremented only when the rank goes from odd to even, which happens at most $W \cdot (N+1)^k$ times, and it is reset when a vertex of color greater than 1 is visited, so σ' ensure that it remains bounded by $W \cdot (N+1)^k$.

Also, since σ ensures to bound the counter values by N, then so does σ'. For the parity condition, there are two cases. Consider a play consistent with σ'. Either from some point onwards the only colors seen are 0 and 1 (with respect to Ω), then the new counter is not reset after this point, but it is incremented only when the rank decreases from odd to even, which corresponds to switches of mode from "Safe" to "Attractor". Since this counter is bounded, the mode stabilises, which by definition of the ranks imply that the stabilised rank is odd, so the mode is "Safe", and from there on only vertices of color 0 (with respect to Ω) are visited, hence Parity is satisfied. Or infinitely many vertices of color greater than 1 are seen (with respect to Ω), but since they coincide for Ω and Ω', the condition Parity is satisfied.

It follows that σ' ensures $B(W \cdot (N+1)^k) \cap \text{Parity}$.

2. Let σ' be a strategy in $\mathcal{G}'$ ensuring $B(N') \cap \text{Parity}$ using $\mathcal{M}'$ as memory structure of size K. We construct σ that simulates σ'; to this end, we need a memory structure which simulates both $\mathcal{M}'$ and the mode alternation, of size $2K$. By definition, plays of σ and plays of σ' are in one-to-one correspondence, so σ ensures $B(N')$. For the parity condition, there are two cases. Consider a play consistent with σ'. Either from some point onwards the only colors seen are 0 and 1 (with respect to Ω), then the new counter is not reset after this point, but it is incremented each time the mode switches from "Safe" to "Attractor"; since this counter is bounded, the mode stabilises, and since the play in $\mathcal{G}'$ satisfies Parity, the stabilised mode is "Safe", implying that from there on only vertices of color 0 (with respect to Ω) are visited, hence satisfy Parity. Or infinitely many vertices of color greater than 1 are seen (with respect to Ω), but since they coincide for Ω and Ω', the condition Parity is satisfied.

∎

The induction hypothesis concludes.

6.2 Extending to thin tree arenas

In this subsection, we extend the results for word arenas to thin tree arenas, proving Theorem 20.

Consider a B-parity game $\mathcal{G}$. Define $N = \text{val}(v_0)$. Let $R : V \to \{0,1\}^*$ witnessing that $\mathcal{G}$ is a thin tree arena. Let σ be a strategy ensuring $B(N) \cap \text{Parity}$ from v_0. We rely on the decomposition of the thin tree $R(V)$ to locally replace σ by strategies using small memory given by Proposition 3.

It follows from Fact 15 that along a play, the rank is non-increasing and decreases only finitely many times. Since the parity condition is prefix-independent, if for each rank Eve plays a strategy ensuring the parity condition, then the resulting strategy ensures the parity condition; however, a closer attention to the counters is required.

We summarize counter actions as follows: let $w \in (\{\varepsilon, i, r\}^k)^*$, its summary $\text{sum}(w) \in \{\varepsilon, i, r\}^k$ is, for each counter, r if the counter is reset in w, i if the counter is incremented by not reset in w, and ε otherwise.

Fact 2 (Summary)

Consider $w = w_1 w_2 \cdots w_n w_\infty$, where $w_1, \ldots, w_n \in (\{\varepsilon, i, r\}^k)^*$ and $w_\infty \in (\{\varepsilon, i, r\}^k)^\omega$. Denote $u = \text{sum}(w_1)\text{sum}(w_2)\cdots\text{sum}(w_n)\text{sum}(w_\infty)$, then:

1. $\text{val}(u) \leqslant \text{val}(w)$,
2. if for all $i \in \{1, \ldots, n, \infty\}$ we have $\text{val}(w_i) \leqslant N'$ and $\text{val}(u) \leqslant N$, then $\text{val}(w) \leqslant N \cdot N'$.

We define a B game $\mathcal{G}'$, where the plays that remain in vertices of the same rank are summarized in one step. It has k counters. Let $\text{rank}(V)$ denote the set of ranks (subset of the countable ordinals), and $\mathcal{S}$ the set of all strategies in $\mathcal{G}$ ensuring $B(N) \cap \text{Parity}$. We define the set of vertices of $\mathcal{G}'$:

$$V' = \begin{cases} V'_E = \text{rank}(V) \times \{1, \ldots, W\} \times \{\varepsilon, i, r\}^k \\ V'_A = \text{rank}(V) \times \{1, \ldots, W\} \times \mathcal{S} \end{cases}$$

We explain how a couple $(\nu, \ell) \in \text{rank}(V) \times \{1, \ldots, W\}$ uniquely determines a vertex in $\mathcal{G}$. First, the rank ν corresponds in R either to a node or to an infinite branch, in the second case we consider the first node in this branch. Second, the component ℓ identifies a vertex in this node.

We say that (ν, ℓ', a) is an outcome of (μ, ℓ, σ) if there exists a play from the vertex corresponding to (μ, ℓ) consistent with σ ending in the vertex corresponding to (ν, ℓ') whose summarized counter actions are a. We define the set of edges of $\mathcal{G}'$, together with the counter actions $\mathbf{c}' : E' \to \{\varepsilon, i, r\}^k$:

$$E' = \begin{cases} \{((\nu, \ell, a) \xrightarrow{a} (\nu, \ell, \sigma)) \mid \ell, \nu, \sigma\} \\ \{((\mu, \ell, \sigma) \xrightarrow{\varepsilon} (\nu, \ell', a)) \mid \text{ if } (\nu, \ell', a) \text{ is an outcome of } (\mu, \ell, \sigma)\}. \end{cases}$$

Lemma 16 (Correctness of the construction)

1. There exists a strategy σ' in $\mathcal{G}'$ ensuring $B(N)$.
2. Assume there exists a strategy σ' in $\mathcal{G}'$ ensuring $B(N')$ using K memory states, then there exists a strategy σ in $\mathcal{G}$ ensuring $B(N' \cdot \alpha(d, W+2, k, N)) \cap \text{Parity}$ using $W \cdot 3^k \cdot K \cdot \text{mem}(d,k)$ memory states.

Proof.

1. We argue that the strategy σ induces a strategy σ' in $\mathcal{G}'$ ensuring $B(N)$. A play consistent with σ' is of the form $u = \text{sum}(w_1)\text{sum}(w_2)\cdots\text{sum}(w_n)\text{sum}(w_\infty)$, where $w = w_1 w_2 \cdots w_n w_\infty$ is a play consistent with σ, following the notations of Fact 2. This fact, item 1., implies that $\text{val}(u) \leqslant \text{val}(w)$, so $\text{val}(u) \leqslant N$. Hence the strategy σ' ensures $B(N)$.

2. Assume there exists a strategy σ' in $\mathcal{G}'$ ensuring $B(N')$ and using a memory structure $\mathcal{M}$ of size K. We construct a strategy in $\mathcal{G}$ ensuring $B(N' \cdot \alpha(d, W+2, k, N)) \cap \text{Parity}$ using $W \cdot 3^k \cdot K \cdot \text{mem}(d,k)$ memory states. The memory structure is the product of four memory structures:

   - a memory structure that keeps track of the $\ell \in \{1, \ldots, W\}$ used when entering the current rank, of size W,
   - a memory structure that keeps track of the summary since entering the current rank, of size 3^k,
   - the memory structure $\mathcal{M}$, of size K,
   - a memory structure of size $\text{mem}(d,k)$, which is used to simulate the strategies obtained from Proposition 3.

   Consider $\nu \in \text{rank}(V)$ that corresponds to an infinite branch of $R(V)$. For every $\ell \in \{1, \ldots, W\}$ and $m \in M$, the strategy σ' picks a strategy $\sigma'(\nu, \ell, m)$ to play in this infinite branch, ensuring $B(N) \cap \text{Parity}$. When playing this strategy, two scenarios are possible: either the play stays forever in the infinite branch, or an outcome is selected and the game continues from there.

   We define an arena $\mathcal{G}_{\nu,\ell}$ obtained from $\mathcal{G}$ as follows: the set of vertices is the vertices belonging to the infinite branch corresponding to ν, and the initial vertex is the one corresponding to (ν, ℓ). We add two vertices, corresponding to the good and the bad outcomes; an outcome is good if it is consistent with $\sigma'(\nu, \ell, m)$, and bad otherwise. It is a word arena of width $W+2$ with $d+1$ colors and k counters. The colouring function is given by three colouring functions: $\mathbf{c}'$ and Ω' induced by $\mathbf{c}$ and Ω, and c_S which gives color 0 to all vertices, except the bad outcome coloured 1.

   We equip $\mathcal{G}_{\nu,\ell}$ with the condition $B(N) \cap \text{Parity} \cap \text{Safe}$, where Safe uses the colouring function c_S. The strategy $\sigma'(\nu, \ell, m)$ induces a winning strategy in this game; thanks to Proposition 3 (combined with Lemma 3 to handle the safety condition), there exists a strategy $\sigma(\nu, \ell, m)$ ensuring $B(\alpha(d, W+2, k, N)) \cap \text{Parity} \cap \text{Safe}$ using $\text{mem}(d,k)$ memory states.

The strategy σ simulates the strategies $\sigma(\nu, \ell, m)$ in the corresponding parts of the game. Observe that this requires to keep track of both the value ℓ and the summary of the current rank, which is done by the memory structure.

We argue that σ ensures $B(N' \cdot \alpha(d, W, k, N)) \cap \text{Parity}$. A play consistent with this strategy is of the form $w = w_1 w_2 \cdots w_n w_\infty$, where for all $i \in \{1, \ldots, n, \infty\}$, we have $\text{val}(w_i) \leqslant \alpha(d, W + 2, k, N)$. Denote $u = \text{sum}(w_1)\text{sum}(w_2) \cdots \text{sum}(w_n)\text{sum}(w_\infty)$, we have $\text{val}(u) \leqslant N'$, since it corresponds to a play consistent with σ'. It follows from Fact 2, item 2., that $\text{val}(w) \leqslant N' \cdot \alpha(d, W + 2, k, N)$. Hence the strategy σ ensures $B(N' \cdot \alpha(d, W + 2, k, N)) \cap \text{Parity}$ and uses $W \cdot 3^k \cdot K \cdot \text{mem}(d, k)$ memory states.

∎

We conclude using Lemma 16. Thanks to Theorem 12, there exists a strategy σ' in $\mathcal{G}'$ ensuring $B(k^k \cdot N^{2k})$ and using a memory structure $\mathcal{M}$ of size $2k!$. It follows that there exists a strategy σ in $\mathcal{G}$ ensuring $B(k^k \cdot N^{2k} \cdot \alpha(d, W + 2, k, N)) \cap \text{Parity})$ using $W \cdot 3^k \cdot 2k! \cdot \text{mem}(d, k)$ memory states.

7 Conclusions

We recall some of the results of this chapter, presented under three angles, asking further questions:

- *Characterisation of the memory requirements for a given condition.* In Subsection 2.3, we characterised the memory requirement $\text{mem}(W)$ for all topologically closed conditions W. Roughly speaking, these conditions depend on prefixes of the plays, and specify what happens in the finite. A similar result is known for Muller conditions, by Dziembowski, Jurdziński and Walukiewicz [DJW97]. The Muller conditions, at the opposite of topologically closed conditions, are the ω-regular conditions that specify only the infinite behaviours of plays. Hence the natural continuation and joint generalisation of these two results is to characterise $\text{mem}(W)$ for all ω-regular conditions W.

  Kopczyński made a step in this direction, proving that the *chromatic* memory requirement is computable for all ω-regular conditions [Kop06; Kop09]. The chromatic memory requirement of a condition W is a priori different from $\text{mem}(W)$, as it does not consider all strategies, but only chromatic strategies (which depend only the colors). Kopczyński conjectured that they the two quantities coincide for ω-regular conditions.

- *Finite-memory determinacy for non-uniform boundedness games over infinite arenas.* We obtained two results: on the negative side, Section 5 shows that B-Büchi games are not finite-memory determined; on the positive side, Subsection 2.2 states that finitary parity games are bounded-memory determined. This leaves some space in between: for instance, are cost-parity games, as introduced in [FZ12; FZ14], bounded-memory determined? Can we characterise the boundedness conditions that are bounded-memory determined?

- *The LoCo conjecture over tree arenas.* We obtained two results: on the negative side, Section 5 shows that the LoCo conjecture does not hold, even over finite arenas; on the positive side, Section 6 shows that the LoCo conjecture holds for thin tree arenas. The negative result can be interpreted as the need for a better understanding of the structural properties of tree arenas. The positive result is a promising step towards this goal, as it strongly relies on the decomposition of thin trees.

  The LoCo conjecture over tree arenas remains an appealing conjecture to tackle.

2 The Value 1 Problem for Probabilistic Automata

Contents

This second chapter deals with probabilistic automata over finite words, and focuses on a decision problem called the value 1 problem. This problem has been shown undecidable in 2010 by Hugo Gimbert and Youssouf Oualhadj [GO10]; the objective of this chapter is to answer the following question:

"To what extent is the value 1 problem undecidable?"

To answer this question, we will construct an algorithm, called the Markov Monoid algorithm, prove that it is *often* correct, show that it is the *most correct* algorithm from the literature, and finally argue that it is *in some sense* optimal.

The basic definitions are given in Section 1. In particular, we define probabilistic languages and the value of a probabilistic automaton, state some of the most important undecidability results, and give a simple proof that the regularity problem is undecidable: given a probabilistic language, determine whether it is regular, obtained in collaboration with Michał Skrzypczak [FS15].

Section 2 develops a profinite theory for probabilistic automata, called prostochastic theory. This gives a topological account of the value 1 problem, which naturally reformulates as an emptiness problem in this new framework. The material presented in this section is taken from [Fij15].

We introduce the Markov Monoid algorithm in Section 3, which is an algebraic algorithm meant to partially solve the value 1 problem. We use the prostochastic theory developed in the previous section to prove some properties of this algorithm, and in particular obtain a characterisation: the Markov Monoid algorithm determines, given a probabilistic automaton, whether it accepts a polynomial prostochastic word.

Section 4 shows that this algorithm indeed solves the value 1 problem for the subclass of leaktight automata, that we introduce, and further studies this subclass of probabilistic automata. In particular, we compare our subclass to the other subclasses introduced in the literature to partially solve the value 1 problem, and prove that the subclass of leaktight automata strictly contains all the other classes.

The results presented in these two sections originate from [FGO12] (conference version), which is co-authored with Hugo Gimbert and Youssouf Oualhadj, and [FGKO15] (journal version), with the same co-authors plus Edon Kelmendi.

The previous section shows that the Markov Monoid algorithm subsumes all previous algorithms. The natural question is of course: how far, and how long can this be pushed? To answer this question, we consider undecidability results in Section 5. We give two results.

The first one is the undecidability of the two-tier value 1 problem, which is a simpler variant of the value 1 problem. The reduction proving this result is similar to the one proving the undecidability of the value 1 problem by Gimbert and Oualhadj in [GO10], but we give here a finer analysis, which exhibits the role of convergence speeds in the undecidability.

The second one is the undecidability of the numberless value 1 problems. This shows that the undecidability remains even when the numerical values of the transition probabilities are not specified, which was the topic of the paper [FGHO14] co-authored with Hugo Gimbert, Florian Horn and Youssouf Oualhadj.

The undecidability results we obtain for numberless problems are then applied to robustness problems. In a robustness problem, the input is a probabilistic automaton, subject to small perturbations of its probabilistic transitions. We show that unfortunately, the robustness variants of both the value 1 problem and the emptiness problem are undecidable.

Section 6 gathers and relates the results obtained in the previous sections, to argue that the Markov Monoid algorithm is *in some sense* optimal. We then discuss perspectives and extensions.

1 Probabilistic Automata

This section is devoted to basic definitions: probabilistic automata in Subsection 1.1, probabilistic languages in Subsection 1.2, and the value of a probabilistic automaton in Subsection 1.3. We state some of the most important undecidability results in Subsection 1.4.

To make things a bit more concrete and interesting, we construct two examples:

- a universally non-regular probabilistic automaton,
- a probabilistic automaton which has value 1 or not, depending on the numerical values of the probabilistic transitions.

The first example is the key ingredient in the undecidability proof of the regularity problem, that we give in Subsection 1.5. The second example is the key ingredient in the undecidability proof of the (two-tier) value 1 problem, that we give in Subsection 5.1.

1.1 Probabilistic Automata

Let Q be a finite set of states.

A distribution over Q is a function $\delta : Q \to [0,1]$ such that $\sum_{q \in Q} \delta(q) = 1$. We denote $\mathcal{D}(Q)$ the set of distributions over Q, which we often consider as vectors indexed by Q. We denote by $\frac{1}{3} \cdot q + \frac{2}{3} \cdot q'$ the distribution that picks q with probability $\frac{1}{3}$ and q' with probability $\frac{2}{3}$, and by q the trivial distribution picking q with probability 1.

For $E \subseteq \mathbb{R}$, we denote $\mathcal{M}_{Q \times Q}(E)$ the set of (square) matrices indexed by Q over E. A matrix $M \in \mathcal{M}_{Q \times Q}(\mathbb{R})$ is stochastic if each line is a distribution over Q; the restriction to stochastic matrices is denoted $\mathcal{S}_{Q \times Q}(E)$. We denote I the identity matrix.

Definition 18 (Probabilistic automaton)

A *probabilistic automaton* $\mathcal{A}$ is given by a finite set of states Q, a transition function $\phi : A \to \mathcal{S}_{Q \times Q}(\mathbb{Q})$, a distribution of initial states $\delta_I \in \mathcal{D}(Q)$ and a set of final states $F \subseteq Q$.

This simple and natural definition was given by Rabin in [Rab63]. Observe that it generalises the definition for classical deterministic automata, in which transitions functions are $\phi : A \to \mathcal{S}_{Q \times Q}(\{0,1\})$.

Sometimes, mostly for undecidability proofs, we restrict ourselves to probabilistic automata whose probabilistic transitions have values 0, $\frac{1}{2}$ or 1, *i.e.* $\phi : A \to \mathcal{S}_{Q \times Q}(\{0, \frac{1}{2}, 1\})$. Recall that a dyadic number is a number of the form $\frac{a}{2^b}$ for a, b in $\mathbb{N}$. For the same reasons, it is sometimes more convenient to use as initial distribution a unique state, which will then be denoted q_0. We call the probabilistic automata satisfying both assumptions *simple*.

A transition function $\phi : A \to \mathcal{S}_{Q\times Q}(\mathbb{Q})$ naturally induces a morphism $\phi : A^* \to \mathcal{S}_{Q\times Q}(\mathbb{Q})$. We sometimes give ϕ as a function $Q \times A \to \mathcal{D}(Q)$, which is equivalent.

We denote $P_{\mathcal{A}}(s \xrightarrow{w} t)$ the probability to go from state s to state t reading w on the automaton $\mathcal{A}$, *i.e.* $\phi(w)(s,t)$.

The *acceptance probability* of a word $w \in A^*$ by $\mathcal{A}$ is $\delta_I \cdot \phi(w) \cdot F$, which we denote $P_{\mathcal{A}}(w)$. Note that here, we consider δ_I as a row vector and F as a Boolean column vector over Q. It is equal to

$$\sum_{s\in Q}\sum_{t\in F} \delta_I(s) \cdot \phi(w)(s,t).$$

In words, it is the probability that a run starting from the initial distribution δ_I ends in a final state (*i.e.* a state in F).

Throughout this chapter, we will only consider finite words. The situation for infinite words is addressed at the end of this chapter.

1.2 Probabilistic Languages

How does a probabilistic automaton define a language? The following threshold semantics was introduced by Rabin [Rab63]:

Definition 19 (Probabilistic language)

Let $\mathcal{A}$ be a probabilistic automaton and $x \in (0,1)$. This induces the probabilistic language

$$L^{>x}(\mathcal{A}) = \{w \in A^* \mid P_{\mathcal{A}}(w) > x\}.$$

This simple definition initiated the study of the class of probabilistic languages. Since deterministic automata are a special case of probabilistic automata, this class contains the class of regular languages. To prove that this inclusion is strict, it would be enough to construct a probabilistic automaton $\mathcal{C}$ such that $L^{>x}(\mathcal{C})$ is non-regular for some x. We prove a stronger result, constructing a probabilistic automaton such that for every number x in $(0,1)$, the language $L^{>x}(\mathcal{C})$ is non-regular. We will use this in Subsection 1.5.

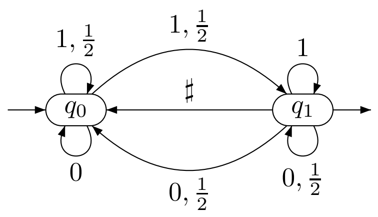


Figure 17: A universally non-regular probabilistic automaton.

In the original paper introducing probabilistic automata, Rabin [Rab63] gave an example of a probabilistic automaton $\mathcal{A}$ such that $L^{>x}(\mathcal{A})$ is non-regular, for all irrational numbers x. The automaton $\mathcal{A}$ computes the binary decomposition function denoted bin, *i.e.* $P_{\mathcal{A}}(u) = \text{bin}(u)$, defined by $\text{bin}(a_1 \cdots a_n) = \frac{a_1}{2^n} + \cdots + \frac{a_n}{2^1}$. We show that adding one letter and one transition to this probabilistic automaton makes it universally non-regular.

Example 2 (Universally non-regular probabilistic automaton)

The automaton $\mathcal{C}$ is represented in Figure 17. The alphabet is $C = \{0, 1, \sharp\}$. The only difference between the automaton $\mathcal{A}$ proposed by Rabin [Rab63] and this one is the letter $\sharp$. As observed by Rabin, a simple induction shows that for u in $\{0,1\}^*$, we have $P_{\mathcal{C}}(u) = \text{bin}(u)$.

We show that for all numbers x in $(0,1)$, the language $L^{>x}(\mathcal{C})$ is non-regular. Let u, v in $\{0,1\}^*$, observe that $P_{\mathcal{C}}(u \cdot \sharp \cdot v) = \text{bin}(u) \cdot \text{bin}(v)$.

Fix x in $(0,1)$. For every u, v in $\{0,1\}^*$ such that $\text{bin}(u) < \text{bin}(v) < x$, there exists w in $\{0,1\}^*$ such that $u \cdot \sharp \cdot w \in L^{>x}(\mathcal{C})$ and $v \cdot \sharp \cdot w \notin L^{>x}(\mathcal{C})$; it suffices to choose w such that $\text{bin}(w)$ is in $\left(\frac{x}{\text{bin}(v)}, \frac{x}{\text{bin}(u)}\right)$. It follows that the left quotients $u^{-1} \cdot L^{>x}(\mathcal{C})$ and $v^{-1} \cdot L^{>x}(\mathcal{C})$ are distinct, so $L^{>x}(\mathcal{C})$ has an infinite number of pairwise distinct left quotients, hence it is not regular.

1.3 The Value of a Probabilistic Automaton

The notion of value for a probabilistic automaton comes from game theory.

Definition 20 (Value of a probabilistic automaton)

The value of a probabilistic automaton $\mathcal{A}$, denoted $\text{val}(\mathcal{A})$, is the supremum over all words of their acceptance probability:

$$\text{val}(\mathcal{A}) = \sup_{w \in A^*} P_{\mathcal{A}}(w).$$

In this chapter, we will be interested in the following decision problem:

Problem 5 (Value 1 problem)

Given a probabilistic automaton $\mathcal{A}$, determine whether $\text{val}(\mathcal{A}) = 1$.

An equivalent formulation of the value 1 problem is as follows: given a probabilistic automaton $\mathcal{A}$, is it true that for all $\varepsilon > 0$, there exists a word w such that $P_{\mathcal{A}}(w) \geqslant 1 - \varepsilon$? In other words, is there a sequence of words $(u_n)_{n \in \mathbb{N}}$ such that $\lim_n P_{\mathcal{A}}(u_n) = 1$?

The value 1 problem can also be reformulated using the notion of *isolated cut-point* introduced by Michael O. Rabin in his seminal paper [Rab63]: an automaton has value 1 if, and only if, the cut-point 1 is *not* isolated.

Yet another formulation of the value 1 problem is by considering probabilistic automata over infinite words, as studied in [BG05; BBG12; BBG08; BBG09]: the value 1 problem is Turing-equivalent to the emptiness problem for probabilistic Büchi automata with probable semantics.

A first (very simple) reduction has been given in [BBG12]: from a probabilistic automaton $\mathcal{A}$ over finite words, one can construct a probabilistic Büchi automaton $\mathcal{A}'$ of linear size, such that $\mathrm{val}(\mathcal{A}) = 1$ if, and only if, $\mathcal{A}'$ is non-empty for the probable semantics. The converse reduction is more involved, and follows from [CSV13], but here the constructed automaton is of exponential size.

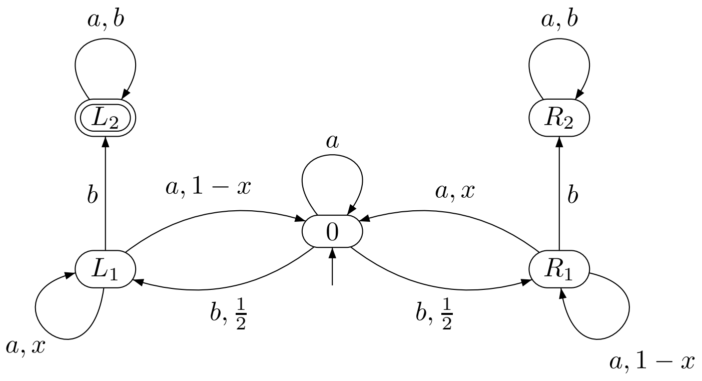


Figure 18: A probabilistic automaton which has value 1 if, and only if, $x > \frac{1}{2}$.

Example 3 (Convergence speeds and the value 1 problem)

We construct a probabilistic automaton, depicted on Figure 18, such that the answer to the value 1 problem depends *quantitatively* on the transition probabilities. This example was introduced in [GO10]. It has value 1 if, and only if, $x > \frac{1}{2}$. The input alphabet is $A = \{a, b\}$, the initial state is the central state 0 and the unique final state is L_2.

We describe its behaviour. After reading one b, the distribution is uniform over L_1, R_1. To reach L_2, one needs to read a b from the state L_1, but on the right-hand side this leads to the non-accepting absorbing state R_2. In order to maximize the probability to reach L_2, one tries to "tip the scales" to the left. If $x \leqslant \frac{1}{2}$, there is no hope to achieve this: reading a letter a gives more chance to stay in R_1 than in L_1 thus all words are accepted with probability at most $\frac{1}{2}$, and $\mathrm{val}(\mathcal{A}) = \frac{1}{2}$. However, if $x > \frac{1}{2}$ then we show that $\mathcal{A}$ has value 1.

We have:

$$P_{\mathcal{A}}(0 \xrightarrow{ba^n} L_1) = \frac{1}{2} \cdot x^n \qquad \text{and} \qquad P_{\mathcal{A}}(0 \xrightarrow{ba^n} R_1) = \frac{1}{2} \cdot (1-x)^n.$$

We fix an integer N and analyse the action of reading $(ba^n)^N$: there are N "rounds", each of them corresponding to reading ba^n from 0. In a round, there are three outcomes: winning (that is, remaining in L_1) with probability $p_n = \frac{1}{2} \cdot x^n$, losing (that is, remaining in R_2) with probability $q_n = \frac{1}{2} \cdot (1-x)^n$, or going to the next round (that is, reaching 0) with probability $1 - (p_n + q_n)$. If a round is won or lost, then the next b leads to an accepting or rejecting sink; otherwise it goes on to the next round, for N rounds. Hence:

$$\begin{aligned} P_{\mathcal{A}}((ba^n)^N) &= \textstyle\sum_{k=1}^{N-1} (1 - (p_n + q_n))^{k-1} \cdot p_n \\ &= p_n \cdot \tfrac{1-(1-(p_n+q_n))^{N-1}}{1-(1-(p_n+q_n))} \\ &= \tfrac{1}{1+\frac{q_n}{p_n}} \cdot \left(1 - (1 - (p_n + q_n))^{N-1}\right) \end{aligned}$$

We now set $N = 2^n$ and assume $x > \frac{1}{2}$. A simple calculation shows that the sequence $((1 - (p_n + q_n))^{2^n - 1})_{n \in \mathbb{N}}$ converges to 0 as n goes to infinity. Furthermore, $\frac{1-x}{x} < 1$, so $\frac{q_n}{p_n} = \left(\frac{1-x}{x}\right)^n$ converges to 0 as n goes to infinity. It follows that the acceptance probability converges to 1 as n goes to infinity. Consequently:

$$\lim_n P_{\mathcal{A}}((ba^n)^{2^n}) = 1.$$

This example witnesses two interesting phenomena:

- the value is discontinuous with respect to the transition probabilities, as for $x = \frac{1}{2}$ the value is $\frac{1}{2}$, and for $x > \frac{1}{2}$ the value is 1;
- the sequence of words $((ba^n)^{2^n})_{n \in \mathbb{N}}$ witnessing the value 1 involves two convergence "speeds": indeed, the words $a^n b$ are repeated an exponential number of times, namely 2^n. We will call such sequences two-tier sequences in Subsection 5.1. One can show that repeating only n times does not lead to words accepted with arbitrarily high probability.

1.4 Undecidability Results

The first and most natural problem for probabilistic automata is the emptiness problem for probabilistic languages.

Problem 6 (Emptiness problem)

Given a probabilistic automaton $\mathcal{A}$, determine whether $L^{>\frac{1}{2}}(\mathcal{A})$ is empty, *i.e.* whether there exists a word w such that $P_{\mathcal{A}}(w) > \frac{1}{2}$.

Note that it is equivalent to asking whether $\text{val}(\mathcal{A}) > \frac{1}{2}$. Unfortunately:

Theorem 21 (Undecidability of the emptiness problem [Paz71])

The emptiness problem is undecidable for simple probabilistic automata.

A nice and simple undecidability proof was given by Gimbert and Oualhadj in [GO10].

This negative result says that the value cannot be computed precisely. One can still try to approximate it. Unfortunately, a second result shows that this is not possible either:

Theorem 22 (Inapproximability of the value [CL89])

There exists no algorithm with the following behaviour.
Given a probabilistic automaton $\mathcal{A}$:

- if $\text{val}(\mathcal{A}) > \frac{2}{3}$, then the algorithm answers "YES",

- if $\mathrm{val}(\mathcal{A}) < \frac{1}{3}$, then the algorithm answers “NO”,
- otherwise, the algorithm can do anything, including not terminating.

An interpretation of this result is that the value cannot be approximated, even up to a constant. In [GO10], Gimbert and Oualhadj introduced the value 1 problem. It is a priori simpler, as it does not ask to compare the value to a given threshold. Unfortunately:

Theorem 23 (Undecidability of the value 1 problem [GO10])

The value 1 problem is undecidable for simple probabilistic automata.

This result is the starting point of this chapter, which aims at understanding to what extent is the value 1 problem undecidable.

Note that in Subsection 5.1, we revisit the undecidability proof of [GO10], and obtain a stronger undecidability result, for the two-tier value 1 problem.

1.5 Undecidability of the Regularity Problem

Problem 7 (Regularity problem)

Given a probabilistic automaton $\mathcal{A}$, determine whether $L^{>\frac{1}{2}}(\mathcal{A})$ is regular.

We give here a very simple proof, obtained in collaboration with Michał Skrzypczak [FS15]. This result was known [Ber74a].

Theorem 24 (Undecidability of the regularity problem)

The regularity problem is undecidable for simple probabilistic automata.

Roughly speaking, the idea is to use the universally non-regular automaton given in Example 2 to “amplify” an irregular behaviour.

Proof. We construct a reduction from the emptiness problem to the regularity problem. Let $\mathcal{A}$ be a probabilistic automaton over the alphabet A. We construct a probabilistic automaton $\mathcal{B}$ such that:

$$L^{>\frac{1}{2}}(\mathcal{A}) \text{ is empty if, and only if, } L^{>\frac{1}{2}}(\mathcal{B}) \text{ is regular.}$$

The automaton $\mathcal{B}$ is over the alphabet $B = A \uplus C$ where $C = \{0, 1, \sharp\}$, and uses the automaton $\mathcal{C}$ from Section 1.2. It is obtained as the sequential composition of $\mathcal{A}$ and $\mathcal{C}$: it starts in $\mathcal{A}$ and from every final state of $\mathcal{A}$ moves by $\sharp$ to the initial state of $\mathcal{C}$. The initial state of $\mathcal{B}$ is the initial set of $\mathcal{A}$, the only final state of $\mathcal{B}$ is the final state of $\mathcal{C}$.

For $u \in A^*$ and $v \in C^*$, we have $\mathbb{P}_{\mathcal{B}}(u \cdot \sharp \cdot v) = \mathbb{P}_{\mathcal{A}}(u) \cdot \mathbb{P}_{\mathcal{C}}(v)$. A word which is not in $A^* \cdot \sharp \cdot C^*$ has no accepting run, so is accepted with probability 0.

- Assume that $L^{>\frac{1}{2}}(\mathcal{A})$ is empty. Thanks to the above observation we have that $L^{>\frac{1}{2}}(\mathcal{B})$ is empty, so in particular it is regular.
- Conversely, assume that $L^{>\frac{1}{2}}(\mathcal{A})$ is non-empty. Let u be a word such that $P_{\mathcal{A}}(u) > \frac{1}{2}$. Observe that $L^{>\frac{1}{2}}(\mathcal{B}) \cap (u \cdot \sharp \cdot C)^* = u \cdot \sharp \cdot L^{>x}(\mathcal{C})$, where $x = \frac{1}{2 \cdot P_{\mathcal{A}}(u)}$ is in $(0, 1)$. By construction of $\mathcal{C}$, the language $L^{>x}(\mathcal{C})$ is non-regular, hence so is M, implying that $L^{>\frac{1}{2}}(\mathcal{B})$ is also non-regular.

∎

2 The Prostochastic Theory

In this section, we introduce the prostochastic theory, which is a profinite theory for probabilistic automata. We construct the free prostochastic monoid in Subsection 2.1.

The aim of this theory is to give a topological account of the value 1 problem; we will show in Subsection 2.2 that the value 1 problem naturally reformulates as an emptiness problem for prostochastic words.

In Subsection 2.3 we show how to construct non-trivial prostochastic words. We will use this material in Section 3 to prove some properties of the Markov Monoid algorithm. Subsection 2.4 is devoted to proving a technical result about powers of a stochastic matrix.

All the material presented here is taken from [Fij15].

2.1 The Free Prostochastic Monoid

We consider the norm $||\cdot||$ defined on $\mathcal{M}_{Q\times Q}(\mathbb{R})$ by

$$||M|| = \max_{s\in Q} \sum_{t\in Q} |M(s,t)|.$$

The following classical properties will be useful:

Fact 3 (Topology of the stochastic matrices)

- For every matrix $M \in \mathcal{S}_{Q\times Q}(\mathbb{R})$, we have $||M|| = 1$,
- For every matrices $M, M' \in \mathcal{M}_{Q\times Q}(\mathbb{R})$, we have $||M \cdot M'|| \leqslant ||M|| \cdot ||M'||$,
- The monoid $\mathcal{S}_{Q\times Q}(\mathbb{R})$ is compact.

The purpose of the prostochastic theory is to construct a compact monoid $\mathcal{P}A^*$ together with a continuous injective morphism $\iota : A^* \to \mathcal{P}A^*$, called the free prostochastic monoid, satisfying the following universal property:

"Every morphism $\phi : A^* \to \mathcal{S}_{Q\times Q}(\mathbb{R})$ extends uniquely
to a continuous morphism $\widehat{\phi} : \mathcal{P}A^* \to \mathcal{S}_{Q\times Q}(\mathbb{R})$."

Here, by "$\widehat{\phi}$ extends ϕ" we mean $\phi = \widehat{\phi} \circ \iota$.

We give two statements about $\mathcal{P}A^*$, the first will be weaker but enough for our purposes in this paper, and the second more precise, and justifying the name "free prostochastic monoid". The reason for giving two statements is that the first avoids a number of technical points that will not play any further role, so the reader interested in the applications to the Markov Monoid algorithm may skip this second statement.

Theorem 25 (Existence of the free prostochastic monoid – weaker statement)

For every finite alphabet A, there exists a compact monoid $\mathcal{P}A^*$ and a continuous injective morphism $\iota : A^* \to \mathcal{P}A^*$ such that every morphism $\phi : A^* \to \mathcal{S}_{Q\times Q}(\mathbb{R})$ extends uniquely to a continuous morphism $\widehat{\phi} : \mathcal{P}A^* \to \mathcal{S}_{Q\times Q}(\mathbb{R})$.

We construct $\mathcal{P}A^*$ and ι. Consider $X = \prod_{\phi:A^*\to\mathcal{S}_{Q\times Q}(\mathbb{R})} \mathcal{S}_{Q\times Q}(\mathbb{R})$, the product of $\mathcal{S}_{Q\times Q}(\mathbb{R})$, with one copy for each morphism $\phi : A^* \to \mathcal{S}_{Q\times Q}(\mathbb{R})$. An element m of X is denoted $(m(\phi))_{\phi:A^*\to\mathcal{S}_{Q\times Q}(\mathbb{R})}$.

We define the product norm on X by $||m|| = \sup\{||m(\phi)|| \mid \phi : A^* \to \mathcal{S}_{Q\times Q}(\mathbb{R})\}$, inducing the product topology on X, implying thanks to Tychonoff's theorem that the monoid X is compact.

Consider the map $\iota : A \to X$ defined by $\iota(a) = (\phi(a))_{\phi:A\to\mathcal{P}}$, it induces a continuous injective morphism $\iota : A^* \to X$. To simplify notations, we sometimes assume that $A \subseteq X$ and denote a for $\iota(a)$.

Denote $\mathcal{P}A^* = \overline{A^*}$, the closure of $A^* \subseteq X$. Note that it is a compact monoid: the compactness follows from the fact that it is closed in X.

By definition, an element $\overline{u}$ of $\mathcal{P}A^*$, called a *prostochastic word*, is obtained as the limit in $\mathcal{P}A^*$ of a sequence $\mathbf{u}$ of finite words. In this case we write $\lim \mathbf{u} = \overline{u}$ and say that $\mathbf{u}$ induces $\overline{u}$.

Note that by definition of the product topology on X, a sequence of finite words $\mathbf{u}$ converges in X if, and only if, for every morphism $\phi : A^* \to \mathcal{S}_{Q\times Q}(\mathbb{R})$, the sequence of stochastic matrices $\phi(\mathbf{u})$ converges.

We say that two converging sequences of finite words $\mathbf{u}$ and $\mathbf{v}$ are equivalent if they induce the same prostochastic word, *i.e.* if $\lim \mathbf{u} = \lim \mathbf{v}$. Observe that two converging sequences of finite words $\mathbf{u}$ and $\mathbf{v}$ are equivalent if, and only if, for every morphism $\phi : A^* \to \mathcal{S}_{Q\times Q}(\mathbb{R})$, we have $\lim \phi(\mathbf{u}) = \lim \phi(\mathbf{v})$.

Proof. We prove that $\mathcal{P}A^*$ satisfies the universal property. Consider a morphism $\phi : A^* \to \mathcal{S}_{Q\times Q}(\mathbb{R})$, and define $\widehat{\phi} : \mathcal{P}A^* \to \mathcal{S}_{Q\times Q}(\mathbb{R})$ by $\widehat{\phi}(\overline{u}) = \lim \phi(\mathbf{u})$, where $\mathbf{u}$ is *some* sequence of finite words inducing $\overline{u}$. This is well defined and extends ϕ. Indeed, consider two equivalent sequences of finite words $\mathbf{u}$ and $\mathbf{v}$ inducing $\overline{u}$. By definition, for all $\psi : A^* \to \mathcal{S}_{Q\times Q}(\mathbb{R})$, we have $\lim \psi(\mathbf{u}) = \lim \psi(\mathbf{v})$, so in particular for ϕ this implies $\lim \phi(\mathbf{u}) = \lim \phi(\mathbf{v})$, and $\widehat{\phi}$ is well defined. Both continuity and uniqueness are clear.

We prove that $\widehat{\phi}$ is a morphism. Consider

$$D = \{(\overline{u}, \overline{v}) \in \mathcal{P}A^* \times \mathcal{P}A^* \mid \widehat{\phi}(\overline{u} \cdot \overline{v}) = \widehat{\phi}(\overline{u}) \cdot \widehat{\phi}(\overline{v})\}.$$

To prove that $\widehat{\phi}$ is a morphism, we prove that $D = \mathcal{P}A^* \times \mathcal{P}A^*$. First of all, $A^* \times A^* \subseteq D$. Since $A^* \times A^*$ is dense in $\mathcal{P}A^* \times \mathcal{P}A^*$, it suffices to show that D is closed. This follows from the continuity of both product functions in $\mathcal{P}A^*$ and in $\mathcal{S}_{Q\times Q}(\mathbb{R})$ as well as of $\widehat{\phi}$. ∎

We give a second, stronger statement about $\mathcal{P}A^*$, which in particular justifies the name "free prostochastic monoid".

From now on, by "monoid" we mean "compact topological monoids". Note that compact also includes Hausdorff: distinct points have disjoint neighbourhoods. The term topological means that the product function is continuous:

$$\left\{ \begin{array}{ccc} \mathcal{P} \times \mathcal{P} & \to & \mathcal{P} \\ (s,t) & \mapsto & s \cdot t \end{array} \right.$$

A monoid is profinite if any two elements can be distinguished by a morphism into a finite monoid, *i.e.* by a finite automaton. (Formally speaking, this is the definition of residually finite monoids, which coincide with profinite monoids for compact monoids, see [Alm05].)

To define prostochastic monoids, we use a stronger distinguishing feature, namely probabilistic automata. Probabilistic automata correspond to stochastic matrices over the rationals; here we use stochastic matrices over the reals, since $\mathcal{S}_{Q\times Q}(\mathbb{R})$ is compact, while $\mathcal{S}_{Q\times Q}(\mathbb{Q})$ is not.

Definition 21 (Prostochastic monoid)

A monoid $\mathcal{P}$ is prostochastic if for every elements $s \neq t$ in $\mathcal{P}$, there exists a continuous morphism $\psi : \mathcal{P} \to \mathcal{S}_{Q\times Q}(\mathbb{R})$ such that $\psi(s) \neq \psi(t)$.

There are much more prostochastic monoids than profinite monoids. Indeed, $\mathcal{S}_{Q\times Q}(\mathbb{R})$ is prostochastic, but not profinite in general.

The following theorem extends Theorem 25. The statement is the same as in the profinite theory, replacing "profinite monoid" by "prostochastic monoid".

Theorem 26 (Existence of the free prostochastic monoid – stronger statement)

For every finite alphabet A,

1. There exists a prostochastic monoid $\mathcal{P}A^*$ and a continuous injective morphism $\iota : A^* \to \mathcal{P}A^*$ such that every morphism $\phi : A^* \to \mathcal{P}$, where $\mathcal{P}$ is a prostochastic monoid, extends uniquely to a continuous morphism $\widehat{\phi} : \mathcal{P}A^* \to \mathcal{P}$.

2. All prostochastic monoids satisfying this universal property are homeomorphic.

The unique prostochastic monoid satisfying the universal property stated in item 1. is called the free prostochastic monoid, and denoted $\mathcal{P}A^*$.

Proof. We prove that $\mathcal{P}A^*$ satisfies the stronger universal property, along the same lines as for the weaker one. Consider a morphism $\phi : A^* \to \mathcal{P}$, and define $\widehat{\phi} : \mathcal{P}A^* \to \mathcal{P}$ by $\widehat{\phi}(\overline{u}) = \lim \phi(\mathbf{u})$, where $\mathbf{u}$ is *some* sequence of finite words inducing $\overline{u}$.

To see that this is well defined, we use the fact that $\mathcal{P}$ is prostochastic. Consider two equivalent sequences of finite words $\mathbf{u}$ and $\mathbf{v}$ inducing $\overline{u}$. Consider a continuous morphism $\psi : \mathcal{P} \to \mathcal{S}_{Q\times Q}(\mathbb{R})$, the composition $\psi \circ \phi$ is a continuous morphism from A^* to $\mathcal{S}_{Q\times Q}(\mathbb{R})$, so since $\mathbf{u}$ and $\mathbf{v}$ are equivalent it follows that $\lim(\psi\circ\phi)(\mathbf{u}) = \lim(\psi\circ\phi)(\mathbf{v})$, *i.e.* $\lim \psi(\phi(\mathbf{u})) = \lim \psi(\phi(\mathbf{v}))$. Since ψ is continuous, this implies $\psi(\lim \phi(\mathbf{u})) = \psi(\lim \phi(\mathbf{v}))$. We proved that for all continuous morphisms $\psi : \mathcal{P} \to \mathcal{S}_{Q\times Q}(\mathbb{R})$, we have $\psi(\lim \phi(\mathbf{u})) = \psi(\lim \phi(\mathbf{v}))$; since $\mathcal{P}$ is prostochastic, it follows that $\lim \phi(\mathbf{u}) = \lim \phi(\mathbf{v})$, and $\widehat{\phi}$ is well defined.

Clearly $\widehat{\phi}$ extends ϕ. Both continuity and uniqueness are clear. We prove that $\widehat{\phi}$ is a morphism. Consider

$$D = \{(\overline{u}, \overline{v}) \in \mathcal{P}A^* \times \mathcal{P}A^* \mid \widehat{\phi}(\overline{u} \cdot \overline{v}) = \widehat{\phi}(\overline{u}) \cdot \widehat{\phi}(\overline{v})\}.$$

To prove that $\widehat{\phi}$ is a morphism, we prove that $D = \mathcal{P}A^* \times \mathcal{P}A^*$. First of all, $A^* \times A^* \subseteq D$. Since $A^* \times A^*$ is dense in $\mathcal{P}A^* \times \mathcal{P}A^*$, it suffices to show that D is closed. This follows from the continuity of both product functions in $\mathcal{P}A^*$ and in $\mathcal{P}$ as well as of $\widehat{\phi}$.

We prove that $\mathcal{P}A^*$ is prostochastic. Let $\overline{u} \neq \overline{v}$ in $\mathcal{P}A^*$. Consider two sequences of finite words $\mathbf{u}$ and $\mathbf{v}$ inducing respectively $\overline{u}$ and $\overline{v}$, there exists a morphism $\phi : A^* \rightarrow \mathcal{S}_{Q\times Q}(\mathbb{R})$ such that $\lim \phi(\mathbf{u}) \neq \lim \phi(\mathbf{v})$. Thanks to the universal property proved in the first point, this induces a continuous morphism $\widehat{\phi} : \mathcal{P}A^* \rightarrow \mathcal{S}_{Q\times Q}(\mathbb{R})$ such that $\widehat{\phi}(\mathbf{u}) \neq \widehat{\phi}(\mathbf{v})$, finishing the proof.

We now prove that there is a unique prostochastic monoid satisfying the universal property, up to homeomorphism. Let $\mathcal{P}_1$ and $\mathcal{P}_2$ two prostochastic monoids satisfying the universal property, together with two continuous injective morphisms $\iota_1 : A^* \rightarrow \mathcal{P}_1$ and $\iota_2 : A^* \rightarrow \mathcal{P}_2$. Thanks to the universal property, ι_1 and ι_2 are extended to continuous morphisms $\widehat{\iota_1} : \mathcal{P}_2 \rightarrow \mathcal{P}_1$ and $\widehat{\iota_2} : \mathcal{P}_1 \rightarrow \mathcal{P}_2$, and $\widehat{\iota_1} \circ \iota_2 = \iota_1$ and $\widehat{\iota_2} \circ \iota_1 = \iota_2$. This implies that $\widehat{\iota_1} \circ \widehat{\iota_2} \circ \iota_1 = \iota_1$; thanks to the universal property again, there exists a unique continuous morphism θ such that $\theta \circ \iota_1 = \iota_1$, and since both $\widehat{\iota_1} \circ \widehat{\iota_2}$ and the identity morphism on $\mathcal{P}_1$ satisfy this equality, it follows that they are equal. Similarly, $\widehat{\iota_2} \circ \widehat{\iota_1}$ is equal to the identity morphism on $\mathcal{P}_2$. It follows that $\widehat{\iota_1}$ and $\widehat{\iota_2}$ are mutually inverse homeomorphisms between $\mathcal{P}_1$ and $\mathcal{P}_2$. ∎

***Remark* 3.** *The free prostochastic monoid $\mathcal{P}A^*$ contains the free profinite monoid $\widehat{A^*}$. To see this, we start by recalling the definition of $\widehat{A^*}$, which is the set of converging sequences up to equivalence, where:*

- *a sequence of finite words $\mathbf{u}$ is converging if for every deterministic automaton $\mathcal{A}$, the sequence is either ultimately accepted by $\mathcal{A}$ or ultimately rejected by $\mathcal{A}$, i.e. there exists $N \in \mathbb{N}$ such that either for all $n \geqslant N$, the word u_n is accepted by $\mathcal{A}$, or for all $n \geqslant N$, the word u_n is rejected by $\mathcal{A}$,*

- *two sequences of finite words $\mathbf{u}$ and $\mathbf{v}$ are equivalent if for every deterministic automaton $\mathcal{A}$, either both sequences are ultimately accepted by $\mathcal{A}$, or both sequences are ultimately rejected by $\mathcal{A}$.*

Clearly:

- *if a sequence of finite words is converging with respect to $\mathcal{P}A^*$, then it is converging with respect to $\widehat{A^*}$, as deterministic automata form a subclass of probabilistic automata,*

- *if two sequences of finite words are equivalent with respect to $\mathcal{P}A^*$, then they are equivalent with respect to $\widehat{A^*}$.*

Every profinite word induces at least one prostochastic word: by compactness of $\mathcal{P}A^*$, every sequence of finite words $\mathbf{u}$ contains a converging subsequence with respect to $\mathcal{P}A^*$. This defines an injection from $\widehat{A^*}$ into $\mathcal{P}A^*$. In particular, this implies that $\mathcal{P}A^*$ is uncountable.

2.2 Reformulation of the Value 1 Problem

The aim of this subsection is to reformulate the value 1 problem, which talks about sequences of finite words, into an emptiness problem over prostochastic words.

Definition 22 (Prostochastic language of a probabilistic automaton)

Let $\mathcal{A}$ be a probabilistic automaton. The prostochastic language of $\mathcal{A}$ is:

$$L(\mathcal{A}) = \{\overline{u} \mid \widehat{\phi}(\overline{u})(q_0, F) = 1\}.$$

We say that $\mathcal{A}$ accepts a prostochastic word $\overline{u}$ if $\overline{u} \in L(\mathcal{A})$.

Theorem 27 (Reformulation of the value 1 problem)

Let $\mathcal{A}$ be a probabilistic automaton. The following are equivalent:

- $\text{val}(\mathcal{A}) = 1$,
- $L(\mathcal{A})$ is non-empty.

Proof. Assume $\text{val}(\mathcal{A}) = 1$, then there exists a sequence of words $\mathbf{u}$ such that $\lim P_{\mathcal{A}}(\mathbf{u}) = 1$. We see $\mathbf{u}$ as a sequence of prostochastic words. By compactness of $\mathcal{P}A^*$ it contains a converging subsequence. The prostochastic word induced by this subsequence belongs to $L(\mathcal{A})$.

Conversely, let $\overline{u}$ in $L(\mathcal{A})$, *i.e.* such that $\widehat{\phi}(\overline{u})(q_0, F) = 1$. Consider a sequence of finite words $\mathbf{u}$ inducing $\overline{u}$. By definition, we have $\lim \phi(\mathbf{u})(q_0, F) = 1$, *i.e.* $\lim P_{\mathcal{A}}(\mathbf{u}) = 1$, implying that $\text{val}(\mathcal{A}) = 1$. ∎

2.3 The Limit Operator, Fast and Polynomial Prostochastic Words

We show in this subsection how to construct non-trivial prostochastic words. To this end, we need to better understand *convergence speeds phenomena*: different limit behaviours can occur, depending on how fast the underlying Markov chains converge.

We define a limit operator ω. Consider the function $f : \mathbb{N} \to \mathbb{N}$ defined by $f(n) = k!$, where k is maximal such that $k! \leqslant n$. The function f grows linearly: roughly, $f(n) \sim n$. The choice of n is arbitrary; one could replace n by any polynomial, or even by any subexponential function, see Remark 4.

The operator ω takes as input a sequence of finite words, and outputs a sequence of finite words. Formally, let $\mathbf{u}$ be a sequence of finite words, define:

$$\mathbf{u}^\omega = (u_n^{f(n)})_{n \in \mathbb{N}}.$$

It is not true in general that if $\mathbf{u}$ converges, then $\mathbf{u}^\omega$ converges. We will show that a sufficient condition is that $\mathbf{u}$ is fast.

We say that a sequence $(M_n)_{n\in\mathbb{N}}$ converges exponentially fast to M if there exists a constant $C > 1$ such that for all n large enough, we have $||M_n - M|| \leqslant C^{-n}$.

Definition 23 (Fast sequence)

A sequence of finite words $\mathbf{u}$ is fast if it converges (we denote $\overline{u}$ the prostochastic word it induces), and for every morphism $\phi : A^* \to \mathcal{S}_{Q\times Q}(\mathbb{R})$, the sequence $(\phi(u_n))_{n\in\mathbb{N}}$ converges exponentially fast.

A prostochastic word is *fast* if it is induced by *some* fast sequence. We denote by $\mathcal{P}A^*_f$ the set of fast prostochastic words. Note that a priori, not all prostochastic words are induced by some fast sequence.

We first prove that $\mathcal{P}A^*_f$ is a submonoid of $\mathcal{P}A^*$.

Lemma 17 (The concatenation of two fast sequences is fast)

Let $\mathbf{u}, \mathbf{v}$ be two fast sequences.
The sequence $\mathbf{u} \cdot \mathbf{v} = (u_n \cdot v_n)_{n\in\mathbb{N}}$ is fast.

Proof. Consider a morphism $\phi : A^* \to \mathcal{S}_{Q\times Q}(\mathbb{R})$ and $n \in \mathbb{N}$.

$$\begin{array}{cl} \multicolumn{2}{l}{||\phi(u_n \cdot v_n) - \widehat{\phi}(\overline{u} \cdot \overline{v})||} \\ = & ||\phi(u_n) \cdot \phi(v_n) - \widehat{\phi}(\overline{u}) \cdot \widehat{\phi}(\overline{v})|| \\ = & ||\phi(u_n) \cdot (\phi(v_n) - \widehat{\phi}(\overline{v})) - (\widehat{\phi}(\overline{u}) - \phi(u_n)) \cdot \widehat{\phi}(\overline{v})|| \\ \leqslant & ||\phi(u_n)|| \cdot ||\phi(v_n) - \widehat{\phi}(\overline{v})|| + ||\widehat{\phi}(\overline{u}) - \phi(u_n)|| \cdot ||\widehat{\phi}(\overline{v})|| \\ = & ||\phi(v_n) - \widehat{\phi}(\overline{v})|| + ||\widehat{\phi}(\overline{u}) - \phi(u_n)||. \end{array}$$

Since $\mathbf{u}$ and $\mathbf{v}$ are fast, the previous inequality implies that $\mathbf{u} \cdot \mathbf{v}$ is fast. ∎

Let $\overline{u}$ and $\overline{v}$ be two fast prostochastic words, thanks to Lemma 17, the prostochastic word $\overline{u} \cdot \overline{v}$ is fast.

The remainder of this subsection is devoted to proving that ω is an operator $\mathcal{P}A^*_f \to \mathcal{P}A^*_f$.

We start by a few definitions. As a convention, M denotes a matrix in $\mathcal{S}_{Q\times Q}(\mathbb{R})$, and m a Boolean matrix.

Definition 24 (Idempotent Boolean matrix, recurrent and transient state)

Consider a matrix $M \in \mathcal{S}_{Q\times Q}(\mathbb{R})$, its Boolean projection $\pi(M)$ is the Boolean matrix such that $\pi(M)(s,t) = 1$ if $M(s,t) > 0$, and $\pi(M)(s,t) = 0$ otherwise.

Let m be a Boolean matrix. It is idempotent if $m \cdot m = m$.
Assume m is idempotent. We say that:

- the state $s \in Q$ is m-recurrent if for all $t \in Q$, if $m(s,t) = 1$, then $m(t,s) = 1$,
- the m-recurrent states $s, t \in Q$ belong to the same recurrence class if $m(s,t) = 1$,
- the state $s \in Q$ is m-transient if it is not m-recurrent.

The main technical tool is the following theorem, stating the exponentially fast convergence of the powers of a stochastic matrix.

Theorem 28 (Powers of a stochastic matrix)

Let $M \in \mathcal{S}_{Q \times Q}(\mathbb{R})$. Denote $P = M^{|Q|}$. Then the sequence $(P^n)_{n \in \mathbb{N}}$ converges exponentially fast to a matrix M^ω, satisfying:

$$\pi(M^\omega)(s,t) = \begin{cases} 1 & \text{if } \pi(P)(s,t) = 1 \text{ and } t \text{ is } \pi(P)\text{-recurrent}, \\ 0 & \text{otherwise.} \end{cases}$$

The proof of Theorem 28 is given in Subsection 2.4.

Lemma 18 (Limit operator for fast sequences)

Let $\mathbf{u}, \mathbf{v}$ be two equivalent fast sequences, inducing the fast prostochastic word $\overline{u}$. Then the sequences $\mathbf{u}^\omega$ and $\mathbf{v}^\omega$ are fast and equivalent, inducing the fast prostochastic word denoted $\overline{u}^\omega$.

Furthermore, for every $\phi : A \to \mathcal{S}_{Q \times Q}(\mathbb{R})$, we have $\widehat{\phi}(\overline{u}^\omega) = \widehat{\phi}(\overline{u})^\omega$.

Proof. Let $\phi : A \to \mathcal{S}_{Q \times Q}(\mathbb{R})$.

Observe that the sequence $(\widehat{\phi}(\overline{u})^{f(n)})_{n \in \mathbb{N}}$ is a subsequence of $(\widehat{\phi}(\overline{u})^{|Q| \cdot n})_{n \in \mathbb{N}}$, so Theorem 28 implies that it converges exponentially fast to $\widehat{\phi}(\overline{u})^\omega$. It follows that there exists a constant $C_1 > 1$ such that for all n large enough, we have $||\widehat{\phi}(\overline{u})^{f(n)} - \widehat{\phi}(\overline{u})^\omega|| \leqslant C_1^{-f(n)}$.

We proceed in two steps, using the following inequality, which holds for every n:

$$||\phi(u_n^{f(n)}) - \widehat{\phi}(\overline{u})^\omega|| \leqslant ||\phi(u_n)^{f(n)} - \widehat{\phi}(\overline{u})^{f(n)}|| + ||\widehat{\phi}(\overline{u})^{f(n)} - \widehat{\phi}(\overline{u})^\omega||.$$

For the left part, we rely on the following equality, where x and y may not commute:

$$x^N - y^N = \sum_{k=0}^{N-1} x^{N-k-1} \cdot (x - y) \cdot y^k.$$

Let $N = f(n)$, this gives:

$$\begin{aligned}
&||\phi(u_n)^N - \widehat{\phi}(\overline{u})^N|| \\
&= ||\sum_{k=0}^{N-1} \phi(u_n)^{N-k-1} \cdot (\phi(u_n) - \widehat{\phi}(\overline{u})) \cdot \widehat{\phi}(\overline{u})^k|| \\
&\leqslant \sum_{k=0}^{N-1} ||\phi(u_n)^{N-k-1}|| \cdot ||\phi(u_n) - \widehat{\phi}(\overline{u})|| \cdot ||\widehat{\phi}(\overline{u})^k|| \\
&\leqslant \sum_{k=0}^{N-1} \underbrace{||\phi(u_n)||^{N-k-1}}_{=1} \cdot ||\phi(u_n) - \widehat{\phi}(\overline{u})|| \cdot \underbrace{||\widehat{\phi}(\overline{u})||^k}_{=1} \\
&= N \cdot ||\phi(u_n) - \widehat{\phi}(\overline{u})||.
\end{aligned}$$

Since $\mathbf{u}$ is fast, there exists a constant $C_2 > 1$ such that $||\phi(u_n) - \widehat{\phi}(\overline{u})|| \leqslant C_2^{-n}$. Altogether, we have

$$||\phi(u_n^{f(n)}) - \widehat{\phi}(\overline{u})^\omega|| \leqslant f(n) \cdot C_2^{-n} + C_1^{-f(n)}.$$

To conclude, observe that for all n large enough, we have $\frac{n}{\log(n)} \leqslant f(n) \leqslant n$. It follows that the sequence $\mathbf{u}^\omega$ is fast, and that $\phi(\mathbf{u}^\omega)$ converges to $\widehat{\phi}(\overline{u})^\omega$.

Furthermore, since $\mathbf{u}$ and $\mathbf{v}$ are equivalent, we have $\lim \phi(\mathbf{u}) = \lim \phi(\mathbf{v})$, *i.e.* $\widehat{\phi}(\overline{u}) = \widehat{\phi}(\overline{v})$, so $\widehat{\phi}(\overline{u})^\omega = \widehat{\phi}(\overline{v})^\omega$, *i.e.* $\lim \phi(\mathbf{u}^\omega) = \lim \phi(\mathbf{v}^\omega)$, This implies that $\mathbf{u}^\omega$ and $\mathbf{v}^\omega$ are equivalent. ∎

We can now define polynomial prostochastic words.
First, ω-expressions are described by the following grammar:

$$E \quad \longrightarrow \quad a \quad | \quad E \cdot E \quad | \quad E^\omega.$$

We define an interpretation $\overline{\cdot}$ of ω-expressions into fast prostochastic words:

- $\overline{a}$ is prostochastic word induced by the constant sequence of the one letter word a,
- $\overline{E_1 \cdot E_2} = \overline{E_1} \cdot \overline{E_2}$,
- $\overline{E^\omega} = \overline{E}^\omega$.

Definition 25 (Polynomial prostochastic word)

The set of *polynomial prostochastic words* is $\{\overline{E} \mid E \text{ is an } \omega\text{-expression}\}$.

***Remark* 4.** *Why the term polynomial?*

Consider an ω-expression E, say $(a^\omega b)^\omega$, and the prostochastic word $\overline{(a^\omega b)^\omega}$, which is induced by the sequence of finite words $((a^{f(n)}b)^{f(n)})_{n\in\mathbb{N}}$. Roughly speaking $f(n) \sim n$, so this sequence represents a polynomial behaviour. Furthermore, the proofs above yield the following robustness property: all converging sequences of finite words $((a^{g(n)}b)^{h(n)})_{n\in\mathbb{N}}$, where $g, h : \mathbb{N} \to \mathbb{N}$ are subexponential functions, are equivalent, so they induce the same polynomial prostochastic word $\overline{(a^\omega b)^\omega}$. We say that a function $g : \mathbb{N} \to \mathbb{N}$ is subexponential if for all constants $C > 1$ we have $\lim_n g(n) \cdot C^{-n} = 0$; all polynomial functions are subexponential.

This justifies the terminology; we say that the polynomial prostochastic words represent all polynomial behaviours.

2.4 Powers of a Stochastic Matrix

In this subsection, we prove Theorem 28.

Let $M \in \mathcal{S}_{Q\times Q}(\mathbb{R})$, consider $P = M^{|Q|}$. It is easy to see that $\pi(P)$ is idempotent. We decompose P as illustrated in Figure 19, by indexing states in the following way:

- first, $\pi(P)$-transient states,
- then, $\pi(P)$-recurrent states, grouped by recurrence class.

$$\left(\begin{array}{c:cccc} Q & & R & & \\ \hdashline & J_1 & 0 & & \\ & 0 & J_2 & & J \\ 0 & & & \ddots & \\ & & & & J_r \end{array}\right)$$

Figure 19: Decomposition of P.

In this decomposition, we have the following properties:

- for all m-transient states $s \in Q$, we have $\sum_{t \ m\text{-transient}} Q(s,t) < 1$, so $||Q|| < 1$,
- the matrices J_i are irreducible: for all states $s, t \in Q$ corresponding to the same J_i, we have $J_i(s,t) > 0$.

The power P^n of P is represented in Figure 20.

$$\left(\begin{array}{c:cccc} Q^n & & \sum_{k=0}^{n-1} Q^k \cdot R \cdot J^{n-1-k} & & \\ \hdashline & J_1^n & 0 & & \\ & 0 & J_2^n & & J^n \\ 0 & & & \ddots & \\ & & & & J_r^n \end{array}\right)$$

Figure 20: Decomposition of P^n.

This decomposition allows to treat separately the three blocks:

1. the block Q^n: thanks to the observation above $||Q|| < 1$, which combined with $||Q^n|| \leqslant ||Q||^n$ implies that $(Q^n)_{n \in \mathbb{N}}$ converges to 0 exponentially fast,
2. the block $\sum_{k=0}^{n-1} Q^k \cdot R \cdot J^{n-1-k}$,
3. the block J^n: it is handled by Lemma 19.

We first focus on item 3., and show that the sequence $(J^n)_{n\in\mathbb{N}}$ converges exponentially fast. Each block J_i is handled separately by the following lemma.

Lemma 19 (Powers of an irreducible stochastic matrix)

Let $J \in \mathcal{S}_{Q\times Q}(\mathbb{R})$ irreducible: for all states $s,t \in Q$, we have $J(s,t) > 0$. Then the sequence $(J^n)_{n\in\mathbb{N}}$ converges exponentially fast to a matrix J^∞.

Furthermore, J^∞ is irreducible.

This lemma is a classical result from Markov chain theory, sometimes called "the Convergence Theorem"; see for instance [LPW08].

We now consider item 2., and show that the sequence $(\sum_{k=0}^{n-1} Q^k \cdot R \cdot J^{n-1-k})_{n\in\mathbb{N}}$ converges exponentially fast. Observe that since $||Q|| < 1$, the matrix $I - Q$ is invertible; denote $N = (I-Q)^{-1}$, it is equal to $\sum_{k\geqslant 0} Q^k$. Denote $J^\infty = \lim_n J^n$, which exists thanks to Lemma 19.

We have:

$$\begin{aligned}
&||\sum_{k=0}^{n-1} Q^k \cdot R \cdot J^{n-1-k} - N \cdot R \cdot J^\infty|| \\
&\quad = ||\sum_{k=0}^{n-1} \left[Q^k \cdot R \cdot \left(J^{n-1-k} - J^\infty\right) + Q^k \cdot R \cdot J^\infty\right] - N \cdot R \cdot J^\infty|| \\
&\quad = ||\sum_{k=0}^{n-1} Q^k \cdot R \cdot \left(J^{n-1-k} - J^\infty\right) + \left(\sum_{k=0}^{n-1} Q^k - N\right) \cdot R \cdot J^\infty|| \\
&\quad \leqslant ||\sum_{k=0}^{n-1} Q^k \cdot R \cdot \left(J^{n-1-k} - J^\infty\right)|| + ||\left(\sum_{k=0}^{n-1} Q^k - N\right) \cdot R \cdot J^\infty||.
\end{aligned}$$

We first consider the right term:

$$\begin{aligned}
&||\left(\sum_{k=0}^{n-1} Q^k - N\right) \cdot R \cdot J^\infty|| \\
&\quad = ||\left(\sum_{k\geqslant n} Q^k\right) \cdot R \cdot J^\infty|| \\
&\quad \leqslant ||\sum_{k\geqslant n} Q^k|| \cdot \underbrace{||R||}_{\leqslant 1} \cdot \underbrace{||J^\infty||}_{=1} \\
&\quad = ||Q^n \cdot N|| \\
&\quad \leqslant ||N|| \cdot ||Q||^n.
\end{aligned}$$

Thus, this term converges exponentially fast to 0.

We then consider the left term. Thanks to Lemma 19, there exists a constant $C > 1$ such that for all $p \in \mathbb{N}$, we have $||J^p - J^\infty|| \leqslant C^{-p}$.

$$\begin{aligned}
&|| \sum_{k=0}^{n-1} Q^k \cdot R \cdot \left(J^{n-1-k} - J^{\infty}\right) || \\
&\quad \leqslant \sum_{k=0}^{n-1} ||Q||^k \cdot \underbrace{||R||}_{\leqslant 1} \cdot ||J^{n-1-k} - J^{\infty}|| \\
&\quad \leqslant \sum_{k=0}^{n-1} ||Q||^k \cdot ||J^{n-1-k} - J^{\infty}|| \\
&\quad = \sum_{k=0}^{n/2} \underbrace{||Q||^k}_{\leqslant 1} \cdot ||J^{n-1-k} - J^{\infty}|| + \sum_{k=n/2+1}^{n-1} ||Q||^k \cdot \underbrace{||J^{n-1-k} - J^{\infty}||}_{\leqslant 2} \\
&\quad \leqslant \frac{C^{-(n/2+1)} - C^{-n}}{1-C} + 2 \cdot \frac{||Q||^{n/2+1} - ||Q||^n}{1 - ||Q||} \\
&\quad \leqslant 2 \cdot \left(\frac{C^{-(n/2+1)}}{1-C} + \frac{||Q||^{n/2+1}}{1-||Q||} \right).
\end{aligned}$$

Thus, this term converges exponentially fast to 0.

We proved that $(P^n)_{n \in \mathbb{N}}$ converges exponentially fast to a matrix M^ω. We conclude the proof of Theorem 28 by observing that:

$$\pi(M^\omega)(s,t) = \begin{cases} 1 & \text{if } \pi(P)(s,t) = 1 \text{ and } t \text{ is } \pi(P)\text{-recurrent,} \\ 0 & \text{otherwise.} \end{cases}$$

Assume first that $\pi(M^\omega)(s,t) = 1$, *i.e.* $M^\omega(s,t) > 0$. It already implies that t is $\pi(P)$-recurrent, looking at the decomposition of P^n. Since $M^\omega = \lim_n P^n$, it follows that for n large enough, we have $P^n(s,t) > 0$. The matrix $\pi(P)$ is idempotent, so we have for all $n \in \mathbb{N}$ the equality $\pi(P^n) = \pi(P)$, implying that $P(s,t) > 0$, *i.e.* $\pi(P)(s,t) = 1$.

Conversely, assume that $\pi(P)(s,t) = 1$ and t is $\pi(P)$-recurrent. Observe that for all $n \in \mathbb{N}$ we have $P^{n+1}(s,t) \geqslant P(s,t) \cdot P^n(t,t)$. For n converging towards infinity, this implies $M^\omega(s,t) \geqslant P(s,t) \cdot M^\omega(t,t)$. Note that $P(s,t) > 0$, and $M^\omega(t,t) > 0$ since t is $\pi(P)$-recurrent and thanks to Lemma 19. It follows that $M^\omega(s,t) > 0$, *i.e.* $\pi(M^\omega)(s,t) = 1$.

3 The Markov Monoid Algorithm

In this section, we introduce the Markov Monoid algorithm, which first appeared in [FGO12], in collaboration with Hugo Gimbert and Youssouf Oualhadj.

The definition of the algorithm is given in Subsection 3.1, and an example in Subsection 3.2. Informally speaking, the Markov Monoid algorithm abstracts the behaviour of a probabilistic automaton by a finite stabilisation monoid called the Markov Monoid. We define two properties in Subsection 3.3.

- *consistency*, which states that all behaviours predicted by the Markov Monoid correspond to behaviours of the probabilistic automaton,
- *completeness*, which states that all behaviours of the probabilistic automaton correspond to behaviours of the Markov Monoid.

We show that the consistency property holds, without any further assumption, for all probabilistic automata, in Subsection 3.4. This gives a characterisation of the Markov Monoid algorithm, which is the main result of [Fij15].

The combination of both consistency and completeness would imply that the Markov Monoid algorithm solves the value 1 problem; since this problem is undecidable, the completeness fails in general. We discuss this further in Subsection 3.5.

3.1 The Algorithm

Consider $\mathcal{A}$ a probabilistic automaton, the Markov Monoid algorithm consists in computing, through a saturation process, the Markov Monoid of $\mathcal{A}$.

It is a monoid of Boolean matrices: all numerical values are projected away to Boolean values. So instead of considering $M \in \mathcal{S}_{Q \times Q}(\mathbb{R})$, we are interested in $\pi(M)$. Hence to define the Markov Monoid, one can consider the underlying non-deterministic automaton $\pi(\mathcal{A})$ instead of the probabilistic automaton $\mathcal{A}$.

The Markov Monoid of $\pi(\mathcal{A})$ contains the transition monoid of $\pi(\mathcal{A})$, which is the monoid generated by $\{\pi(\phi(a)) \mid a \in A\}$ and closed under (Boolean matrix) products. Informally speaking, the transition monoid accounts for the Boolean action of every finite word. Formally, for a word $w \in A^*$, the element $\langle w \rangle$ of the transition monoid of $\pi(\mathcal{A})$ satisfies the following: $\langle w \rangle(s,t) = 1$ if, and only if, there exists a run from s to t reading w on $\pi(\mathcal{A})$.

The Markov Monoid generalises the transition monoid by introducing a new operator, the stabilisation. On the intuitive level first: let $M \in \mathcal{S}_{Q \times Q}(\mathbb{R})$, it can be interpreted as a Markov chain; its Boolean projection $\pi(M)$ give the structural properties of this Markov chain. The stabilisation $\pi(M)^\sharp$ accounts for $\lim_n M^n$, *i.e.* the behaviour of the Markov chain M in the limit. The formal definition of the stabilisation operator is as follows:

Definition 26 (Stabilisation)

Let m be a Boolean idempotent matrix.
The stabilisation of m is denoted $m^\sharp$ and defined by:

$$m^\sharp(s,t) = \begin{cases} 1 & \text{if } m(s,t) = 1 \text{ and } t \text{ is } m\text{-recurrent}, \\ 0 & \text{otherwise.} \end{cases}$$

The definition of the stabilisation matches the intuition that in the Markov chain $\lim_n M^n$, the probability to be in non-recurrent states converges to 0.

Definition 27 (Markov Monoid)

The Markov Monoid of $\mathcal{A}$ is the smallest set of Boolean matrices containing $\{\pi(\phi(a)) \mid a \in A\}$ and closed under product and stabilisation of idempotents.

ALGORITHM 1: The Markov Monoid algorithm.

```
Data: A probabilistic automaton.
$\mathcal{M} \leftarrow \{\pi(\phi(a)) \mid a \in A\} \cup \{I\}$.
repeat
    if there is $m, m' \in \mathcal{M}$ such that $m \cdot m' \notin \mathcal{M}$ then
        add $m \cdot m'$ to $\mathcal{M}$
    end
    if there is $m \in \mathcal{M}$ such that $m$ is idempotent and $m^\sharp \notin \mathcal{M}$ then
        add $m^\sharp$ to $\mathcal{M}$
    end
until there is nothing to add;
if there is a value 1 witness in $\mathcal{M}$ then
    return YES;
else
    return NO;
end
```

In proofs and examples, we will often see Boolean matrices as graphs over the set Q; such a graph has an edge between two states $s, t \in Q$ if $m(s,t) = 1$.

The Boolean matrices m handled by the Markov Monoid algorithm have a special property: for all states $s \in Q$, there exists a state $t \in Q$ such that $m(s,t) = 1$. From now on, we only consider Boolean matrices having this property.

On an intuitive level, a Boolean matrix in the Markov Monoid reflects the asymptotic effect of a sequence of finite words.

The Markov Monoid algorithm, detailed in Algorithm 1 computes the Markov Monoid, and looks for *value* 1 *witnesses*:

Definition 28 (Value 1 witness)

A Boolean matrix m is a value 1 witness if: for all states $s, t \in Q$, if $\delta_I(s) > 0$ and $m(s,t) = 1$, then $t \in F$.

The Markov Monoid algorithm answers “YES” if there exists a value 1 witness in the Markov Monoid, and “NO” otherwise.

For proof purposes, we give an equivalent presentation of the Markov Monoid through ω-expressions. Given a probabilistic automaton $\mathcal{A}$, we define an interpretation $\langle\cdot\rangle$ of ω-expressions into Boolean matrices:

- $\langle a \rangle$ is $\pi(\phi(a))$,
- $\langle E_1 \cdot E_2 \rangle$ is $\langle E_1 \rangle \cdot \langle E_2 \rangle$,
- $\langle E^\omega \rangle$ is $\langle E \rangle^\sharp$, only defined if $\langle E \rangle$ is idempotent.

Then the Markov Monoid of $\mathcal{A}$ is $\{\langle E \rangle \mid E \text{ an } \omega\text{-expression}\}$.

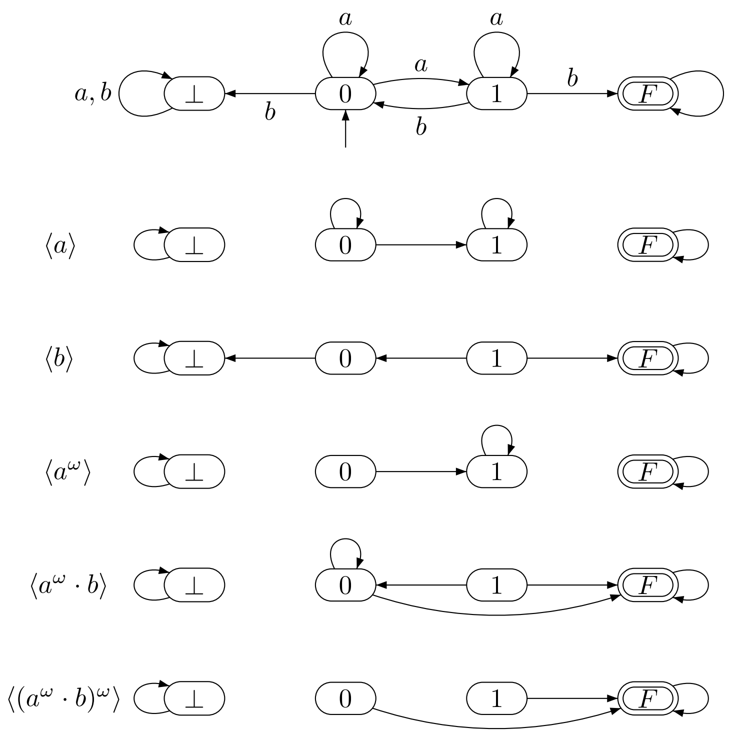
3.2 An Example

We apply the Markov Monoid algorithm on an example. As explained, the Markov Monoid does not take into account the numerical values of the probabilistic transition, so as input we can consider the underlying non-deterministic automaton of a probabilistic automaton. This is what we do in Figure 21: the non-deterministic automaton is represented at the top.

We do not represent here all elements of its Markov Monoid; the tool ACME computed it [FK14], it contains 42 elements. We chose to represent here only 5 elements:

- The first two correspond to the letters a and b.
- The third one is $\langle a^\omega \rangle$, its represents the behaviour of $(a^n)_{n\in\mathbb{N}}$. Indeed, when reading a from the state 0, two events happen with positive probability: looping around the state 0 and going to state 1. So, reading the word a^n from 0 gives a probability to remain in the state 0 converging to 0 when n goes to infinity. Formally, this is reflected by the fact that the state 0 is not $\langle a \rangle$-recurrent.
- The fourth one is $\langle a^\omega \cdot b \rangle$, it illustrates the concatenation between $\langle a^\omega \rangle$ and $\langle b \rangle$. Note that it is not a value 1 witness: from the only initial state 0, we have $\langle a^\omega \cdot b \rangle(0,0) = 1$, and 0 is not final.
- The fifth one is $\langle (a^\omega \cdot b)^\omega \rangle$, it illustrates that stabilisation can be nested. Observe that it is a value 1 witness. This matches the calculation showing that $\lim_n \mathcal{P}_{\mathcal{A}}((a^n \cdot b)^n) = 1$, implying that $\mathcal{A}$ has value 1.

  The pedantic reader may object that $\langle a^\omega \cdot b \rangle$ is not idempotent, so $\langle (a^\omega \cdot b)^\omega \rangle$ is not well defined. He would be right, and would be answered that the element represented here is in fact $\langle ((a^\omega \cdot b)^2)^\omega \rangle$, which does not alter the observations made above.



a, b
a
a
a
b
b
b
$\langle a \rangle$
$\langle b \rangle$
$\langle a^\omega \rangle$
$\langle a^\omega \cdot b \rangle$
$\langle (a^\omega \cdot b)^\omega \rangle$

Figure 21: A non-deterministic automaton and part of its Markov Monoid.

3.3 Properties

Two key properties, *consistency* and *completeness*, state that the Markov monoid reflect exactly every possible asymptotic effect of a sequence of words.

Definition 29 (Reification)

A sequence $(u_n)_{n \in \mathbb{N}}$ of words reifies a Boolean matrix m if for all states $s, t \in Q$, the sequence $(P_\mathcal{A}(s \xrightarrow{u_n} t))_{n \in \mathbb{N}}$ converges and:

$$m(s,t) = 1 \iff \lim_n P_\mathcal{A}(s \xrightarrow{u_n} t) > 0.$$

Note that if $(u_n)_{n \in \mathbb{N}}$ reifies m, then any subsequence of $(u_n)_{n \in \mathbb{N}}$ also does. We will use this simple observation several times.

Definition 30 (Consistency and completeness)

A set of Boolean matrices $\mathcal{M}$ is:

- *consistent* with $\mathcal{A}$ if for every Boolean matrix $m \in \mathcal{M}$, there exists a sequence of words $(u_n)_{n \in \mathbb{N}}$ which reifies m.

- *complete* for $\mathcal{A}$ if for every sequence of words $(u_n)_{n \in \mathbb{N}}$, there exists $m \in \mathcal{M}$ such that for all states $s, t \in Q$:

$$\limsup_n P_{\mathcal{A}}(s \xrightarrow{u_n} t) = 0 \implies m(s,t) = 0.$$

Thanks to value 1 witnesses, the answer to the value 1 problem can be read in a consistent and complete set of Boolean matrices:

Lemma 20 (A criterion for value 1)

If $\mathcal{M}$ is consistent with $\mathcal{A}$ and complete for $\mathcal{A}$, then $\mathcal{A}$ has value 1 if, and only if, $\mathcal{M}$ contains a value 1 witness.

Specifically:

- If $\mathcal{M}$ is consistent with $\mathcal{A}$ and contains a value 1 witness, then $\mathcal{A}$ has value 1,
- If $\mathcal{M}$ is complete for $\mathcal{A}$ and $\mathcal{A}$ has value 1, then $\mathcal{A}$ contains a value 1 witness.

Proof. We prove the first item. Assume that $\mathcal{M}$ is consistent with $\mathcal{A}$ and contains a value 1 witness m. Since $\mathcal{M}$ is consistent, there exists a sequence $(u_n)_{n \in \mathbb{N}}$ reifying m. It follows that for $s \in Q$ and $t \notin F$, we have $\lim_n \delta_I(s) \cdot P_{\mathcal{A}}(s \xrightarrow{u_n} t) = 0$. Since for all $n \in \mathbb{N}$, we have $\sum_{s \in Q} \sum_{t \in Q} \delta_I(s) \cdot P_{\mathcal{A}}(s \xrightarrow{u_n} t) = 1$, this implies $\lim_n P_{\mathcal{A}}(u_n) = \sum_{s \in Q} \sum_{t \in F} \lim_n \delta_I(s) \cdot P_{\mathcal{A}}(s \xrightarrow{u_n} t) = 1$, so $\mathcal{A}$ has value 1.

We now prove the second item. Assume that $\mathcal{M}$ is complete for $\mathcal{A}$ and that $\mathcal{A}$ has value 1. Then there exists a sequence of words $(u_n)_{n \in \mathbb{N}}$ such that $\lim_n P_{\mathcal{A}}(u_n) = 1$, *i.e.* $\lim_n \sum_{s \in Q} \sum_{t \in F} \delta_I(s) \cdot P_{\mathcal{A}}(s \xrightarrow{u_n} t) = 1$. Since for all $n \in \mathbb{N}$, we have $\sum_{s \in Q} \sum_{t \in Q} \delta_I(s) \cdot P_{\mathcal{A}}(s \xrightarrow{u_n} t) = 1$, then for $s \in Q$ such that $\delta_I(s) > 0$ and $t \notin F$, it implies that $\limsup_n P_{\mathcal{A}}(s \xrightarrow{u_n} t) = 0$.

Since $\mathcal{M}$ is complete, there exists a Boolean matrix $m \in \mathcal{M}$ such that for all states $s, t \in Q$:

$$\limsup_n P_{\mathcal{A}}(s \xrightarrow{u_n} t) = 0 \implies m(s,t) = 0.$$

Then m is a value 1 witness: let $s \in Q$ such that $\delta_I(s) > 0$ and $t \in Q$ such that $m(s,t) = 1$, then $\limsup_n P_{\mathcal{A}}(s \xrightarrow{u_n} t) > 0$, hence $t \in F$. ∎

3.4 Consistency

The aim of this section is to prove that the Markov Monoid algorithm is consistent, *i.e.* that for every probabilistic automaton $\mathcal{A}$, the Markov Monoid of $\mathcal{A}$ is consistent with $\mathcal{A}$.

Fix a probabilistic automaton $\mathcal{A}$. Unravelling the definitions, this amounts to associate to every ω-expression E, inducing the element $\langle E \rangle$ of the Markov Monoid, a sequence of words that reifies $\langle E \rangle$. We show here that any sequence of finite words inducing the polynomial prostochastic word $\overline{E}$ reifies $\langle E \rangle$.

It follows that for every ω-expression E, the element $\langle E \rangle$ of the Markov Monoid of $\mathcal{A}$ is a value 1 witness if, and only if, the polynomial prostochastic word $\overline{E}$ is accepted by $\mathcal{A}$. This implies the following characterisation of the Markov Monoid algorithm:

Theorem 29 (Consistency and characterisation of the Markov Monoid algorithm)

For every probabilistic automaton $\mathcal{A}$, the Markov Monoid of $\mathcal{A}$ is consistent with $\mathcal{A}$.

The Markov Monoid algorithm answers "YES" on input $\mathcal{A}$ if, and only if, there exists a polynomial prostochastic word accepted by $\mathcal{A}$.

A direct corollary of the consistency is the absence of false negatives:

Corollary 2 (No false negatives for the Markov Monoid algorithm)

If the Markov Monoid algorithm answers "YES" on input $\mathcal{A}$, then $\mathcal{A}$ has value 1.

Theorem 29 follows from the following proposition.

Proposition 4 (Characterisation of the Markov Monoid algorithm)

For every ω-expression E, for every $\phi : A \to \mathcal{S}_{Q \times Q}(\mathbb{R})$, we have

$$\pi(\widehat{\phi}(\overline{E})) = \langle E \rangle.$$

Consequently, for every probabilistic automaton $\mathcal{A}$:

- any sequence inducing the polynomial prostochastic word $\overline{E}$ reifies $\langle E \rangle$,
- the element $\langle E \rangle$ of the Markov Monoid is a value 1 witness if, and only if, the polynomial prostochastic word $\overline{E}$ is accepted by $\mathcal{A}$.

Proof. We prove the first part of Proposition 4 by induction on the ω-expression E, which essentially amounts to gather the results from Section 2.

The base case is $a \in A$, clear.

The product case: let $E = E_1 \cdot E_2$, and $\phi : A \to \mathcal{S}_{Q \times Q}(\mathbb{R})$.

We prove that $\pi(\widehat{\phi}(\overline{E})) = \langle E \rangle$. By definition $\overline{E} = \overline{E_1} \cdot \overline{E_2}$, so $\widehat{\phi}(\overline{E}) = \widehat{\phi}(\overline{E_1}) \cdot \widehat{\phi}(\overline{E_2})$ because $\widehat{\phi}$ is a morphism, and $\pi(\widehat{\phi}(\overline{E})) = \pi(\widehat{\phi}(\overline{E_1})) \cdot \pi(\widehat{\phi}(\overline{E_2}))$. Also by definition, we have $\langle E \rangle = \langle E_1 \rangle \cdot \langle E_2 \rangle$, so the conclusion follows from the induction hypothesis.

The iteration case: let $E = F^\omega$, and $\phi : A \to \mathcal{S}_{Q \times Q}(\mathbb{R})$.

We prove that $\pi(\widehat{\phi}(\overline{E})) = \langle E \rangle$. By definition, $\overline{E} = \overline{F}^\omega$, so $\widehat{\phi}(\overline{E}) = \widehat{\phi}(\overline{F}^\omega)$, which is equal to $\widehat{\phi}(\overline{F})^\omega$ thanks to Lemma 18. Now, $\pi(\widehat{\phi}(\overline{F})^\omega) = \pi(\widehat{\phi}(\overline{F}))^\sharp$ thanks to Theorem 28. By induction hypothesis, $\pi(\widehat{\phi}(\overline{F})) = \langle F \rangle$, which concludes.

Consider a sequence $\mathbf{u}$ inducing the polynomial prostochastic word $\overline{E}$. Thanks to the first item, $\pi(\widehat{\phi}(\overline{E})) = \langle E \rangle$, so $\pi(\lim \phi(\mathbf{u})) = \langle E \rangle$, which means that $\mathbf{u}$ reifies $\overline{E}$.

Assume that $\langle E \rangle$ is a value 1 witness, *i.e.* for all states $s, t \in Q$, if $\delta_I(s) > 0$ and $\langle E \rangle(s,t) = 1$, then $t \in F$. Then if $\delta_I(s) > 0$ and $t \notin F$, we have $\lim \phi(\mathbf{u})(s,t) = 0$.

Since for all $s \in Q$, we have $\sum_{t \in Q} \lim \phi(\mathbf{u})(s,t) = 1$, it follows that if $\delta_I(s) > 0$, we have $\sum_{t \in F} \lim \phi(\mathbf{u})(s,t) = 1$. Since $\sum_{s \in Q} \delta_I(s) = 1$, this implies $\sum_{s \in Q} \delta_I(s) \sum_{t \in F} \lim \phi(\mathbf{u})(s,t) = 1$, *i.e.* $\lim P_{\mathcal{A}}(\mathbf{u}) = 1$, so the polynomial prostochastic word $\overline{E}$ induced by $\mathbf{u}$ is accepted by $\mathcal{A}$.

Conversely, assume that the polynomial prostochastic word $\overline{E}$ is accepted by $\mathcal{A}$. Since it is induced by $\mathbf{u}$, it follows that $\delta_I \cdot \lim \phi(\mathbf{u}) \cdot F = 1$. Consider two states $s, t \in Q$ such that $\delta_I(s) > 0$ and $\langle E \rangle(s,t) = 1$. It follows that $\pi(\lim \phi(\mathbf{u}))(s,t) = 1$, so $\lim \phi(\mathbf{u})(s,t) > 0$. The combination of $\delta_I(s) > 0$, $\lim \phi(\mathbf{u})(s,t) > 0$ and $\delta_I \cdot \lim \phi(\mathbf{u}) \cdot F = 1$ implies that $t \in F$, hence $\langle E \rangle$ is a value 1 witness. ∎

3.5 No Completeness

In this subsection, we explain why the Markov Monoid algorithm cannot be complete:

- we asserted in Lemma 20 that if the Markov Monoid algorithm is consistent and complete, then it is correct, *i.e.* solves the value 1 problem,
- we asserted in Theorem 29 that the Markov Monoid algorithm is consistent,
- the value 1 problem has been shown to be undecidable in [GO10] (we will revisit the undecidability proof in Subsection 5.1).

The logical implication of these three statements is that the Markov Monoid algorithm is not complete.

A concrete example of a probabilistic automaton $\mathcal{A}$ for which the Markov Monoid is not complete is given by Example 3 from Subsection 1.3. We make three observations about the Markov Monoid of $\mathcal{A}$.

- We showed that $\mathcal{A}$ has value 1 if, and only if, $x > \frac{1}{2}$. Since the Markov Monoid does not depend on the numerical values of the probabilistic transitions, and in particular on x, it cannot detect this, thus it is not complete.
- The Markov Monoid of $\mathcal{A}$ does not contain any value 1 witness. Indeed, the underlying non-deterministic automaton consists of two symmetric parts, left and right. Nothing distinguishes those two parts, except for the numerical values of the probabilistic transitions. It follows that all elements m of the Markov Monoid of $\mathcal{A}$ are also symmetric, and in particular, $m(0, L_2) = m(0, R_2)$, implying that m is not a value 1 witness.
- The sequence of words witnessing that $\mathcal{A}$ has value 1 for $x > \frac{1}{2}$ is $((ba^n)^{2^n})_{n \in \mathbb{N}}$. This sequence is not fast, and induces no polynomial prostochastic words. We will call in Subsection 5.1 such a sequence a two-tier sequence, and show that such sequences are enough to imply the undecidability of the value 1 problem.

4 Leaktight Automata

In this section, we define the class of leaktight automata, which was introduced in [FGO12] with Hugo Gimbert and Youssouf Oualhadj.

The main result of [FGO12] is that the value 1 problem is decidable and PSPACE-complete for leaktight automata. More specifically, the class of leaktight automata is defined by a structural property on the Markov Monoid, which implies that the Markov Monoid algorithm is complete for this class.

We define the extended Markov Monoid algorithm in Subsection 4.1, and the class of leaktight automata in Subsection 4.2, where we prove that the extended Markov Monoid algorithm solves the value 1 problem for leaktight automata. In Subsection 4.3, we show that this problem is actually PSPACE-complete.

In the journal version [FGKO15], which features Edon Kelmendi as a third co-author, we investigate the properties of the class of leaktight automata. In particular, we show that the class of leaktight automata strictly contain hierarchical, $\sharp$-acyclic and simple automata, implying that it is the largest class of probabilistic automata for which the value 1 problem is known to be decidable. We report on these results in Subsection 4.4.

4.1 The Extended Markov Monoid Algorithm

The undecidability of the value 1 problem comes from the necessity to track down vanishing probabilities. One of the phenomena that makes tracking vanishing probabilities difficult are *leaks*. A leak occurs in an automaton when a sequence of words turns a set of states $C \subseteq Q$ into a recurrence class C *on the long run*, but *on the short run*, some of the probability of the recurrence class is "leaking" to a *different* recurrence class.

Such leaks occur in the automaton depicted in the left hand side of Figure 22 with the sequence of words $(a^n b)_{n \in \mathbb{N}}$. As n grows large, the probability to reach L_2 from L_1 while reading the word $a^n b$ vanishes, thus the sets $\{L_1\}$ and $\{L_2\}$ are two different recurrence classes on the long run (*i.e.* asymptotically), however on the short run remains a small yet positive probability to reach L_2 from L_1.

The right hand side of Figure 22 shows the asymptotic behaviour of $(a^n b)_{n \in \mathbb{N}}$.

The automaton in Example 3 from Subsection 1.3 contains two symmetric parts identical to Figure 22, it features one leak on the left hand side and another in the right hand side. As a consequence, the real asymptotic behaviour is complex and depends on the compared speeds of these leaks.

To formalize the notion of leak, we will extend the Markov Monoid, and consider matrices over a three valued semiring, denoted $\mathbb{B}_\varepsilon$, instead of the Boolean semiring. The three values are the two Boolean values 0 and 1, plus a third value ε. The operations $\vee$

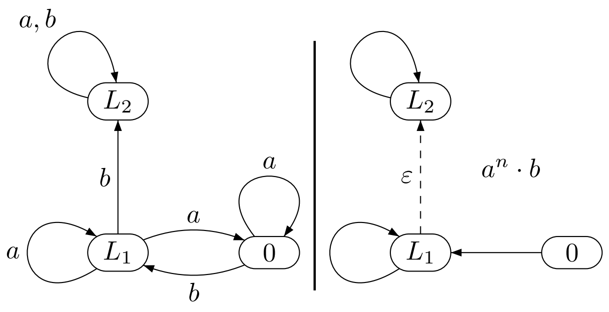


Figure 22: $(a^n \cdot b)_{n \in \mathbb{N}}$ is a leak from L_1 to L_2.

and $\wedge$ are defined by:

$\vee$	0	ε	1
0	0	ε	1
ε	ε	ε	1
1	1	1	1

$\wedge$	0	ε	1
0	0	0	0
ε	0	ε	ε
1	0	ε	1

The matrices over $\mathbb{B}_\varepsilon$ are called extended Boolean matrices, and usually denoted m. As for Boolean matrices, an extended Boolean matrix m can be seen as graphs over the set Q; such a graph has two different kinds of edges, an edge (s,t) is "normal" if $m(s,t) = 1$, and a ε-edge if $m(s,t) = \varepsilon$.

On an intuitive level, the value ε is used for vanishing probabilities.

The semiring structure of $\mathbb{B}_\varepsilon$ induces a monoid structure for $\mathcal{M}_{Q \times Q}(\mathbb{B}_\varepsilon)$. The notion of m-recurrent states is defined as for the Boolean case: a state $q \in Q$ is m-recurrent if for all states $t \in Q$, if $m(s,t) = 1$, then $m(t,s) = 1$. We define the stabilisation operator as follows:

Definition 31 (Stabilisation)

Let m be an extended Boolean idempotent matrix.
The stabilisation of m is denoted $m^\sharp$ and defined by:

$$m^\sharp(s,t) = \begin{cases} 1 & \text{if } m(s,t) = 1 \text{ and } t \text{ is } m\text{-recurrent}, \\ \varepsilon & \text{if } m(s,t) = 1 \text{ and } t \text{ is } m\text{-transient}, \\ \varepsilon & \text{if } m(s,t) = \varepsilon, \\ 0 & \text{otherwise.} \end{cases}$$

Definition 32 (Extended Markov Monoid)

The extended Markov Monoid of $\mathcal{A}$ is the smallest set of extended Boolean matrices containing $\{\pi(\phi(a)) \mid a \in A\}$ and closed under product and stabilisation of idempotents.

Note that initially, the extended Boolean matrices only contain values 0 or 1; the value ε is introduced by the stabilisation operator.

As for the Markov Monoid, for every m in the extended Markov Monoid of a probabilistic automaton, for all states $s \in Q$, there exists a state $t \in Q$ such that $m(s,t) = 1$. From now on, we only consider extended Boolean matrices having this property. This will be crucial in the proof of Lemma 22.

Definition 33 (Value 1 witness)

An extended Boolean matrix m is a value 1 witness if: for all states $s, t \in Q$, if $\delta_I(s) > 0$ and $m(s,t) = 1$, then $t \in F$.

Algorithm 2 computes the extended Markov monoid, and looks for value 1 witnesses.

If there is a value 1 witness, then the algorithm answers "YES": indeed, in such case the automaton has value 1. Otherwise, the algorithm looks for leaks[1]:

- if there are no leaks, then algorithm answers "NO"; indeed, the automaton is leaktight, and it does not have value 1 thanks to Theorem 30,
- if there is a leak, the algorithm answers "FAIL"; indeed, the automaton is not leaktight, and nothing can be said.

ALGORITHM 2: The extended Markov Monoid algorithm.

```
Data: A probabilistic automaton.
$\mathcal{M}_\varepsilon \leftarrow \{\pi(\phi(a)) \mid a \in A\} \cup \{I\}$.
repeat
    if there is $m, m' \in \mathcal{M}_\varepsilon$ such that $m \cdot m' \notin \mathcal{M}_\varepsilon$ then
        add $m \cdot m'$ to $\mathcal{M}_\varepsilon$
    end
    if there is $m \in \mathcal{M}_\varepsilon$ such that $m$ is idempotent and $m^\sharp \notin \mathcal{M}_\varepsilon$ then
        add $m^\sharp$ to $\mathcal{M}_\varepsilon$
    end
until there is nothing to add;
if there is a value 1 witness in $\mathcal{M}_\varepsilon$ then
    return YES;
else
    if there is no leak in $\mathcal{M}_\varepsilon$ then
        return NO;
    else
        return FAIL: the automaton is not leaktight;
    end
end
```

Recall that the Markov Monoid of $\mathcal{A}$ is $\{\langle E \rangle \mid E \text{ an } \omega\text{-expression}\}$. Similarly, the extended Markov Monoid of $\mathcal{A}$ is $\{\langle E \rangle_\varepsilon \mid E \text{ an } \omega\text{-expression}\}$, where $\langle \cdot \rangle_\varepsilon$ is defined using multiplication and stabilisation in $\mathbb{B}_\varepsilon$.

[1]Leaks are special elements of the extended Markov Monoid, see the definition of leaks and leaktight automata in Subsection 4.2.

Lemma 21 (Relation between the extended and the classical Markov Monoid)

For every ω-expression E, the Boolean matrix $\langle E \rangle$ is obtained from the extended Boolean matrix $\langle E \rangle_\varepsilon$ by replacing ε by 0 in all entries.

In particular, $\langle E \rangle$ is a value 1 witness if, and only if, $\langle E \rangle_\varepsilon$ is a value 1 witness.

This lemma is easily proved by induction on ω-expressions. It follows that both algorithms give the same answer. Consequently, if the extended Markov Monoid algorithm answers "YES" on input $\mathcal{A}$, then $\mathcal{A}$ has value 1.

We extend the definition of completeness for the extended Markov Monoid algorithm.

Definition 34 (Completeness for the extended Markov Monoid algorithm)

A set of extended Boolean matrices $\mathcal{M}$ is *complete* for $\mathcal{A}$ if for each sequence of words $(u_n)_{n \in \mathbb{N}}$, there exists $m \in \mathcal{M}$ such that for all states $s, t \in Q$:

$$\limsup_n \mathbb{P}_{\mathcal{A}}(s \xrightarrow{u_n} t) = 0 \implies m(s,t) \neq 1.$$

The same proof as for Lemma 20 shows that if $\mathcal{M}$ is complete for $\mathcal{A}$ and $\mathcal{A}$ has value 1, then $\mathcal{A}$ contains a value 1 witness. Together with the observation above, this implies that if $\mathcal{M}$ is complete for $\mathcal{A}$, then the extended Markov Monoid algorithm correctly determines whether $\mathcal{A}$ has value 1.

4.2 Completeness for Leaktight Automata

In this subsection, we define leaktight automata, and show that the extended Markov Monoid algorithm is complete for leaktight automata.

Definition 35 (Leak)

An extended Boolean idempotent matrix m is a *leak* from the state $s \in Q$ to the state $t \in Q$ if:

1. s and t are m-recurrent,
2. $m(s,t) = \varepsilon$.

Definition 36 (Leaktight automata)

A probabilistic automaton $\mathcal{A}$ is leaktight if its extended Markov Monoid does not contain any leak.

We establish our main result:

Theorem 30 (Completeness for leaktight automata)

For every probabilistic leaktight automaton $\mathcal{A}$, the extended Markov Monoid of $\mathcal{A}$ is complete for $\mathcal{A}$.

Consequently, the extended Markov Monoid algorithm is correct for leaktight automata, implying that the value 1 problem is decidable for leaktight automata.

The remainder of this section is devoted to the proof of Theorem 30. The technical core of the proof relies on an algebraic argument based on the existence of $\sharp$-factorisation trees of bounded height.

Factorisation trees for monoids have been introduced by Simon [Sim90]. Roughly speaking, Simon's factorisation theorem states that given a morphism $\phi : A^* \to M$ from the set of finite words over A into a finite monoid M, the following holds: for all words u, the computation of $\phi(u)$ can be factorised in a tree whose depth is bounded independently of the length of the word.

Simon later developed the notion of decomposition trees to solve the limitedness problem for distance automata [Sim94]. To this end, he defined a stabilisation operator for monoids of matrices over the tropical semiring $(\mathbb{N} \cup \{\infty\}, \min, +)$. Then Kirsten extended this technique to desert automata and nested distance desert automata [Kir05]. After him, Colcombet generalised this approach by defining stabilisation monoids [Col09; Col13b], which are monoids equipped with a stabilisation operator, and proved the existence of $\sharp$-factorisation trees of bounded depth. The formal definition is as follows:

Definition 37 (Stabilisation monoid)

A *stabilisation monoid* $(M, \cdot, \sharp)$ is a finite monoid $(M, \cdot)$ equipped with a stabilisation operator $\sharp : E(M) \to E(M)$, where $E(M)$ is the set of idempotents of M, such that:

$$\begin{aligned} (m \cdot m')^\sharp \cdot m &= m \cdot (m' \cdot m)^\sharp &&\text{for } m \cdot m' \in E(M) \text{ and } m' \cdot m \in E(M), \\ (m^\sharp)^\sharp &= m^\sharp &&\text{for } m \in E(M), \\ m^\sharp \cdot m &= m^\sharp &&\text{for } m \in E(M). \end{aligned}$$

Lemma 22 (The Markov Monoid is a stabilisation monoid)

The extended Markov Monoid of a probabilistic automaton is a stabilisation monoid.

Proof. First of all, the extended Markov monoid is a monoid for the multiplication: the identity matrix I is the neutral element, and the multiplication is associative.

Now, let us prove the three properties required for the stabilisation operator $\sharp$.

First property. Let m, m' such that $m \cdot m'$ and $m' \cdot m$ are idempotent. Let $s, t \in Q$, we have the following equivalence: $\left((m \cdot m')^\sharp \cdot m\right)(s,t) = 1$ if, and only if, there exist two states $r, q \in Q$ such that $m(s,r) = 1$, $m'(r,q) = 1$, q is $(m \cdot m')$-recurrent and $m(q,t) = 1$. Similarly, $\left(m \cdot (m' \cdot m)^\sharp\right)(s,t) = 1$ if, and only if, there exist two states $r, q \in Q$ such that $m(s,r) = 1$, $m'(r,q) = 1$, $m(q,t) = 1$ and t is $(m' \cdot m)$-recurrent.

We show that those two statements are equivalent. Assume that the first holds, and prove that t is $(m' \cdot m)$-recurrent. Consider a state $p \in Q$ such that $(m' \cdot m)(t,p) = 1$. There exists a state $\ell \in Q$ such that $m'(p,\ell) = 1$. Observe that $m(q,t) = 1$, $(m' \cdot m)(t,p) = 1$ and $m'(p,\ell) = 1$, so $(m \cdot m')^2(q,\ell) = 1$. Since $m \cdot m'$ is idempotent, this implies $(m \cdot m')(q,\ell) = 1$. Recall that q is $(m \cdot m')$-recurrent, so $(m \cdot m')(\ell,q) = 1$. Altogether, $m'(p,\ell) = 1$, $(m \cdot m')(\ell,q) = 1$ and $m(q,t) = 1$ imply that $(m' \cdot m)^2(p,t) = 1$. Since $m' \cdot m$ is idempotent, this implies $(m' \cdot m)(p,t) = 1$, so t is $(m' \cdot m)$-recurrent, and we proved that the first statement implies the second. The converse, showing that the second statement implies the first, is very similar and omitted.

Second property. The equality $(m^\sharp)^\sharp = m^\sharp$ follows from the observation that the notions of m-recurrence and $m^\sharp$-recurrence coincide.

Third property. The equality $m^\sharp \cdot m = m^\sharp$ follows from the observation that given two states $r,t \in Q$, if r is m-recurrent and $m(r,t) = 1$, then t is m-recurrent (under the assumption that m is idempotent). ∎

Definition 38 ($\sharp$-factorisation tree)

Consider a stabilisation monoid $(M,\cdot,\sharp)$ and a monoid morphism $\psi : A^* \to M$.

A *$\sharp$-factorisation tree* of a word $u \in A^*$ is a finite unranked ordered tree, whose nodes have labels in $A^* \times M$ and such that:

i) the root is labelled by (u,m), for some $m \in M$,

ii) every internal node with two children (called *multiplication* nodes) labelled by (u_1,m_1) and (u_2,m_2) is labelled by $(u_1 \cdot u_2, m_1 \cdot m_2)$,

iii) every internal node with three or more children (called *stabilisation* nodes) is labelled by $(u_1 \ldots u_n, m^\sharp)$ for some $m \in E(M)$, and its children are labelled by $(u_1,m),\ldots,(u_n,m)$.

iv) every leaf is labelled by $(a,\psi(a))$ where a is a letter, or $(\varepsilon,\psi(\varepsilon))$.

Note that in a factorisation tree, the second label is not always the image of the first component under ψ; indeed, it is an element of the stabilisation monoid $(M,\cdot,\sharp)$ whereas the image of a finite word under ψ is an element of the monoid $(M,\cdot)$. However, the projection of second label into this submonoid (which consists in ignoring the operator $\sharp$) is indeed the image of the first component under ψ.

The following theorem was stated for the tropical semiring in [Sim94], and generalised in [Col09]. A simple proof can be found in [Tor11].

Theorem 31 (Existence of $\sharp$-factorisation trees of bounded depth)

Consider a stabilisation monoid $(M,\cdot,\sharp)$ and a monoid morphism $\psi : A^* \to M$.
Every word $u \in A^*$ has a $\sharp$-factorisation tree whose depth is less than $3 \cdot |M|$.

In the proof of completeness of the extended Markov Monoid algorithm for leaktight automata, the following lemma, called the Lower bound lemma, is instrumental. It associates to every word a $\sharp$-factorisation tree of bounded depth and propagates bounds on the probabilities by induction on the levels of this tree.

Lemma 23 (Lower bound lemma)

Let $\mathcal{A}$ be a leaktight automaton, and $p_{\min}$ the smallest non-zero transition probability of $\mathcal{A}$. Then for all words $u \in A^*$, there exists m in the extended Markov monoid such that, for all states $s, t \in Q$:

$$\begin{aligned} m(s,t) \neq 0 &\iff P_{\mathcal{A}}(s \xrightarrow{u} t) > 0, \\ m(s,t) = 1 &\implies P_{\mathcal{A}}(s \xrightarrow{u} t) \geqslant p_{\min}^{2^{3|Q|^2+1}}. \end{aligned}$$

Proof. Consider a finite word $u \in A^*$; by Theorem 31 applied to the extended Markov Monoid $\mathcal{M}_\varepsilon$ of $\mathcal{A}$, which is a stabilisation monoid thanks to Lemma 22, and the morphism $\psi : A \to M$ defined by $\psi(a) = \pi(\phi(a))$, there exists a $\sharp$-factorisation tree of depth at most $3 \cdot |\mathcal{M}_\varepsilon|$, whose root is labelled by (u, m) for some extended Boolean matrix m of $\mathcal{M}_\varepsilon$.

The depth of a node in this tree is defined in a bottom-up fashion: the leaves have depth zero, and a node has depth one plus the maximum of the depths of its children.

We prove by a bottom-up induction on h that for every node (u, m) of this tree at depth h, for all states $s, t \in Q$:

$$m(s,t) \neq 0 \iff P_{\mathcal{A}}(s \xrightarrow{u} t) > 0, \tag{2.1}$$

$$m(s,t) = 1 \implies P_{\mathcal{A}}(s \xrightarrow{u} t) \geqslant p_{\min}^{2^h}. \tag{2.2}$$

The case $h = 0$ is for leaves. Here, either u is a letter a and $m = \pi(\phi(a))$, or u is the empty word ε and $m = I$. Then both (2.1) and (2.2) hold.

Assume $h > 0$, there are two cases.

First case: a multiplication node labelled by (u, m) with two children labelled by (u_1, m_1) and (u_2, m_2). By definition $u = u_1 \cdot u_2$ and $m = m_1 \cdot m_2$.

We first prove that (2.1) holds. Indeed, for two states $s, t \in Q$, we have $(m_1 \cdot m_2)(s,t) \neq 0$ if, and only if, there exists a state $r \in Q$ such that $m_1(s,r) \neq 0$ and $m_2(r,t) \neq 0$. On the other side, since:

$$P_{\mathcal{A}}(s \xrightarrow{u} t) = \sum_{r \in Q} P_{\mathcal{A}}(s \xrightarrow{u_1} r) \cdot P_{\mathcal{A}}(r \xrightarrow{u_2} t),$$

then $P_{\mathcal{A}}(s \xrightarrow{u} t) > 0$ if, and only if, there exists a state $r \in Q$ such that $P_{\mathcal{A}}(s \xrightarrow{u_1} r) \cdot P_{\mathcal{A}}(r \xrightarrow{u_2} t) > 0$, which is equivalent to $P_{\mathcal{A}}(s \xrightarrow{u_1} r) > 0$ and $P_{\mathcal{A}}(r \xrightarrow{u_2} t) > 0$. We conclude with the induction hypothesis.

Now we prove that (2.2) holds. Consider two states $s, t \in Q$ such that $m(s,t) = 1$. Then there exists a state $r \in Q$ such that $m_1(s,r) = 1$ and $m_2(r,t) = 1$. So:

$$P_{\mathcal{A}}(s \xrightarrow{u} t) \geqslant P_{\mathcal{A}}(s \xrightarrow{u_1} r) \cdot P_{\mathcal{A}}(r \xrightarrow{u_2} t) \geqslant p_{\min}^{2^h} \cdot p_{\min}^{2^h} = p_{\min}^{2^{h+1}},$$

where the second inequality is by induction hypothesis. This completes the proof of (2.2).

Second case: a stabilisation node labelled by $(u, m^\sharp)$ with k sons labelled by $(u_1, m), \ldots, (u_k, m)$. By definition, $u = u_1 \cdots u_k$, and m is idempotent.

The proof that (2.1) holds is similar to the concatenation node case.

Now we prove that (2.2) holds. Consider two states $s, t \in Q$ such that $m^\sharp(s,t) = 1$. Since $k \geqslant 3$:

$$P_\mathcal{A}(s \xrightarrow{u} t) \geqslant P_\mathcal{A}(s \xrightarrow{u_1} t) \cdot \sum_{q \in Q} P_\mathcal{A}(t \xrightarrow{u_2 \cdots u_{k-1}} q) \cdot P_\mathcal{A}(q \xrightarrow{u_k} t).$$

To establish (2.2) we prove that $P_\mathcal{A}(s \xrightarrow{u_1} t) \geqslant p_{\min}^{2^h}$ and for all states $q \in Q$, we have $P_\mathcal{A}(t \xrightarrow{u_2 \cdots u_{k-1}} q) > 0 \implies P_\mathcal{A}(q \xrightarrow{u_k} t) \geqslant p_{\min}^{2^h}$.

Since $m^\sharp(s,t) = 1$, by definition $m(s,t) = 1$ and t is m-recurrent. The induction hypothesis for the node (u_1, m) implies that $P_\mathcal{A}(s \xrightarrow{u_1} t) \geqslant p_{\min}^{2^h}$.

Consider a state $q \in Q$ such that $P_\mathcal{A}(t \xrightarrow{u_2 \cdots u_{k-1}} q) > 0$, we prove that $P_\mathcal{A}(q \xrightarrow{u_k} t) \geqslant p_{\min}^{2^h}$. For that we use the hypothesis that m is not a leak. By induction hypothesis for each child, (2.1) implies that $m^{k-2}(t,q) \neq 0$. Since m is idempotent, $m(t,q) \neq 0$. We argue that $m(q,t) = 1$. Let $\ell \in Q$ an m-recurrent state such that $m(q,\ell) = 1$. Then $m(t,\ell) \neq 0$, and t, ℓ are m-recurrent. Since m is not a leak, and in particular not a leak from t to ℓ, it follows that $m(t,\ell) = 1$, which implies that $m(\ell,t) = 1$ since t is m-recurrent. Together with $m(q,\ell) = 1$, this implies $m(q,t) = 1$. Thus, by induction hypothesis and according to (2.2), $P_\mathcal{A}(q \xrightarrow{u_k} t) \geqslant p_{\min}^{2^h}$.

Now, putting all inequalities together:

$$\begin{aligned} P_\mathcal{A}(s \xrightarrow{u} t) &\geqslant P_\mathcal{A}(s \xrightarrow{u_1} t) \cdot \sum_{q \in Q} P_\mathcal{A}(t \xrightarrow{u_2 \cdots u_{k-1}} q) \cdot P_\mathcal{A}(q \xrightarrow{u_k} t) \\ &\geqslant p_{\min}^{2^h} \cdot \sum_{q \in Q} P_\mathcal{A}(t \xrightarrow{u_2 \cdots u_{k-1}} q) \cdot p_{\min}^{2^h} \\ &= p_{\min}^{2^{h+1}}, \end{aligned}$$

where the last equality holds because $\sum_{q \in Q} P_\mathcal{A}(t \xrightarrow{u_2 \cdots u_{k-1}} q) = 1$. This completes the proof of (2.2).

To conclude, note that $\mathcal{M}_\varepsilon$ has less than $3^{|Q|^2}$ elements. ∎

Relying on the lower bound lemma (Lemma 23), we prove the completeness of the extended Markov monoid for leaktight automata.

Let $\mathcal{A}$ be a leaktight automaton, and $(u_n)_{n \in \mathbb{N}}$ a sequence of finite words. By Lemma 23, for each word u_n there exists an extended Boolean matrix m_n in the extended Markov monoid such that for all states $s, t \in Q$:

$$m_n(s,t) = 1 \implies P_\mathcal{A}(s \xrightarrow{u_n} t) \geqslant p_{\min}^{2^{3^{|Q|^2+1}}}.$$

Since the set of extended Boolean matrices is finite, there exists $N \in \mathbb{N}$ such that $\{n \in \mathbb{N} \mid m_N = m_n\}$ is infinite. To complete the proof, we prove that m_N satisfies, for all states $s, t \in Q$:

$$\limsup P_\mathcal{A}(s \xrightarrow{u_n} t) = 0 \implies m_N(s,t) = 0.$$

Assume $\limsup P_\mathcal{A}(s \xrightarrow{u_n} t) = 0$, then $P_\mathcal{A}(s \xrightarrow{u_n} t) < p_{\min}^{2^{3^{|Q|^2+1}}}$ for n sufficiently large. Since $m_N = m_n$ for infinitely many $n \in \mathbb{N}$, this implies $m_N(s,t) \neq 1$, which completes the proof of Theorem 30.

4.3 Complexity of the Value 1 Problem for Leaktight Automata

In this section, we show two results:

- the value 1 problem for leaktight automaton is PSPACE-hard,
- the extended Markov Monoid can be computed in polynomial space.

As a corollary, we obtain that the value 1 problem for leaktight automata is PSPACE-complete.

4.3.1 PSPACE-hardness

We start with the PSPACE-hardness of the value 1 problem for leaktight automata. To this end, we give a reduction from the emptiness problem of n deterministic automata. To prove that the reduction indeed constructs leaktight automata, we need to show that deterministic automata are leaktight, and the closure under parallel composition.

Lemma 24 (Deterministic automata are leaktight)

Deterministic automata are leaktight.

Proof. For every extended Boolean matrix $m \in \{\pi(\phi(a)) \mid a \in A\} \cup \{I\}$, for all states $s \in Q$, there exists a unique state $t \in Q$ such that $m(s,t) = 1$. In particular, each recurrence class is formed of only one state with a self-loop. This property is preserved by multiplication, and implies that the stabilisation operator is trivial, *i.e.* $m^\sharp = m$. Consequently, all extended Boolean matrices m in the extended Markov Monoid have entries 0 or 1, which implies that there are no leaks. ∎

Definition 39 (Parallel composition)

Let $\mathcal{A}$ and $\mathcal{B}$ two probabilistic automata over the disjoint set of states $Q^\mathcal{A}$ and $Q^\mathcal{B}$, with the same alphabet A.

The parallel composition $\mathcal{A}||\mathcal{B}$ of $\mathcal{A}$ and $\mathcal{B}$ is the probabilistic automaton over the set of states $Q^\mathcal{A} \cup Q^\mathcal{B}$, whose initial distribution is $\delta_I = \frac{1}{2} \cdot \delta_I^\mathcal{A} + \frac{1}{2} \cdot \delta_I^\mathcal{B}$, set of final states is $F = F^\mathcal{A} \cup F^\mathcal{B}$, and transition function is

$$\phi(q,a) = \begin{cases} \phi_\mathcal{A}(q,a) & \text{if } q \in Q_\mathcal{A}, \\ \phi_\mathcal{B}(q,a) & \text{if } q \in Q_\mathcal{B}. \end{cases}$$

By definition, for $u \in A^*$, we have $P_{\mathcal{A}||\mathcal{B}}(u) = \frac{1}{2} \cdot P_\mathcal{A}(u) + \frac{1}{2} \cdot P_\mathcal{B}(u)$.

Lemma 25 (Stability of the leaktight class by parallel composition)

The class is leaktight automata is stable by parallel composition.

Proof. The extended Markov Monoid $\mathcal{M}_\varepsilon^{\mathcal{A}\|\mathcal{B}}$ of the parallel composition embeds into the direct product $\mathcal{M}_\varepsilon^{\mathcal{A}} \times \mathcal{M}_\varepsilon^{\mathcal{B}}$ of the extended Markov Monoids of each automaton.

Note that for $m \in \mathcal{M}_\varepsilon^{\mathcal{A}\|\mathcal{B}}$ and two states $s, t \in Q^{\mathcal{A}} \cup Q^{\mathcal{B}}$, if $m(s,t) = 1$, then either $s, t \in Q^{\mathcal{A}}$ or $s, t \in Q^{\mathcal{B}}$. Relying on this, we map $m \in \mathcal{M}_\varepsilon^{\mathcal{A}\|\mathcal{B}}$ to $(m[\mathcal{A}], m[\mathcal{B}])$, where $m[\mathcal{A}]$ is the restriction to $Q^{\mathcal{A}}$ and similarly for $\mathcal{B}$. An easy induction shows that this map is an embedding into $\mathcal{M}_\varepsilon^{\mathcal{A}} \times \mathcal{M}_\varepsilon^{\mathcal{B}}$.

Consequently, if none of the extended Markov Monoids of $\mathcal{A}$ and $\mathcal{B}$ contain a leak, then neither does the extended Markov Monoid of their parallel composition. ∎

Now that we proved that deterministic automata are leaktight, and the closure under parallel composition, the PSPACE-hardness of the value 1 problem for leaktight automata is easy.

Proposition 5 (PSPACE-hardness of the value 1 problem for leaktight automata)

The value 1 problem for leaktight automaton is PSPACE-hard.

Proof. We give a reduction from the following problem: given n deterministic automata over finite words, decide whether the intersection of the languages they accept is empty. This problem is PSPACE-hard.

The reduction is as follows: given n deterministic automata, we construct the parallel composition of the n automata, where each copy is reached with probability $\frac{1}{n}$. This automaton has value 1 if, and only if, the intersection of the languages is not empty, and is leaktight by Lemma 24 and Lemma 25. ∎

4.3.2 Bounding the $\sharp$-height in the extended Markov Monoid

We now consider the running complexity of the extended Markov Monoid algorithm. A naïve argument shows that it terminates in less than $3^{|Q|^2}$ stabilisations, since each stabilisation adds a new extended Boolean matrix in the monoid and there are at most $3^{|Q|^2}$ different Boolean matrices. This gives an EXPTIME upper bound.

A better complexity can be achieved by looking for a value 1 witness or a leak in a non-deterministic way. The algorithm guesses the witness by its decomposition into multiplications and stabilisations. The key observation, made by Daniel Kirsten [Kir05] in the context of distance desert automata, is that the $\sharp$-height, that is the number of nested applications of the stabilisation operator, can be restricted to at most $|Q|$.

We fix a probabilistic automaton $\mathcal{A}$, and define the $\sharp$-hierarchy inside the extended Markov Monoid $\mathcal{M}_\varepsilon$ of $\mathcal{A}$ as follows:

$$\begin{aligned} S_0 &= \{\pi(\phi(a)) \mid a \in A\}^*, \\ S_{p+1} &= (S_p \cup \{m^\sharp \mid m \in E(S_p)\})^*, \end{aligned}$$

where T^* is the set of extended Boolean matrices obtained as multiplications of extended Boolean matrices in T.

Clearly:

$$S_0 \subseteq S_1 \subseteq S_2 \subseteq \cdots \subseteq \mathcal{M}_\varepsilon$$

Definition 40 (♯-height of a Boolean matrix)

The ♯-height of an extended Boolean matrix $m \in \mathcal{M}_\varepsilon$ is the minimal $p \in \mathbb{N}$ such that $m \in S_p$.

Theorem 32 (Linear bound on the ♯-height)

Every extended Boolean matrix $m \in \mathcal{M}_\varepsilon$ has ♯-height at most $|Q|$, *i.e.* $S_{|Q|-1} = \mathcal{M}_\varepsilon$. We say that the ♯-hierarchy collapses at level $|Q| - 1$.

In the following, we adapt Daniel Kirsten's proof from [Kir05] to the setting of probabilistic automata. Roughly speaking, the proof consists in associating a quantity to each idempotent element of the extended Markov Monoid, and to show the following:

- the quantity is bounded above by $|Q|$ and below by 1,
- the quantity strictly decreases when stabilising an unstable element (*i.e.* such that $m^\sharp \neq m$),
- the quantity does not increase when concatenating.

Let m be an extended Boolean idempotent matrix, we define $\sim_m$ the relation over Q by $s \sim_m t$ if $m(s,t) = 1$ and $m(t,s) = 1$. The relation $\sim_m$ is symmetric, and since m is idempotent, it is transitive. If for some state $s \in Q$ there exists a state $t \in Q$ such that $s \sim_m t$, then $s \sim_m s$ since m is idempotent. Consequently, the restriction of $\sim_m$ to the set

$$Z_m = \{s \in Q \mid s \sim_m s\}$$

is reflexive, *i.e.* $\sim_m$ is an equivalence relation on Z_m. From now on by equivalence class of $\sim_m$ we mean an equivalence class of $\sim_m$ on Z_m. We denote by $[s]_m$ the equivalence class of s, and by $\mathbf{Cl}(m)$ the set of equivalence classes of $\sim_m$. The quantity associated with m is $|\mathbf{Cl}(m)|$, the number of equivalence classes of $\sim_m$, that is the number of non-trivial connected components in the underlying graph of m. Note that $1 \leqslant |\mathbf{Cl}(m)| \leqslant |Q|$.

The following lemma gives two useful observations. We order the three values $0, \varepsilon$ and 1 by $0 < \varepsilon < 1$. Observe that for m, m' two extended Boolean matrices and two states $s, t \in Q$, we have:

$$(m \cdot m')(s,t) = \max_{r \in Q} \min(m(s,r), m'(r,t)).$$

Lemma 26 (Facts about multiplication)

- Let m, m' be two extended Boolean matrices and three states $s, t, r \in Q$. Then $(m \cdot m')(s,t) \geqslant m(s,r) \cdot m'(r,t)$.
- Let m be an idempotent extended Boolean matrix and two states $s, t \in Q$. Then there exists a state $r \in Q$ such that $m(s,t) = m(s,r) \cdot m(r,r) \cdot m(r,t)$.

Proof. The first item follows from the equality above.

Consider now the second item. For all states $r \in Q$, since m is idempotent we have:

$$m(s,t) = m^3(s,t) = \sum_{p,q \in Q} m(s,p) \cdot m(p,q) \cdot m(q,t) \geqslant m(s,r) \cdot m(r,r) \cdot m(r,t).$$

Since m is idempotent, we have $m = m^{|Q|+2}$, so there exist $r_0 = s, \ldots, r_{|Q|+2} = t$ such that $m(s,t) = m(r_0, r_1) \cdot m(r_1, r_2) \cdots m(r_{|Q|+1}, r_{|Q|+2})$. By a counting argument, there exist i, j such that $1 \leqslant i < j \leqslant |Q| + 1$ and $r_i = r_j$, denote it by r. We have:

$$\begin{aligned} m(s,r) &= m^i(s,r) \geqslant m(r_0,r_1) \cdots m(r_{i-1}, r_i), \\ m(r,r) &= m^{j-i}(r,r) \geqslant m(r_i, r_{i+1}) \cdots m(r_{j-1}, r_j), \\ m(r,t) &= m^{|Q|+2-j}(r,t) \geqslant m(r_j, r_{j+1}) \cdots m(r_{|Q|+1}, r_{|Q|+2}). \end{aligned}$$

Hence, $m(s,r) \cdot m(r,r) \cdot m(r,t) \geqslant m(r_0,r_1) \cdot m(r_1,r_2) \cdots m(r_{|Q|+1}, r_{|Q|+2}) = m(s,t)$, and the second item follows. ∎

The following lemma shows that the quantity $|\mathbf{Cl}(m)|$ strictly decreases when stabilising an unstable extended Boolean matrix.

Lemma 27 ($|\mathbf{Cl}(\cdot)|$ decreases with stabilisation)

Let m be an extended Boolean idempotent matrix.

- $\mathbf{Cl}(m^\sharp) \subseteq \mathbf{Cl}(m)$,
- if $m^\sharp \neq m$, then $\mathbf{Cl}(m^\sharp) \neq \mathbf{Cl}(m)$.

Proof. We prove the first item. Let $s \in Z_{m^\sharp}$, we show that $[s]_{m^\sharp} = [s]_m$. For all states $t \in Q$ such that $s \sim_{m^\sharp} t$, we have $s \sim_m t$, so $[s]_{m^\sharp} \subseteq [s]_m$. Conversely, let $t \in [s]_m$; we have $m(s,t) = 1$ and $m(t,s) = 1$. Since $s \sim_{m^\sharp} s$, we have $m^\sharp(s,s) = 1$, so in particular s is m-recurrent. It follows from $m(s,t) = 1$ that t is also m-recurrent, so $m^\sharp(s,t) = 1$, and similarly $m^\sharp(t,s) = 1$. Thus $s \sim_{m^\sharp} t$, *i.e.* $t \in [s]_{m^\sharp}$, which implies the equality $[s]_{m^\sharp} = [s]_m$. In other words, the equivalence classes for $m^\sharp$ are also equivalence classes for m, so $\mathbf{Cl}(m^\sharp) \subseteq \mathbf{Cl}(m)$.

We now prove the second item. Assume $m \neq m^\sharp$; it implies that there exist two states $s, t \in Q$ such that $m^\sharp(s,t) = \varepsilon$, $m(s,t) = 1$ and t is m-transient. By Lemma 26, there exists r such that $m(s,t) = m(s,r) \cdot m(r,r) \cdot m(r,t)$, so $m(s,r) = m(r,r) = m(r,t) = 1$. Note that r is m-transient, as $m(r,t) = 1$ would imply that t is m-recurrent. So $m^\sharp(r,r) = \varepsilon$. Thus, $r \in Z_m$ but $r \notin Z_{m^\sharp}$. Hence, there is a class $[r]_m$ in $\mathbf{Cl}(m)$, but there is no class $[r]_{m^\sharp}$ in $\mathbf{Cl}(m^\sharp)$. ∎

The following lemma shows that the quantity $|\mathbf{Cl}(m)|$ is common to all idempotents in the same $\mathcal{J}$-class. The notion of $\mathcal{J}$-class is a classical notion for the theory of monoids, derived from one of the four Green's relations called the $\mathcal{J}$-preorder (for details about Green's relations and its applications to automata theory see [Col11]).

Define $m \leqslant_{\mathcal{J}} m'$ if there exist two extended Boolean matrices n, n' such that $n \cdot m' \cdot n' = m$, and $m \mathcal{J} m'$, *i.e.* m and m' are in the same $\mathcal{J}$-class, if $m \leqslant_{\mathcal{J}} m'$ and $m' \leqslant_{\mathcal{J}} m$.

Lemma 28 ($|\mathbf{Cl}(\cdot)|$ decreases with multiplication)

Consider two extended Boolean idempotent matrices m, m'. If $m \leqslant_{\mathcal{J}} m'$, then $|\mathbf{Cl}(m)| \leqslant |\mathbf{Cl}(m')|$.

Proof. Let n, n' be two extended Boolean idempotent matrices such that $m \leqslant_{\mathcal{J}} m'$; consider two extended Boolean matrices n, n' such that $n \cdot m' \cdot n' = m$. First, without loss of generality we assume that $n \cdot m' = n$ and $m' \cdot n' = n'$. Indeed, if n or n' do not satisfy these conditions, then we consider $n = n \cdot m'$ and $n' = m' \cdot n'$.

We construct a partial surjective mapping $\beta : \mathbf{Cl}(m') \to \mathbf{Cl}(m)$, which depends on the choice of n and n'. For all states $s \in Z_{m'}$ and $t \in Z_m$ satisfying $n(t,s) \cdot m'(s,s) \cdot n'(s,t) = 1$ we set $\beta([s]_{m'}) = [t]_m$. To complete the proof, we have to show that β is well defined and that β is indeed surjective.

We show that β is well defined. Let $s, s' \in Z_{m'}$ and $t, t' \in Z_m$, and assume $n(t,s) \cdot m'(s,s) \cdot n'(s,t) = 1$ and $n(t',s') \cdot m'(s',s') \cdot n'(s',t') = 1$. By definition $\beta([s]_{m'}) = [t]_m$ and $\beta([s']_{m'}) = [t']_m$. To show that β is well defined, we have to show that if $[s]_{m'} = [s']_{m'}$, then $[t]_m = [t']_m$. Assume $[s]_{m'} = [s']_{m'}$, *i.e.*, $s \sim_{m'} s'$, so $m'(s,s') = 1$. Since $n(t,s) \cdot m'(s,s) \cdot n'(s,t) = 1$, we have $n(t,s) = n'(s,t) = 1$. Similarly, $n(t',s') \cdot m'(s',s') \cdot n'(s',t') = 1$, so $n(t',s') = n'(s',t') = 1$. Consequently, $n(t,s) \cdot m'(s,s') \cdot n'(s',t') = 1$, so $m(t,t') = (n \cdot m' \cdot n')(t,t') = 1$. Symmetrically, $n(t',s') \cdot m'(s',s) \cdot n'(s,t) = 1$, so $m(t',t) = 1$, implying $t \sim_m t'$, *i.e.* $[t]_m = [t']_m$.

We show that β is surjective. Let $t \in Z_m$. We exhibit some state $s \in Q$ such that $\beta([s]_{m'}) = [t]_m$. Since $m(t,t) = 1$ and $m = n \cdot m' \cdot n'$, there exist two states $p, q \in Q$ such that $n(t,p) = m'(p,q) = n'(q,t) = 1$. By Lemma 26, there exists a state $s \in Q$ such that $m'(p,s) \cdot m'(s,s) \cdot m'(s,q) = m'(p,q) = 1$, so $m'(p,s) = m'(s,s) = m'(s,q) = 1$, implying $s \in Z_{m'}$. We have $n(t,s) = (n \cdot m')(t,s) \geqslant n(t,p) \cdot m'(p,s) = 1$, and $n'(s,t) = (m' \cdot n')(s,t) \geqslant m'(s,q) \cdot n'(q,t) = 1$. To sum up, $n(t,s) \cdot m'(s,s) \cdot n'(s,t) = 1$, and hence, $\beta([s]_{m'}) = [t]_m$. ∎

The following lemma wraps up the previous two lemma. For technical convenience, we set $S_{-1} = \varnothing$.

Lemma 29 ($|\mathbf{Cl}(\cdot)|$ decreases with the $\sharp$-height)

Let m be an extended Boolean idempotent matrix and $p \geqslant 0$. If $m \in S_p \backslash S_{p-1}$, then $|\mathbf{Cl}(m)| \leqslant |Q| - p$.

Proof. We proceed by induction on p. For $p = 0$, the assertion is obvious. Let $p \geqslant 0$, we show the claim for $p+1$. Let m be an extended Boolean idempotent matrix such that $m \in S_{p+1} \backslash S_p$. By definition, $m = m'_1 \cdots m'_k$ where for all i, either $m'_i \in S_p$ or $m'_i = m_i^\sharp$ for $m_i \in S_p$ and $m'_i \notin S_p$.

If for all i, $m'_i \in S_p$, then $m = m'_1 \cdots m'_k \in S_p$, which is a contradiction. Consequently, there exists i such that $m'_i = m_i^\sharp$ for $m_i \in S_p$ and $m'_i \notin S_p$. Since $m_i \in S_p$ and $m'_i = m_i^\sharp \notin S_p$, we have $m_i^\sharp \neq m_i$. Towards contradiction, assume $m_i \in S_{p-1}$, then $p \geqslant 1$, and this implies $m_i^\sharp \in S_p$, which is a contradiction. Hence, $m_i \in S_p \backslash S_{p-1}$.

By induction, we have $|\mathbf{Cl}(m_i)| \leqslant |Q| - p$. Since $m_i^\sharp \neq m_i$, by Lemma 27 we have $|\mathbf{Cl}(m_i^\sharp)| < |\mathbf{Cl}(m_i)|$. Since $m \leqslant_{\mathcal{J}} m_i^\sharp$, by Lemma 28 we have $|\mathbf{Cl}(m)| \leqslant |\mathbf{Cl}(m_i^\sharp)|$. Altogether, it follows $|\mathbf{Cl}(m)| \leqslant |Q| - (p+1)$. ∎

It follows from Lemma 29 that $S_{|Q|-1} = S_{|Q|} = \mathcal{M}_\varepsilon$, *i.e.* the $\sharp$-hierarchy collapses at level $|Q|-1$, proving Theorem 32.

The bound is almost tight, as shown in Figure 23. The only value 1 witness of this automaton is $(\cdots((a_0^\sharp\ a_1)^\sharp\ a_2)^\sharp\ a_3)^\sharp\ \cdots a_{n-1})^\sharp$, whose $\sharp$-height is $n = |Q|-2$. This

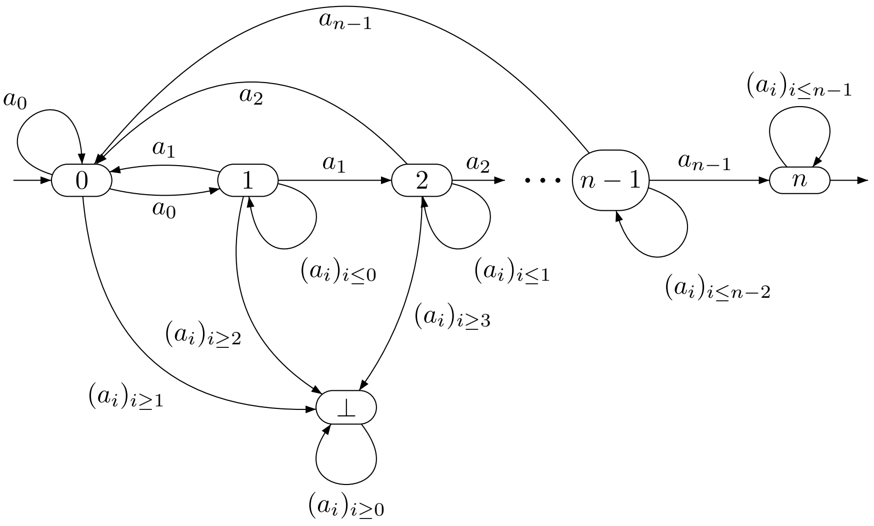


Figure 23: A leaktight automaton with value 1 and $\sharp$-height $|Q|-2$.

automaton is leaktight, so the extended Markov Monoid algorithm will find the value 1 witness and correctly answers that it has value 1.

We derive from Theorem 32 the following complexity result, improving over the naïve approach to implement the extended Markov Monoid algorithm.

Theorem 33 (The extended Markov Monoid algorithm in polynomial space)

There exists an algorithm which checks in polynomial space whether an automaton is leaktight and whether in such case it has value 1.

Following Theorem 30, checking whether a leaktight automaton has value 1 boils down to finding a value 1 witness in the extended Markov Monoid. Similarly, checking whether an automaton is *not* leaktight boils down to finding a leak in the extended Markov Monoid. Note that in both cases, checking whether a given extended Boolean matrix is a witness is easily done in polynomial time.

Since we aim at proving that those two tasks can be computed in PSPACE, which is closed under complementation, it suffices to show how to find a witness in the extended Markov Monoid.

We describe an algorithm to guess a witness in the extended Markov Monoid. The key property given by Theorem 32 is that we can restrict ourselves to at most $|Q|$ nested stabilisations.

As the corresponding property was proved by Daniel Kirsten [Kir05] in the context of distance automata, also to obtain a PSPACE algorithm, the following algorithm is also an adaptation of [Kir05]. Rather than a formal proof, we here give an intuitive description of the algorithm.

A witness can be described as a tree whose nodes are labelled by extended Boolean matrices, of depth at most $2 \cdot |Q| + 1$, as follows:

- a leaf is labelled either by $\pi(\phi(a))$ for $a \in A$ or by I,
- an internal node can be a *multiplication node*, then it is labelled by $m = m_1 \cdots m_k$ for $k \leqslant 3^{|Q|^2}$ and has k children, labelled by $m_1, \ldots, m_k$,
- an internal node can be a *stabilisation node*, then it is labelled by $m^\sharp$ and has one child labelled m.

We describe an algorithm that guesses such a tree. It starts from the root, and travels over nodes in a depth-first way: from top to bottom (and up again) and from left to right. In a node, the algorithm stores the branch that leads to this node, and for each node in the branch the extended Boolean matrix obtained by multiplying all the left siblings of this node. From a node, the algorithm guesses an extended Boolean matrix, and whether it will be a leaf, a multiplication node or an stabilisation node. In the first case, it goes up and checks the consistency of this guess. In the two other cases, it updates the value of this node by multiplying the new guess with the previous value and goes down.

Although the tree is of exponential size, in each step the algorithm only stores $2 \cdot |Q| + 1$ extended Boolean matrices at most, so it runs in polynomial space.

4.4 Comparisons with the Other Classes

In this section, we report on the results obtained in the journal version [FGKO15], in collaboration with Hugo Gimbert, Edon Kelmendi and Youssouf Oualhadj. In addition to the results of the conference version [FGO12], the aim of this journal version is to compare the class of leaktight automata to other subclasses.

Two subclasses of probabilistic automata were constructed in order to decide the value 1 problem on such instances. The first class was the $\sharp$-acyclic automata by Hugo Gimbert and Youssouf Oualhadj [GO10]. Later but concurrently, two different works have been published in the very same conference. The first one introduces simple automata and structurally simple automata, by Krishnendu Chatterjee and Mathieu Tracol [CT12], and the second is ours (in collaboration with Hugo Gimbert and Youssouf Oualhadj), introducing leaktight automata [FGO12].

Although geared towards the same goal (deciding the value 1 problem), the two classes came from different perspectives. The paper of Krishnendu Chatterjee and Mathieu Tracol relies on a theorem from Probability Theory, called the jet decompositions of (infinite) Markov Chains. Ours relies on Simon's theorem and comes from Algebra.

Theorem 34 (Inclusion of the different subclasses)

The class of $\sharp$-acyclic, simple and structurally simple automata are all strictly included in the class of leaktight automata.

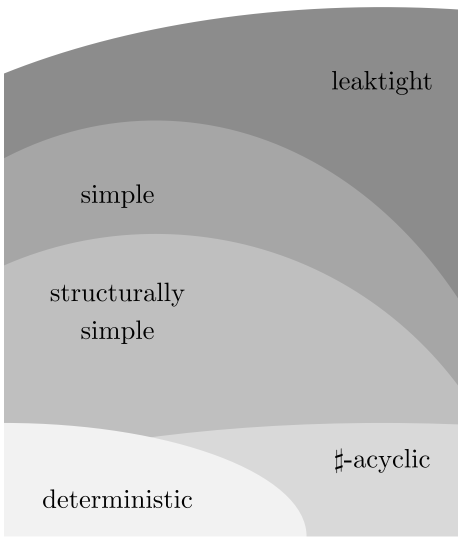


Figure 24: Inclusion of the different subclasses.

We do not give here more details, and refer to [FGKO15] for both definitions and proofs. The content of Theorem 34 is made a bit more precise in Figure 24.

Another subclass of probabilistic automata, called hierarchical automata, has been defined in [CSV11]. The main result is that hierarchical automata recognize exactly the class of ω-regular languages. Unfortunately, the emptiness problem, which is undecidable for the general class of probabilistic automata, is still undecidable for hierarchical automata.

Theorem 35 (Inclusion of the hierarchical automata)

The class of hierarchical automata is strictly included in the class of leaktight automata.

Consequently, the value 1 problem is decidable for hierarchical automata, and the emptiness problem is undecidable for leaktight automata.

5 Undecidability Results

In this section, we give three undecidability results:

- the first is the undecidability of the two-tier value 1 problem,
- the second is the undecidability of the numberless value 1 problems, as proved in [FGHO14].
- the third is the undecidability of the robustness variants of the value 1 and the emptiness problem.

These results aim at better understanding the limits of what can be done about the value 1 problem, *i.e.* to what extend is this problem undecidable.

The proof of the first result is different from the original proof from [GO10]. It shows that the undecidability appears when considering different convergence speeds.

The second result departs from the other undecidability results as it does not consider probabilistic automata, but their numberless counterparts, *i.e.* the underlying non-deterministic automata. In other words, this result shows that even abstracting away the numerical values of the probabilistic transitions does not lead to decidability.

We then introduce robustness problems, in which a probabilistic automaton is given, but its probabilistic transitions can be modified up to a small perturbation. A decision problem for probabilistic automaton induces a robustness variant, where perturbations are allowed. We obtain as a corollary of the undecidability of the numberless problems that robustness problems are also undecidable.

5.1 Undecidability of the Two-tier Value 1 Problem

In this subsection, we revisit the undecidability proof of the value 1 problem. The undecidability result, stated as Theorem 23 in Subsection 1.4, is due to Gimbert and Oualhadj [GO10]. We give here an a finer analysis of the reduction, which yields the undecidability of a simpler variant, called the two-tier value 1 problem. We will further discuss this undecidability result in Section 6.

Theorem 36 (Undecidability of the two-tier value 1 problem)

The following problem is undecidable: given a simple probabilistic automaton $\mathcal{A}$, determine whether there exist two finite words u, v such that $\lim_n P_{\mathcal{A}}((u \cdot v^n)^{2^n}) = 1$.

Note that the two-tier value 1 problem is a priori much easier than the value 1 problem, as it restricts the set of sequences of finite words to very simple sequences. We call such sequences two-tier, because they exhibit two different behaviours: the word v is repeated

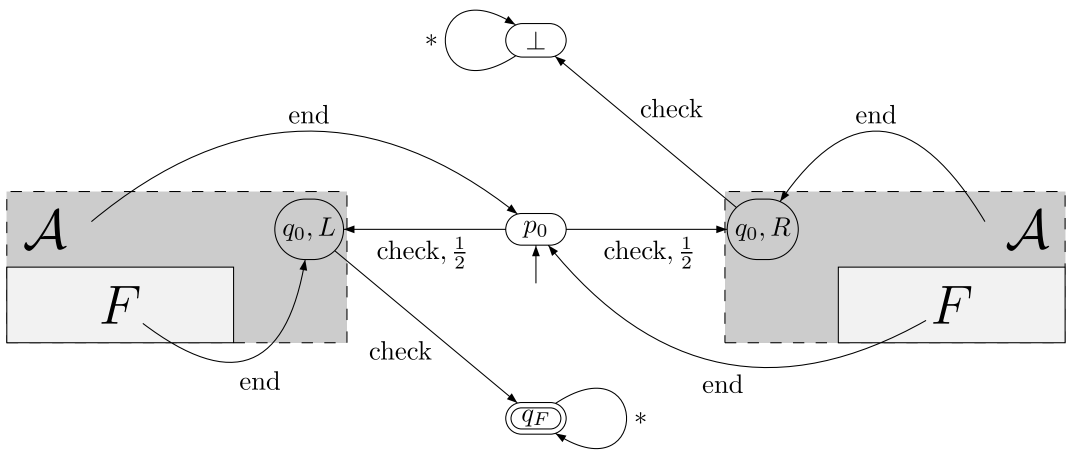


Figure 25: Reduction.

a linear number of times, namely n, while the word $u \cdot v^n$ is repeated an exponential number of times, namely 2^n.

The reduction uses the probabilistic automaton described in Example 3 from Subsection 1.3; the calculations made in the proof to follow and in this example are very similar.

Proof. We construct a reduction from the emptiness problem for simple probabilistic automata to the two-tier value 1 problem. Let $\mathcal{A}$ be a simple probabilistic automaton. We construct a simple probabilistic automaton $\mathcal{B}$ such that the following holds:

$$\begin{gathered} L^{>\frac{1}{2}}(\mathcal{A}) \text{ is non-empty if, and only if,} \\ \text{there exist two finite words } u, v \text{ such that } \lim_n P_{\mathcal{B}}((u \cdot v^n)^{2^n}) = 1. \end{gathered}$$

Without loss of generality we assume that the initial state q_0 has no ingoing transitions.

The alphabet of $\mathcal{B}$ is $B = A \uplus \{\text{check}, \text{end}\}$, its set of states is $Q_{\mathcal{B}} = Q \times \{L, R\} \uplus \{p_0, \bot, q_F\}$, its transition function is ϕ', the only initial state is p_0 and the only final state is q_F. We describe ϕ' as a function $\phi' : Q_{\mathcal{B}} \times B \to \mathcal{D}(Q_{\mathcal{B}})$:

$$\begin{cases} \phi'(p_0, \text{check}) & = \frac{1}{2} \cdot (q_0, L) + \frac{1}{2} \cdot (q_0, R) \\ \phi'((q, d), a) & = (\phi(q, a), d) \text{ for } a \in A \text{ and } d \in \{L, R\} \\ \phi'((q_0, L), \text{check}) & = q_F \\ \phi'((q, L), \text{end}) & = q_0 \text{ if } q \in F \\ \phi'((q, L), \text{end}) & = p_0 \text{ if } q \notin F \\ \phi'((q_0, R), \text{check}) & = \bot \\ \phi'((q, R), \text{end}) & = p_0 \text{ if } q \in F \\ \phi'((q, R), \text{end}) & = q_0 \text{ if } q \notin F \\ \phi'(q_F, *) & = q_F \end{cases}$$

where as a convention, if a transition is not defined, it leads to $\bot$.

Assume that there exists a finite word w such that $P_{\mathcal{A}}(w) > \frac{1}{2}$, then we claim that $\lim_n P_{\mathcal{B}}((\text{check} \cdot (w \cdot \text{end})^n)^{2^n}) = 1$. Denote $x = P_{\mathcal{A}}(w)$.

We have

$$P_{\mathcal{A}}(p_0 \xrightarrow{\text{check}\cdot(w\cdot\text{end})^n} (q_0, L)) = \frac{1}{2} \cdot x^n,$$

and

$$P_{\mathcal{A}}(p_0 \xrightarrow{\text{check}\cdot(w\cdot\text{end})^n} (q_0, R)) = \frac{1}{2} \cdot (1-x)^n.$$

We fix an integer N and analyse the action of reading $(\text{check} \cdot (w \cdot \text{end})^n)^N$: there are N "rounds", each of them corresponding to reading $\text{check} \cdot (w \cdot \text{end})^n$ from p_0. In a round, there are three outcomes: winning (that is, remaining in (q_0, L)) with probability $p_n = \frac{1}{2} \cdot x^n$, losing (that is, remaining in (q_0, R)) with probability $q_n = \frac{1}{2} \cdot (1-x)^n$, or going to the next round (that is, reaching p_0) with probability $1 - (p_n + q_n)$. If a round is won or lost, then the next check leads to an accepting or rejecting sink; otherwise it goes on to the next round, for N rounds. Hence:

$$\begin{aligned}
&P_{\mathcal{A}}((\text{check} \cdot (w \cdot \text{end})^n)^N) \\
&= \sum_{i=1}^{N-1} (1 - (p_n + q_n))^{i-1} \cdot p_n \\
&= p_n \cdot \frac{1 - (1 - (p_n + q_n))^{N-1}}{1 - (1 - (p_n + q_n))} \\
&= \frac{1}{1 + \frac{q_n}{p_n}} \cdot \left(1 - (1 - (p_n + q_n))^{N-1}\right)
\end{aligned}$$

We now set $N = 2^n$. A simple calculation shows that the sequence $((1 - (p_n + q_n))^{N-1})_{n\in\mathbb{N}}$ converges to 0 as n goes to infinity. Furthermore, $\frac{q_n}{p_n} = (\frac{1-x}{x})^n$, which converges to 0 as n goes to infinity since $x > \frac{1}{2}$. It follows that the acceptance probability converges to 1 as n goes to infinity. Consequently:

$$\lim_n P_{\mathcal{A}}((\text{check} \cdot (w \cdot \text{end})^n)^{2^n}) = 1.$$

Conversely, assume that for all finite words w, we have $P_{\mathcal{A}}(w) \leqslant \frac{1}{2}$. We claim that every finite word in B^* is accepted by $\mathcal{B}$ with probability at most $\frac{1}{2}$. First of all, using simple observations we restrict ourselves to words of the form

$$w = \text{check} \cdot w_1 \cdot \text{end} \cdot w_2 \cdot \text{end} \cdots\ w_n \cdot \text{end} \cdot w',$$

with $w_i \in A^*$ and $w' \in B^*$. Since $P_{\mathcal{A}}(w_i) \leqslant \frac{1}{2}$ for every i, it follows that in $\mathcal{B}$, after reading the last letter end in w before w', the probability to be in (q_0, L) is smaller or equal than the probability to be in (q_0, R). This implies the claim. It follows that the value of $\mathcal{B}$ is not 1, and in particular for two finite words u, v, we have $\lim_n P_{\mathcal{B}}((u \cdot v^n)^{2^n}) < 1$. ∎

5.2 Undecidability of the Numberless Value 1 Problems

In this subsection, we report on results obtained in collaboration with Hugo Gimbert, Florian Horn and Youssouf Oualhadj, published in [FGHO14].

The main contribution is an undecidability result for numberless probabilistic automata, which are probabilistic automata where the numerical values of the probabilistic transitions are not specified. The motivation for introducing this model was to obtain a decidable variant of the value 1 problem, as the undecidability proof seemed to strongly rely on numerical manipulations. The negative result presented here shows that these numerical manipulations can be encoded in the structure of the automaton.

Definition 41 (Numberless probabilistic automaton)

A *numberless probabilistic automaton* is given by a finite set of states Q, a numberless transition function $T \subseteq Q \times A \times Q$, an initial state $q_0 \in Q$ and a set of final states $F \subseteq Q$.

The numberless transition function T is an abstraction of transition functions. We say that a transition function $\phi : A \to \mathcal{S}_{Q \times Q}(\mathbb{Q})$ is consistent with T if for all letters $a \in A$ and states $s, t \in Q$, we have $\phi(a)(s,t) > 0$ if, and only if, $(s,a,t) \in T$.

A numberless probabilistic automaton $\mathcal{M}$ together with a transition function ϕ consistent with T defines a probabilistic automaton $\mathcal{M}[\phi]$. We say that $\mathcal{M}[\phi]$ is an *instance* of $\mathcal{M}$. Conversely, a probabilistic automaton $\mathcal{A}$ induces an underlying numberless probabilistic automaton $[\mathcal{A}]$, where T is defined by $(q,a,p) \in T$ if $\phi(a)(q,p) > 0$.

We consider two variants of the value 1 problem for numberless probabilistic automata:

Problem 8 (Numberless value 1 problem)

- **The existential value 1 problem:** given a numberless probabilistic automaton $\mathcal{M}$, determine whether there exists ϕ such that $\text{val}(\mathcal{M}[\phi]) = 1$.
- **The universal value 1 problem:** given a numberless probabilistic automaton $\mathcal{M}$, determine whether for all ϕ, we have $\text{val}(\mathcal{M}[\phi]) = 1$.

There exists a numberless probabilistic automaton $\mathcal{M}$ such that:

- there exists ϕ such that $\text{val}(\mathcal{M}[\phi]) = 1$,
- there exists ϕ' such that $\text{val}(\mathcal{M}[\phi']) < 1$.

Indeed, an example is given by Example 3 from Subsection 1.3: we proved that for $x > \frac{1}{2}$, it has value 1, but not for $x \leqslant \frac{1}{2}$.

The main contribution of [FGHO14] is the following undecidability result.

Theorem 37 (Recursive Inseparability of the numberless value 1 problems)

There exists no algorithm with the following behaviour.
Given a numberless probabilistic automaton $\mathcal{M}$:

- if all instances have value 1, then the algorithm answers “YES”,
- if no instance has value 1, then the algorithm answers “NO”,
- otherwise, the algorithm can do anything, including not terminating.

The remainder of this subsection is devoted to proving this result. Note that it in particular implies that both the universal and the existential value 1 problems are undecidable.

5.2.1 Overall construction

Proposition 6 (Construction)

There exists an effective construction which takes as input a simple probabilistic automaton $\mathcal{A}$ and constructs a numberless probabilistic automaton $\mathcal{M}$ such that

$$\mathrm{val}(\mathcal{A}) = 1 \iff \forall \phi, \mathrm{val}(\mathcal{M}[\phi]) = 1 \iff \exists \phi, \mathrm{val}(\mathcal{M}[\phi]) = 1.$$

We first explain how Proposition 6 implies Theorem 37. Assume towards contradiction that there exists an algorithm A as described in the statement of the theorem. We show using Proposition 6 that this would imply that the value 1 problem is decidable for probabilistic automata. Indeed, let $\mathcal{A}$ be a simple probabilistic automaton, applying the construction yields a numberless probabilistic automaton $\mathcal{M}$ such that

$$\mathrm{val}(\mathcal{A}) = 1 \iff \forall \phi, \mathrm{val}(\mathcal{M}[\phi]) = 1 \iff \exists \phi, \mathrm{val}(\mathcal{M}[\phi]) = 1.$$

In particular, either all instances of $\mathcal{M}$ have value 1, or no instance of $\mathcal{M}$ has value 1. So the algorithm A on input $\mathcal{M}$ terminates, and:

- if it answers "YES" then $\mathrm{val}(\mathcal{A}) = 1$,
- if it answers "NO" then $\mathrm{val}(\mathcal{A}) < 1$,

allowing to determine whether $\mathcal{A}$ has value 1 or not.

We now give a high level description of the construction; it follows two steps.

The first step is to build from $\mathcal{A}$ a family of probabilistic automata $\mathcal{B}_\lambda$ such that for all $0 < \lambda < 1$:

- the transition function of $\mathcal{B}_\lambda$ is $\phi_\lambda : A \to \mathcal{S}_{Q \times Q}(\{0, \lambda, 1-\lambda, 1\})$,
- $\mathrm{val}(\mathcal{B}_\lambda) = \mathrm{val}(\mathcal{A})$.

The second step is to build from the $\mathcal{B}_\lambda$'s a numberless probabilistic automaton $\mathcal{M}$ such that, for each probabilistic transition function ϕ, there exists λ such that $\mathcal{M}[\phi]$ has value 1 if, and only if, $\mathcal{B}_\lambda$ has value 1.

It follows that:

$$\begin{aligned}
\exists \phi, \mathrm{val}(\mathcal{M}[\phi]) = 1 &\implies \exists \lambda, \mathrm{val}(\mathcal{B}_\lambda) = 1 \\
&\implies \mathrm{val}(\mathcal{A}) = 1 \\
&\implies \forall \lambda, \mathrm{val}(\mathcal{B}_\lambda) = 1 \\
&\implies \forall \phi, \mathrm{val}(\mathcal{M}[\phi]) = 1.
\end{aligned}$$

5.2.2 The fair coin construction

Let $\mathcal{A}$ be a simple probabilistic automaton over the alphabet A. We construct a family of probabilistic automata $\mathcal{B}_\lambda$ for $0 < \lambda < 1$ over the alphabet $B = A \cup \{\sharp\}$, whose transition function is $\phi_\lambda : A \to \mathcal{S}_{Q \times Q}(\{0, \lambda, 1-\lambda, 1\})$.

The automaton $\mathcal{B}_\lambda$ is a copy of $\mathcal{A}$ where every probabilistic transition of $\mathcal{A}$ is replaced by the gadget illustrated in Figure 26. The initial and final states are the same in $\mathcal{A}$ and in $\mathcal{B}_\lambda$.

The left hand side shows part of automaton $\mathcal{A}$: a probabilistic transition from q reading a, leading to r or s each with probability half. The right hand side shows how this behaviour is simulated by $\mathcal{B}_\lambda$: the letter a leads to an intermediate state q_a, from which we can read a new letter $\sharp$. Each time a pair of $\sharp$'s is read, the automaton B_λ goes to r with probability $\lambda \cdot (1-\lambda)$, goes to s with probability $(1-\lambda)\cdot\lambda$, and stays in q_a with probability $\lambda^2 + (1-\lambda)^2$. Reading a letter other than $\sharp$ while the automata is still in one of the new states leads to a rejecting sink state $\perp$. Thus, the probability of going to r is equal to the probability of going to s, and we can make the probability of a simulation failure as low as we want by increasing the number of $\sharp$'s between two successive letters from A.

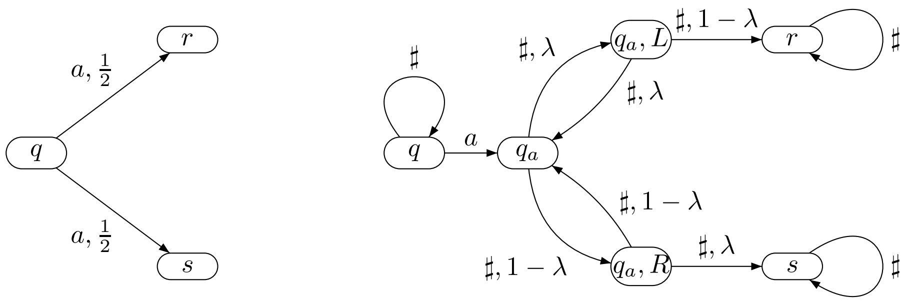


Figure 26: The fair coin gadget.

Let u be a word of A^*. We denote by $[u]^k$ the word of B^* where each letter $a \in A$ of u is replaced by $a \cdot \sharp^{2k}$. Conversely, if w is a word of B^*, we denote by $\tilde{w}$ the word obtained from w by removing all occurrences of the letter $\sharp$.

Intuitively, a run of $\mathcal{A}$ on the word u is simulated by a run of $\mathcal{B}_\lambda$ on the word $[u]^k$. Whenever there is a transition in $\mathcal{A}$, $\mathcal{B}_\lambda$ makes k attempts to simulate it through the gadget of Figure 26, and each attempt succeeds with probability $(1 - 2\lambda \cdot (1-\lambda))$, so each transition fails with probability:

$$A_{\lambda,k} = 1 - (1 - 2\lambda \cdot (1-\lambda))^k \,.$$

Lemma 30 (Correctness of the fair coin construction)

The probabilistic automaton $\mathcal{B}_\lambda$ satisfies, for all states $q, r \in Q$ and $k \in \mathbb{N}$:

1. for every letter $a \in A$, we have $P_{\mathcal{B}_\lambda}(q \xrightarrow{[a]^k} r) = A_{\lambda,k} \cdot P_{\mathcal{A}}(q \xrightarrow{a} r)$,
2. for all words $u \in A^*$, we have $P_{\mathcal{B}_\lambda}(q \xrightarrow{[u]^k} r) = A_{\lambda,k}^{|u|} \cdot P_{\mathcal{A}}(q \xrightarrow{u} r)$,

3. for all words $w \in B^*$, we have $P_{\mathcal{B}_\lambda}(q \xrightarrow{w} r) \leqslant P_{\mathcal{A}}(q \xrightarrow{\tilde{w}} r)$.

Proof. We prove the first item. Denote $\phi_{\mathcal{A}}(q,a) = \frac{1}{2} \cdot r + \frac{1}{2} \cdot s$. First, we have $P_{\mathcal{B}_\lambda}(q \xrightarrow{[a]^k} r) = P_{\mathcal{B}_\lambda}(q_a \xrightarrow{\sharp^{2k}} r))$. Reading two letters $\sharp$ from q_a is interpreted as a round, with three possible outcomes: reaching r with probability $\lambda(1-\lambda)$, reaching s with probability $\lambda(1-\lambda)$, or coming back to q_a with probability $1 - 2 \cdot \lambda(1-\lambda)$. Then

$$P_{\mathcal{B}_\lambda}(q_a \xrightarrow{\sharp^{2k}} r) = \sum_{i=1}^{k} (1 - 2 \cdot \lambda(1-\lambda))^{i-1} \cdot \lambda(1-\lambda) = \frac{A_{\lambda,k}}{2}.$$

The second item follows from the first item by induction.

The third item is by construction. ∎

Proposition 7 (Fair coin construction)

For all $0 < \lambda < 1$, we have $\mathrm{val}(\mathcal{B}_\lambda) = \mathrm{val}(\mathcal{A})$.

Proof. Consider $(u_n)_{n\in\mathbb{N}}$ a sequence of words in A^*. Denote $k_n = \max(|u_n|, n)$ and consider the sequence $(w_n)_{n\in\mathbb{N}}$ defined by $w_n = [u_n]^{k_n}$. Thanks to the second item of Lemma 30, we have

$$P_{\mathcal{B}_\lambda}(w_n) = A_{\lambda,k_n}^{|u_n|} \cdot P_{\mathcal{A}}(u_n).$$

The sequence $\left(A_{\lambda,k_n}^{|u_n|}\right)_{n\in\mathbb{N}}$ converges to 1, so $\lim_n P_{\mathcal{B}_\lambda}(w_n) = \lim_n P_{\mathcal{A}}(u_n)$. Thus $\mathrm{val}(\mathcal{A}) \leqslant \mathrm{val}(\mathcal{B}_\lambda)$.

The third item of Lemma 30 implies that $\mathrm{val}(\mathcal{B}_\lambda) \leqslant \mathrm{val}(\mathcal{A})$. ∎

5.2.3 The simulation construction

All the $\mathcal{B}_\lambda$'s induce the same numberless probabilistic automaton, that we denote by $\mathcal{B}$. Observe that there are many other instances of $\mathcal{B}$, as illustrated in Figure 26, the transitions from the states (q_a, L) and (q_a, R) to the state q_a add up to 1, which is not true for all instances of $\mathcal{B}$.

We construct a numberless probabilistic automaton $\mathcal{M}$ over an extended alphabet C whose instances simulate all the $\mathcal{B}_\lambda$'s, but only them. The idea is that the new numberless probabilistic automaton will only have *one* probabilistic transition. An instance of this probabilistic transition translates to a value for λ. Figure 27 describes a first attempt at this (notice that our convention is that a non-drawn transition means a loop, rather than a transition to a sink state).

In this automaton, that we call $\mathcal{M}'$, a single shared probabilistic transition $q_R \xrightarrow{\$} \lambda \cdot s_0 + (1-\lambda) \cdot s_1$ is used for all probabilistic transitions. In order to make all the probabilistic transitions of $\mathcal{B}$ happen in q_R, we use new letters to detect where the runs come from before the probabilistic transition, and where it should go afterwards.

For each letter b in B and state q in Q, we introduce two new letters check(b,q) and apply(b,q). We also introduce a new letter next_transition. Reading a b in $\mathcal{B}$ corresponds to reading $\widehat{b}$ in $\mathcal{M}'$, defined by:

$$\widehat{b} = \mathrm{check}(b,q_0) \cdot \$ \cdot \mathrm{apply}(b,q_0) \ \cdots\ \mathrm{check}(b,q_{n-1}) \cdot \$ \cdot \mathrm{apply}(b,q_{n-1}) \cdot \mathrm{next\_transition},$$

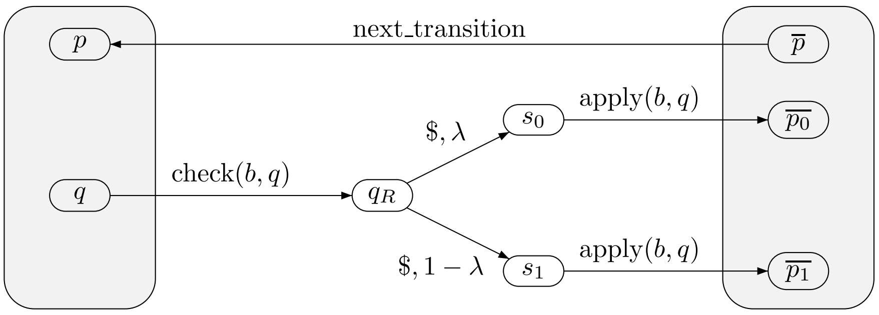


Figure 27: Naive fusion of the probabilistic transitions.

where $\{q_0, q_1, \ldots, q_{n-1}\}$ is the set of states of $\mathcal{B}$. The function $\widehat{\_}$ extends to a morphism.

There are two copies of the set of states, which are the light gray boxes on the left and on the right of Figure 27. We illustrate the action of check$(b,q) \cdot \$ \cdot$ apply(b,q), which aims at simulating the transition from q reading b:

- the letter check(b,q) leads from the left copy of q to q_R
- the letter $\sharp$ leads to $\lambda \cdot s_0 + (1-\lambda) \cdot s_1$,
- the letter apply(b,q) leads from s_0 to the λ-valued successor of (q,b) in $\mathcal{B}_\lambda$, and from s_1 to the $(1-\lambda)$-valued successor of (q,b) in $\mathcal{B}_\lambda$.

Note that reading check$(b,q) \cdot \$ \cdot$apply$(b,q)$ leaves everything else unchanged, as the actions of the letters on the other states are just self-loops. Once the transitions from every state have been simulated, the new letter next_transition sends the run back to the left part.

We get for any word $u \in B^*$, $P_{\mathcal{B}_\lambda}(u) = P_{\mathcal{B}'_\lambda}(\widehat{u})$.

The problem with the automaton $\mathcal{M}'$ is that one can "cheat", either by not testing an unwelcome state, or by changing state and letter between a check and the subsequent apply. For instance, the word check$(b,q) \cdot \$ \cdot$ apply(a,p) does not correspond to any behaviour of $\mathcal{B}$. We say that a word is *fair* if it is of the form $\widehat{u}$ for $u \in B^*$.

The idea is to simulate a word u in B^* by $(\widehat{u} \cdot \text{next\_word})^n$, for a large n, where next_word is a new letter. Each time $\widehat{u}$ is read, the run of the new automaton is split in two parts: a proportion of the run is for the simulation itself, as described above, and the rest goes to a wait state, waiting for the next word. Once the word $\widehat{u}$ is read, the new letter next_word has the following effect:

- the proportion of the run that simulated u and is accepting goes to the initial state of the fairness checker $\mathcal{D}$,
- the proportion of the run that simulated u and is not accepting goes to the non-accepting sink state $\bot$,
- the proportion of the run that ended up in the wait state goes back to the initial state q_0, for the next simulation.

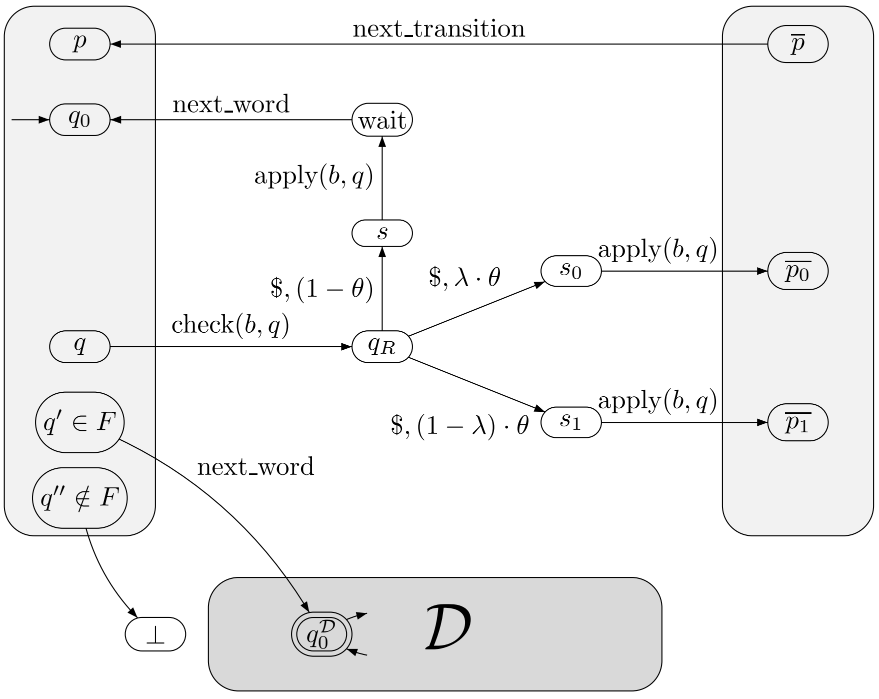


Figure 28: The numberless probabilistic automaton $\mathcal{M}$.

The fairness checker $\mathcal{D}$ is a deterministic automaton recognizing the regular language $\{\widehat{u} \cdot \text{next\_word} \mid u \in B^*\}^*$.

The resulting automaton, denoted $\mathcal{M}$ is described in Figure 28. The structure of the automaton of Figure 27 is still there, but it has been augmented with an extra layer of scrutiny: each time we use the probabilistic transition, there is now a positive probability $(1-\theta)$ to go to a new wait state. The final state of $\mathcal{D}$ is the only final state in all of $\mathcal{M}$.

Following Figure 28, we denote $\mathcal{M}[\lambda,\theta]$ an instance of $\mathcal{M}$. The alphabet of $\mathcal{M}$ is denoted C.

Consider the action of reading $(\widehat{u} \cdot \text{next\_word})^n$. The probability to be in the fairness checker increases each time a letter next_word is read, by a proportion of the acceptance probability of $\widehat{u}$. If at some point an unfair word is read, then the probability to remain in the fairness checker drops to 0, hence cheating is punished. A run can still cheat before the *first* next_word letter, but the benefits of doing so are limited: the probability that $\mathcal{M}[\lambda,\theta]$ accepts a word at that point is at most θ (except if the empty word is accepting, but that case is trivial).

A more formal proof follows.

Lemma 31 (Correctness of the simulation construction)

1. Let u be a word of B^* of length k and w a word of $(C \setminus \{\text{next\_word}\})^*$. We have

$$P_{\mathcal{M}[\lambda,\theta]}(\widehat{u} \cdot \text{next\_word} \cdot w) = \theta^k \cdot P_{\mathcal{B}_\lambda}(u) + \left(1-\theta^k\right) \cdot P_{\mathcal{M}[\lambda,\theta]}(w).$$

2. Let u be a word in B^* of length k and $\ell \geqslant 1$. We have
$$P_{\mathcal{M}[\lambda,\theta]}((\widehat{u} \cdot \text{next\_word})^\ell) = (1 - (1-\theta^k)^\ell) \cdot P_{\mathcal{B}_\lambda}(u).$$

3. Let w be a word of $(C \backslash \{\text{next\_word}\})^*$ and w' a word of C^*. We have
$$P_{\mathcal{M}[\lambda,\theta]}(w \cdot \text{next\_word} \cdot w') \leqslant \theta + (1-\theta) \cdot P_{\mathcal{M}[\lambda,\theta]}(w').$$

4. Let $w_1, \ldots, w_k$ be k words of $(C \backslash \{\text{next\_word}\})^*$ and w be the word $w_1 \cdot \text{next\_word} \cdots w_k \cdot \text{next\_word}$. Then, for any $1 \leqslant i \leqslant k$, if $w_i \notin \widehat{B^*}$, we have
$$P_{\mathcal{M}[\lambda,\theta]}(w) \leqslant P_{\mathcal{M}[\lambda,\theta]}(w_i \cdot \text{next\_word} \cdots w_k \cdot \text{next\_word}).$$

5. Let w be a word in C^* such that $P_{\mathcal{M}[\lambda,\theta]}(w) > \theta$. Then there exists a word $u \in B^*$ such that
$$P_{\mathcal{B}_\lambda}(u) \geqslant \frac{P_{\mathcal{M}[\lambda,\theta]}(w) - \theta}{1-\theta}.$$

Proof.

1. This follows from the observation that after reading $\widehat{u} \cdot \text{next\_word}$, the distribution is
$$(1-\theta^k) \cdot q_0 \;+\; \theta^k \cdot P_{\mathcal{B}_\lambda}(u) \cdot q_0^{\mathcal{D}} \;+\; \theta^k \cdot (1 - P_{\mathcal{B}_\lambda}(u)) \cdot \perp.$$

2. This follows from the first item by induction on ℓ.

3. This follows from the observation that after reading $w \cdot \text{next\_word}$, the run is in one of the following three states: q_0 with probability at most $1-\theta$, $q_0^{\mathcal{D}}$ with probability at most θ, and $\perp$.

4. After reading $w_{<i} = w_1 \cdot \text{next\_word} \cdots w_{i-1} \cdot \text{next\_word}$, the run is in one of the following three states: q_0, $q_0^{\mathcal{D}}$, and $\perp$. As $w_i \notin \widehat{B^*}$, reading w_i from $q_0^{\mathcal{D}}$ leads to $\perp$. Thus,
$$P_{\mathcal{M}[\lambda,\theta]}(w) = P_{\mathcal{M}[\lambda,\theta]}(q_0 \xrightarrow{w_{<i}} q_0) \cdot P_{\mathcal{M}[\lambda,\theta]}(w_i \cdot \text{next\_word} \cdots w_k \cdot \text{next\_word}),$$
which concludes.

5. Denote $w = w_1 \cdot \text{next\_word} \cdots w_p \cdot \text{next\_word}$ with $w_1, \ldots, w_p \in (C \backslash \{\text{next\_word}\})^*$. Thanks to the third item, $p > 1$, and thank to the fourth item, we can assume without loss of generality that $w_2, \ldots, w_p$ belong to $\widehat{B^*}$. Let $u_2, \ldots, u_p$ be the words of B^* such that $w_i = \widehat{u_i}$. Denote $w' = \widehat{u_1} \cdot \text{next\_word} \cdot \ldots \cdot \widehat{u_p} \cdot \text{next\_word}$, thanks to the third item we have $P_{\mathcal{M}[\lambda,\theta]}(w) \leqslant \theta + (1-\theta) \cdot P_{\mathcal{M}[\lambda,\theta]}(w')$. It follows that $P_{\mathcal{M}[\lambda,\theta]}(w') \geqslant \frac{P_{\mathcal{M}[\lambda,\theta]}(w) - \theta}{1-\theta}$.
Denote $w'' = \widehat{u_2} \cdot \text{next\_word} \cdot \ldots \cdot \widehat{u_p} \cdot \text{next\_word}$. Without loss of generality, we assume that $P_{\mathcal{M}[\lambda,\theta]}(w'') \leqslant P_{\mathcal{M}[\lambda,\theta]}(w')$. Denote k the length of u_1, thanks to the first item we have
$$P_{\mathcal{M}[\lambda,\theta]}(w') = \theta^k \cdot P_{\mathcal{B}_\lambda}(u_1) + \left(1-\theta^k\right) \cdot P_{\mathcal{M}[\lambda,\theta]}(w'').$$
It follows that $P_{\mathcal{B}_\lambda}(u_1) \geqslant P_{\mathcal{M}[\lambda,\theta]}(w')$, which concludes.

∎

Proposition 8 (Simulation construction)

For all $0 < \lambda < 1$ and $0 < \theta < 1$, we have $\mathrm{val}(\mathcal{B}_\lambda) = 1$ if, and only if, $\mathrm{val}(\mathcal{M}[\lambda,\theta]) = 1$.

Proof. Let $(u_n)_{n \in \mathbb{N}}$ a sequence of words in B^* such that $\lim_n P_{\mathcal{B}_\lambda}(u_n) = 1$. Denote $k_n = \max(2^{|}u_n|, n)$ and consider the sequence $(w_n)_{n\in\mathbb{N}}$ defined by $w_n = (\widehat{u_n} \cdot \mathrm{next\_word})^{k_n}$.

Thanks to the second item of Lemma 31, we have

$$P_{\mathcal{M}[\lambda,\theta]}(w_n) = (1 - (1 - \theta^{|u_n|})^{k_n}) \cdot P_{\mathcal{B}_\lambda}(u_n).$$

A closer look at the sequence $((1 - \theta^{|u_n|})^{k_n})_{n\in\mathbb{N}}$ shows that it converges to 0. It follows that $\lim_n P_{\mathcal{M}[\lambda,\theta]}(w_n) = 1$, so $\mathrm{val}(\mathcal{M}[\lambda,\theta]) = 1$.

Conversely, let $(w_n)_{n\in\mathbb{N}}$ be a sequence of words in C^* such that $\lim_n P_{\mathcal{M}[\lambda,\theta]}(w_n) = 1$. Without loss of generality we can assume that for all $n \in \mathbb{N}$, we have $P_{\mathcal{M}[\lambda,\theta]}(w_n) > 1 - \theta_w$. It follows from the fifth item of Lemma 31 that there exists a sequence $(u_n)_{n\in\mathbb{N}}$ of words in B^* such that for all $n \in \mathbb{N}$, we have $P_{\mathcal{B}_\lambda}(u_n) \geqslant \frac{P_{\mathcal{M}[\lambda,\theta]}(w_n) - \theta}{1-\theta}$. Thus $\lim_n P_{\mathcal{B}_\lambda}(u_n) = 1$, so $\mathrm{val}(\mathcal{B}_\lambda) = 1$. ∎

Proposition 6 is a direct corollary of Proposition 7 and Proposition 8.

5.3 Undecidability of the Robustness Problems

In this subsection, we consider robustness problems: rather than giving as input a probabilistic automaton, we look at a probabilistic automaton up to small modifications of its probabilistic transitions. This question appears naturally in practice: the probabilistic transitions may not be known with absolute precision, and may be subject to small perturbations.

In his original paper, Rabin [Rab63] considered the influence of small perturbations of the probabilistic transitions on the languages defined by the automaton. He showed that for the class of probabilistic automaton called actual automata, in which there is always a positive probability to go from any state to any other, the language is robust, *i.e.* is not affected by small perturbations. This result motivated our investigations of the robustness problems.

Here we prove the undecidability of three robustness problems, relying on the undecidability of the numberless problems.

Consider a probabilistic automaton $\mathcal{A}$ and $\varepsilon > 0$. We say that the transition function $\phi' : A \to \mathcal{S}_{Q\times Q}(\mathbb{Q})$ is ε-close to $\mathcal{A}$ if for all letters $a \in A$ and states $s, t \in Q$, we have:

- $|\phi(a)(s,t) - \phi'(a)(s,t)| < \varepsilon$, and
- $\phi(a)(s,t) > 0$ if, and only if, $\phi'(a)(s,t) > 0$.

Problem 9 (Robustness problems)

- **The existential robustness value 1 problem:** given a probabilistic automaton $\mathcal{A}$ and $\varepsilon > 0$, determine whether there exists ϕ' which is ε-close to $\mathcal{A}$ such that $\mathrm{val}(\mathcal{A}[\phi']) = 1$.
- **The universal robustness value 1 problem:** given a probabilistic automaton $\mathcal{A}$ and $\varepsilon > 0$, determine whether for all ϕ' which are ε-close to $\mathcal{A}$ we have $\mathrm{val}(\mathcal{A}[\phi']) = 1$.
- **The universal robustness emptiness problem:** given a probabilistic automaton $\mathcal{A}$ and $\varepsilon > 0$, determine whether there exists ϕ' which is ε-close to $\mathcal{A}$ such that $\mathrm{val}(\mathcal{A}[\phi']) \geqslant \frac{1}{2}$.

The existential robustness emptiness problem is missing from this definition; indeed, it is trivial, and reduces to a simple reachability question.

The undecidability of the numberless value 1 problems easily imply the undecidability of the three robustness problems:

Theorem 38 (Undecidability of the robustness problems)

The three robustness problems are undecidable.

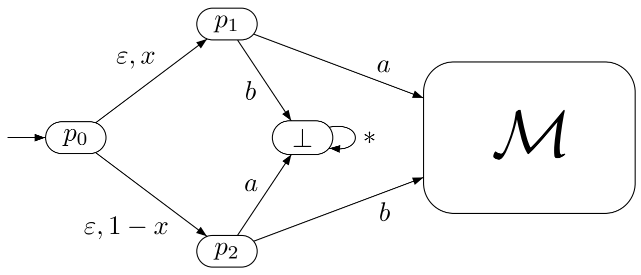


Figure 29: Reduction.

Proof. The undecidability of both robustness value 1 problems is obtained using Proposition 6, using the same reasoning as given there to prove Theorem 37.

To show that the universal robustness emptiness problem is undecidable, we construct a reduction from the universal robustness value 1 problem.

For the sake of explanation, we give a reduction from the universal numberless value 1 problem to the universal numberless emptiness problem, and later explain how this induces a similar reduction for robustness problems instead of numberless problems.

Let $\mathcal{M}$ be a numberless probabilistic automaton. We construct $\mathcal{M}'$ a numberless probabilistic automaton such that:

$$\forall \phi_{\mathcal{M}},\ \mathrm{val}(\mathcal{M}[\phi_{\mathcal{M}}]) = 1 \iff \forall \phi_{\mathcal{M}'},\ \mathrm{val}(\mathcal{M}'[\phi_{\mathcal{M}'}]) \geqslant \frac{1}{2}.$$

The reduction is illustrated in Figure 29. Let a, b two new letters (not in A). The automaton $\mathcal{M}'$ contains a copy of $\mathcal{M}$, and four new states: p_0, p_1, p_2 and a sink state $\perp$. The state p_0 is initial, and leads non-deterministically to p_1 or p_2 with an ε-transition. The state p_1 leads to the initial state of $\mathcal{M}$ with a and to $\perp$ with b, and vice-versa for p_2. Observe that a transition function $\phi'_{\mathcal{M}}$ for $\mathcal{M}'$ is given by a transition function $\phi_{\mathcal{M}}$ for $\mathcal{M}$ and a value x for the ε-transition from p_0. Also, remark that for $w \in A^*$ and $\phi'_{\mathcal{M}}$ extending $\phi_{\mathcal{M}}$, we have $P_{\mathcal{M}'[\phi_{\mathcal{M}'}]}(a \cdot w) = x \cdot P_{\mathcal{M}[\phi_{\mathcal{M}}]}(w)$ and $P_{\mathcal{M}'[\phi_{\mathcal{M}'}]}(b \cdot w) = (1-x) \cdot P_{\mathcal{M}[\phi_{\mathcal{M}}]}(w)$.

We prove the equivalence:

- Assume that for all $\phi_{\mathcal{M}}$, we have $\text{val}(\mathcal{M}[\phi_{\mathcal{M}}]) = 1$. Fix $\phi_{\mathcal{M}}$, and consider x, inducing $\phi_{\mathcal{M}'}$. Let $\varepsilon > 0$, there exists a word w such that $P_{\mathcal{M}[\phi_{\mathcal{M}}]}(w) \geqslant 1 - \varepsilon$. If $x \geqslant \frac{1}{2}$, then $P_{\mathcal{M}'[\phi_{\mathcal{M}'}]}(a \cdot w) \geqslant \frac{1-\varepsilon}{2}$, and if $x < \frac{1}{2}$ then $P_{\mathcal{M}'[\phi_{\mathcal{M}'}]}(b \cdot w) \geqslant \frac{1-\varepsilon}{2}$. In both cases there exists a word accepted with probability at least $\frac{1-\varepsilon}{2}$, so $\text{val}(\mathcal{M}'[\phi_{\mathcal{M}'}]) \geqslant \frac{1}{2}$.

- Assume that for all $\phi_{\mathcal{M}'}$, we have $\text{val}(\mathcal{M}'[\phi'_{\mathcal{M}}]) \geqslant \frac{1}{2}$. Let $\phi_{\mathcal{M}}$, and denote by $\phi'_{\mathcal{M}}$ its extension with $x = \frac{1}{2}$. Let $\varepsilon > 0$, there exists a word w such that $P_{\mathcal{M}'[\phi_{\mathcal{M}'}]}(w) \geqslant \frac{1}{2} - \varepsilon$. Consider w' the word obtained from w by removing its first letter, then $P_{\mathcal{M}[\phi_{\mathcal{M}}]}(w') \geqslant 1 - 2 \cdot \varepsilon$. It follows that $\text{val}(\mathcal{M}[\phi_{\mathcal{M}}]) = 1$.

This proves the correctness of the reduction. To obtain a reduction for robustness problems, we follow the same construction and set x to be $\frac{1}{2}$. The same proof as above concludes. ∎

6 Conclusions

At the end of the day, how did we answer the initial question: "to what extent is the value 1 problem undecidable"? This section gathers the different elements of answers in Subsection 6.1, supporting the claim that the Markov Monoid algorithm is optimal.

Subsection 6.2 opens perspectives for future research.

6.1 Optimality

The first observation is that probabilistic automata may exhibit complicated convergence phenomena with different convergence speeds, as witnessed by Example 3 from Subsection 1.3.

We construct in Section 3 an algebraic structure, extending the classical notion of transition monoid of a non-deterministic automaton, called the Markov Monoid. In addition to concatenation, the Markov Monoid additionally features a stabilisation operator, which accounts for some limit behaviours. This gives rise to an algorithm, called the Markov Monoid algorithm, which aims at partially solving the value 1 problem.

Using the prostochastic theory developed in Section 2, we give a characterisation of the Markov Monoid algorithm. Roughly speaking, it says that the Markov Monoid algorithm captures exactly all polynomial behaviours, which are described by polynomial prostochastic words.

We show in Section 4 that the Markov Monoid algorithm subsumes all previous known algorithms to solve the value 1. Indeed, we prove that it is correct for the subclass of leaktight automata, and that the class of leaktight automata strictly contains all subclasses for which the value 1 problem has been shown to be decidable.

At this point, the Markov Monoid algorithm is the best algorithm *so far*. But can we go further? Section 5 gives two arguments supporting the claim that the Markov Monoid algorithm is *in some sense* optimal.

We first discuss the undecidability of the numberless value 1 problems, proved in Subsection 5.2. Since the Markov Monoid does not take into account the numerical values of the probabilistic transitions, it is natural to ask whether loosely specifying these numerical values makes the problem easier. To formalize this, we consider numberless probabilistic automata, which can be instantiated as probabilistic automata by specifying numerical values for the probabilistic transitions. We give a construction that takes as input a probabilistic automaton $\mathcal{A}$, and outputs a numberless probabilistic automaton $\mathcal{C}$ such that the following are equivalent:

- $\mathcal{A}$ has value 1,
- there exists an instance of $\mathcal{C}$ that has value 1,
- all instances of $\mathcal{C}$ have value 1.

Consequently, the numerical values attached to $\mathcal{C}$ do not matter, and whether $\mathcal{C}$ has value 1 or not does not depend on them. Besides the undecidability result that follows, this also shows that abstracting away the numerical values, as the Markov Monoid does, does not make the problem easier.

We now discuss the undecidability of the two-tier value 1 problem, proved in Subsection 5.1. A two-tier sequence of finite words is a sequence of the form $((u \cdot v^n)^{2^n})_{n \in \mathbb{N}}$, for u, v two finite words. Intuitively, they combine two different behaviours. The first is linear, it consists in repeating n times the word v. The second is exponential, it consists in repeating 2^n times the word $u \cdot v^n$. Our interpretation is that the undecidability of the value 1 problem arises when polynomial and exponential behaviours are combined. Since the Markov Monoid captures exactly all polynomial behaviours, it is in this sense optimal.

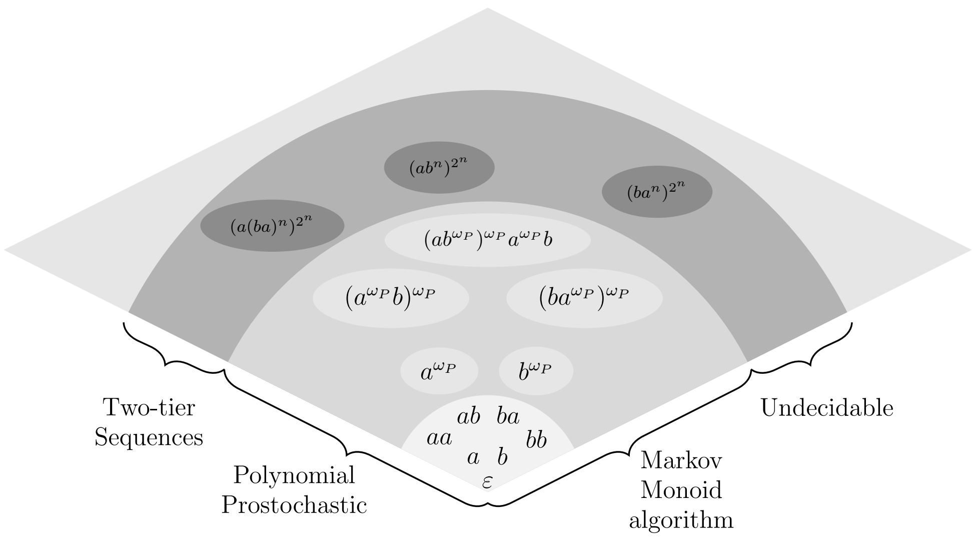


Figure 30: Optimality of the Markov Monoid algorithm.

6.2 Perspectives

This chapter focuses on

$$\text{the } \underbrace{\text{value 1}}_{(1)} \text{ problem for } \underbrace{\text{probabilistic automata}}_{(2)} \text{ over } \underbrace{\text{finite words}}_{(3)}.$$

There are at least three natural follow-up questions:

(1) What about other values, such that $\frac{1}{2}$, *i.e.* the emptiness problem?

(2) What about generalisations of probabilistic automata, such as partially observable Markov decision processes?

(3) What about infinite words, or trees?

The other natural and burning question is:

(4) What is the practical value of the Markov Monoid algorithm?

We discuss these four points further.
The emptiness problem. There are both positive and negative results:

- the emptiness problem is undecidable for acyclic probabilistic automata (only self-loops are allowed), which is a quite severe structural restriction,
- different subclasses of probabilistic automata have been defined for which this problem has been shown decidable. For instance, the class of one-level hierarchical automata have been introduced very recently in [CSVB15], and shown to have a decidable emptiness problem while recognizing non-regular languages.

The general question we ask is:

"To what extent is the emptiness problem undecidable?"

In particular, can we find an algorithm that is *often* correct? Can we quantify *how often*? Can we argue that it is *optimal*? We hope that the prostochastic theory may help in answering these questions.

Several variants of the emptiness problem have been proposed. For instance, adding approximation may lead to decidable problems; a promising step was done in this direction recently in [AAGT15].

Partially observable Markov decision processes. Extending subclasses of probabilistic automata to partially observable Markov decision processes to decide the value 1 problem has already started: this has been done for the class of $\sharp$-acyclic automata [GO14]. No extensions of the class of leaktight automata have been defined so far.

Beyond finite words. We investigated the setting of infinite words in [FGKO15], and proved a generic result allowing to reduce the value 1 problem from infinite words to finite words. Note that this is different from the equivalence between the value 1 problem for finite words and the emptiness problem for probabilistic Büchi automata with positive semantics, as discussed in Subsection 1.3.

Defining probabilistic automata over infinite trees is not obvious. An interesting and well behaved definition was proved by Carayol, Haddad in Serre in [CHS11; CHS14], called *qualitative automata*. We extended the definition to alternating qualitative automata in [FPS13]. A lot of open questions remain about this class of languages over infinite trees.

Practical value of the Markov Monoid algorithm. Together with Denis Kuperberg, we implemented the Markov Monoid algorithm in the tool ACME, providing benchmarks, examples and inspiration to go forward. The experiments on ACME have been reported in [FK14], co-authored with Denis Kuperberg. We pushed this further, and more specifically faster; together with Hugo Gimbert, Edon Kelmendi and Denis Kuperberg, we created the tool ACME^{++} with the same purpose as ACME, but faster by several orders of magnitude.

Conclusion

What is the relation between the two chapters, *i.e.* between finite-memory determinacy for boundedness games and the value 1 problem for probabilistic automata?

The short answer is "none". We give a more constructive answer, arguing that a contribution of this document is to show that some techniques used in one context can be used in the other.

A first very rough approximation is that both chapters are about quantitative extensions of automata theory, the first automata with counters and the second probabilistic automata. A second closer inspection reveals that the two chapters are technically unrelated; the definitions, notions, theorems and results are completely independent. However, one can see the influence of the first over the second at a higher level. Indeed, we developed here tools for probabilistic automata and expand on ideas that have successfully been applied to automata with (and also without) counters.

The first example is the algebraic notion of *stabilisation monoids*, which has been introduced by Colcombet in the theory of regular cost functions [Col09], together with its analysis using Green's relations [Kir05; Col11].

The introduction of the Markov Monoid algorithm was motivated by defining a stabilisation monoid for probabilistic automata.

The second example is the algebraic and combinatorial notion of *factorisation forests of bounded height*, which has been introduced by Simon in the study of distance automata, and later developed by Colcombet and Toruńczyk for similar purposes [Sim94; Col10; Tor11].

The class of leaktight automata, and more specifically the proof that the Markov Monoid is complete for leaktight automata, is based on the notion of factorisation forests.

The third example is the topological notion of *profinite words* for regular languages and regular cost functions [Pin09; Tor11].

The prostochastic theory for probabilistic automata is an extension of the profinite theory for classical automata. The idea of using a profinite theory for probabilistic automata was inspired (although technically unrelated) by the treatment of $\mathrm{MSO} + \mathbb{U}$ using profinite techniques by Toruńczyk.

Bibliography

Personal references cited in this document

[CF13] Krishnendu Chatterjee and Nathanaël Fijalkow. "Infinite-state Games with Finitary Conditions". In: *CSL*. 2013, pp. 181–196. DOI: `10.4230/LIPIcs.CSL.2013.181`. URL: `http://dx.doi.org/10.4230/LIPIcs.CSL.2013.181` (cited on pp. 20, 22, 27, 29, 30, 32, 34, 35, 45).

[CFH14] Thomas Colcombet, Nathanaël Fijalkow, and Florian Horn. "Playing Safe". In: *FSTTCS*. 2014, pp. 379–390. DOI: `10.4230/LIPIcs.FSTTCS.2014.379`. URL: `http://dx.doi.org/10.4230/LIPIcs.FSTTCS.2014.379` (cited on pp. 20, 32, 35, 42).

[Fij15] Nathanaël Fijalkow. "Profinite Techniques for Probabilistic Automata, with Applications to the Markov Monoid Algorithm". In: *CoRR* abs/1501.02997 (2015). URL: `http://arxiv.org/abs/1501.02997` (cited on pp. 80, 89, 100).

[FGHO14] Nathanaël Fijalkow, Hugo Gimbert, Florian Horn, and Youssouf Oualhadj. "Two Recursively Inseparable Problems for Probabilistic Automata". In: *MFCS*. 2014, pp. 267–278. DOI: `10.1007/978-3-662-44522-8_23`. URL: `http://dx.doi.org/10.1007/978-3-662-44522-8_23` (cited on pp. 80, 123, 125, 126).

[FGKO15] Nathanaël Fijalkow, Hugo Gimbert, Edon Kelmendi, and Youssouf Oualhadj. "Deciding the value 1 Problem for Probabilistic Leaktight Automata". In: *Logical Methods in Computer Science* 11.1 (2015) (cited on pp. 80, 107, 121, 122, 138).

[FGO12] Nathanaël Fijalkow, Hugo Gimbert, and Youssouf Oualhadj. "Deciding the Value 1 Problem for Probabilistic Leaktight Automata". In: *LICS*. 2012, pp. 295–304. DOI: `10.1109/LICS.2012.40`. URL: `http://dx.doi.org/10.1109/LICS.2012.40` (cited on pp. 80, 100, 107, 121).

[FH13] Nathanaël Fijalkow and Florian Horn. "Les Jeux d'accessibilité généralisée". In: *Technique et Science Informatiques* 32.9-10 (2013), pp. 931–949. DOI: `10.3166/tsi.32.931-949`. URL: `http://dx.doi.org/10.3166/tsi.32.931-949` (cited on p. 42).

[FHKS15] Nathanaël Fijalkow, Florian Horn, Denis Kuperberg, and Michał Skrzypczak. “Trading Bounds for Memory in Games with Counters”. In: *ICALP (2)*. 2015 (cited on pp. 21, 63, 69).

[FK14] Nathanaël Fijalkow and Denis Kuperberg. “ACME: Automata with Counters, Monoids and Equivalence”. In: *ATVA*. 2014, pp. 163–167. DOI: `10.1007/978-3-319-11936-6_12`. URL: `http://dx.doi.org/10.1007/978-3-319-11936-6_12` (cited on pp. 102, 138).

[FPS13] Nathanaël Fijalkow, Sophie Pinchinat, and Olivier Serre. “Emptiness Of Alternating Tree Automata Using Games With Imperfect Information”. In: *FSTTCS*. 2013, pp. 299–311. DOI: `10.4230/LIPIcs.FSTTCS.2013.299`. URL: `http://dx.doi.org/10.4230/LIPIcs.FSTTCS.2013.299` (cited on pp. 20, 53, 60, 138).

[FS15] Nathanaël Fijalkow and Michał Skrzypczak. “Undecidability of the Regularity Problem for Probabilistic Automata”. 2015 (cited on pp. 80, 87).

[FZ12] Nathanaël Fijalkow and Martin Zimmermann. “Cost-Parity and Cost-Streett Games”. In: *FSTTCS*. 2012, pp. 124–135. DOI: `10.4230/LIPIcs.FSTTCS.2012.124`. URL: `http://dx.doi.org/10.4230/LIPIcs.FSTTCS.2012.124` (cited on pp. 20, 32, 34, 78).

[FZ14] Nathanaël Fijalkow and Martin Zimmermann. “Cost-Parity and Cost-Streett Games”. In: *Logical Methods in Computer Science* 10.2 (2014). DOI: `10.2168/LMCS-10(2:14)2014`. URL: `http://dx.doi.org/10.2168/LMCS-10(2:14)2014` (cited on pp. 20, 32, 34, 35, 78).

Personal references not cited in this document

[CF11a] Krishnendu Chatterjee and Nathanaël Fijalkow. “A Reduction from Parity Games to Simple Stochastic Games”. In: *GandALF*. 2011, pp. 74–86. DOI: `10.4204/EPTCS.54.6`. URL: `http://dx.doi.org/10.4204/EPTCS.54.6`.

[CF11b] Krishnendu Chatterjee and Nathanaël Fijalkow. “Finitary Languages”. In: *LATA*. 2011, pp. 216–226. DOI: `10.1007/978-3-642-21254-3_16`. URL: `http://dx.doi.org/10.1007/978-3-642-21254-3_16`.

[FP14] Nathanaël Fijalkow and Charles Paperman. “Monadic Second-Order Logic with Arbitrary Monadic Predicates”. In: *MFCS*. 2014, pp. 279–290. DOI: `10.1007/978-3-662-44522-8_24`. URL: `http://dx.doi.org/10.1007/978-3-662-44522-8_24`.

Other references cited in this document

[AAGT15] Manindra Agrawal, S. Akshay, Blaise Genest, and P. S. Thiagarajan. “Approximate Verification of the Symbolic Dynamics of Markov Chains”. In: *Journal of the ACM* 62.1 (2015), p. 2. DOI: `10.1145/2629417`. URL: `http://doi.acm.org/10.1145/2629417` (cited on p. 138).

[AH00] Luca de Alfaro and Thomas A. Henzinger. “Concurrent Omega-Regular Games”. In: *LICS*. 2000, pp. 141–154. DOI: `10.1109/LICS.2000.855763`. URL: `http://dx.doi.org/10.1109/LICS.2000.855763` (cited on p. 18).

[Alm05] Jorge Almeida. “Profinite Semigroups and Applications”. In: *Structural Theory of Automata, Semigroups, and Universal Algebra* 207 (2005), pp. 1–45 (cited on p. 91).

[AELP01] Rajeev Alur, Kousha Etessami, Salvatore La Torre, and Doron Peled. “Parametric Temporal Logic for Model Measuring”. In: *ACM Transactions on Computational Logics* 2.3 (2001), pp. 388–407. DOI: `10.1145/377978.377990`. URL: `http://doi.acm.org/10.1145/377978.377990` (cited on p. 14).

[AH98] Rajeev Alur and Thomas A. Henzinger. “Finitary Fairness”. In: *ACM Transactions on Programming Languages and Systems* 20.6 (1998), pp. 1171–1194 (cited on pp. 14, 34).

[BBG08] Christel Baier, Nathalie Bertrand, and Marcus Größer. “On Decision Problems for Probabilistic Büchi Automata”. In: *FoSSaCS*. 2008, pp. 287–301 (cited on p. 84).

[BBG09] Christel Baier, Nathalie Bertrand, and Marcus Größer. “Probabilistic Acceptors for Languages over Infinite Words”. In: *SOFSEM*. 2009, pp. 19–33 (cited on p. 84).

[BBG12] Christel Baier, Nathalie Bertrand, and Marcus Größer. “Probabilistic ω-automata”. In: *Journal of the ACM* 59.1 (2012), p. 1 (cited on pp. 18, 84, 85).

[BG05] Christel Baier and Marcus Größer. “Recognizing omega-regular Languages with Probabilistic Automata”. In: *LICS*. 2005, pp. 137–146 (cited on p. 84).

[Ber74a] Alberto Bertoni. “Mathematical Methods of the Theory of Stochastic Automata”. In: *MFCS*. 1974, pp. 9–22. DOI: `10.1007/3-540-07162-8_662`. URL: `http://dx.doi.org/10.1007/3-540-07162-8_662` (cited on pp. 17, 87).

[Ber74b] Alberto Bertoni. “The Solution of Problems Relative to Probabilistic Automata in the Frame of the Formal Languages Theory”. In: *GI Jahrestagung*. 1974, pp. 107–112 (cited on p. 17).

[BMT77] Alberto Bertoni, Giancarlo Mauri, and Mauro Torelli. “Some Recursive Unsolvable Problems Relating to Isolated Cutpoints in Probabilistic Automata”. In: *ICALP*. 1977, pp. 87–94 (cited on p. 17).

[BBG14] Nathalie Bertrand, Thomas Brihaye, and Blaise Genest. “Deciding the Value 1 Problem for Reachability in 1-Clock Decision Stochastic Timed Automata”. In: *QEST*. 2014, pp. 313–328. DOI: `10.1007/978-3-319-10696-0_25`. URL: `http://dx.doi.org/10.1007/978-3-319-10696-0_25` (cited on p. 18).

[BCKPV14] Achim Blumensath, Thomas Colcombet, Denis Kuperberg, Paweł Parys, and Michael Vanden Boom. “Two-Way Cost Automata and Cost Logics over Infinite Trees”. In: *CSL-LICS*. 2014, p. 16. DOI: `10.1145/2603088.2603104`. URL: `http://doi.acm.org/10.1145/2603088.2603104` (cited on pp. 15, 31, 53).

[BOW14] Achim Blumensath, Martin Otto, and Mark Weyer. “Decidability Results for the Boundedness Problem”. In: *Logical Methods in Computer Science* 10.3 (2014). DOI: `10.2168/LMCS-10(3:2)2014`. URL: `http://dx.doi.org/10.2168/LMCS-10(3:2)2014` (cited on p. 12).

[Boj04] Mikołaj Bojańczyk. “A Bounding Quantifier”. In: *CSL*. 2004, pp. 41–55 (cited on pp. 12, 26, 27).

[Boj11] Mikołaj Bojańczyk. “Weak MSO with the Unbounding Quantifier”. In: *Theory of Computing Systems* 48.3 (2011), pp. 554–576. DOI: `10.1007/s00224-010-9279-2`. URL: `http://dx.doi.org/10.1007/s00224-010-9279-2` (cited on p. 12).

[Boj14] Mikołaj Bojańczyk. “Weak MSO+U with Path Quantifiers over Infinite Trees”. In: *ICALP*. 2014, pp. 38–49. DOI: `10.1007/978-3-662-43951-7_4`. URL: `http://dx.doi.org/10.1007/978-3-662-43951-7_4` (cited on p. 12).

[BC06] Mikołaj Bojańczyk and Thomas Colcombet. “Bounds in ω-Regularity”. In: *LICS*. 2006, pp. 285–296 (cited on p. 12).

[BIS13] Mikołaj Bojańczyk, Tomasz Idziaszek, and Michał Skrzypczak. “Regular Languages of Thin Trees”. In: *STACS*. 2013, pp. 562–573. DOI: `10.4230/LIPIcs.STACS.2013.562`. URL: `http://dx.doi.org/10.4230/LIPIcs.STACS.2013.562` (cited on p. 51).

[BPT15] Mikołaj Bojańczyk, Paweł Parys, and Szymon Toruńczyk. “The MSO+U theory of (N, <) is Undecidable”. In: *CoRR* abs/1502.04578 (2015). URL: `http://arxiv.org/abs/1502.04578` (cited on pp. 12, 27).

[BT12] Mikołaj Bojańczyk and Szymon Toruńczyk. “Weak MSO+U over Infinite Trees”. In: *STACS*. 2012, pp. 648–660 (cited on p. 12).

[Büc60] J. Richard Büchi. “Weak second-order arithmetic and finite automata”. In: *Mathematical Logic Quarterly* 6 (1960), pp. 66–92 (cited on p. 10).

[BE58] J. Richard Büchi and Calvin C. Elgot. “Decision problems of weak second-order arithmetics and finite automata”. In: *Notices of the AMS* 5 (1958) (cited on p. 10).

[BL69] J. Richard Büchi and Lawrence H. Landweber. “Definability in the Monadic Second-Order Theory of Successor”. In: *Journal of Symbolic Logic* 34.2 (1969), pp. 166–170 (cited on p. 10).

[CHS11] Arnaud Carayol, Axel Haddad, and Olivier Serre. “Qualitative Tree Languages”. In: *LICS*. 2011, pp. 13–22. DOI: `10.1109/LICS.2011.28`. URL: `http://dx.doi.org/10.1109/LICS.2011.28` (cited on p. 138).

[CHS14] Arnaud Carayol, Axel Haddad, and Olivier Serre. “Randomization in Automata on Infinite Trees”. In: *ACM Transactions on Computational Logic* 15.3 (2014), p. 24. DOI: `10.1145/2629336`. URL: `http://doi.acm.org/10.1145/2629336` (cited on p. 138).

[CSV11] Rohit Chadha, A. Prasad Sistla, and Mahesh Viswanathan. “Power of Randomization in Automata on Infinite Strings”. In: *Logical Methods in Computer Science* 7.3 (2011) (cited on p. 122).

[CSV13] Rohit Chadha, A. Prasad Sistla, and Mahesh Viswanathan. “Probabilistic Automata with Isolated Cut-Points”. In: *MFCS*. 2013, pp. 254–265 (cited on pp. 18, 85).

[CSVB15] Rohit Chadha, A. Prasad Sistla, Mahesh Viswanathan, and Yue Ben. “Decidable and Expressive classes of Probabilistic Automata”. In: *FOSSACS*. 2015, pp. 254–265 (cited on p. 138).

[CD12] Krishnendu Chatterjee and Laurent Doyen. “Energy Parity Games”. In: *Theoretical Computer Science* 458 (2012), pp. 49–60. DOI: `10.1016/j.tcs.2012.07.038`. URL: `http://dx.doi.org/10.1016/j.tcs.2012.07.038` (cited on p. 14).

[CDGO14] Krishnendu Chatterjee, Laurent Doyen, Hugo Gimbert, and Youssouf Oualhadj. “Perfect-Information Stochastic Mean-Payoff Parity Games”. In: *FOSSACS*. 2014, pp. 210–225. DOI: `10.1007/978-3-642-54830-7_14`. URL: `http://dx.doi.org/10.1007/978-3-642-54830-7_14` (cited on p. 14).

[CHH09] Krishnendu Chatterjee, Thomas A. Henzinger, and Florian Horn. “Finitary Winning in omega-regular Games”. In: *ACM Transactions on Computational Logics* 11.1 (2009) (cited on pp. 14, 20, 34, 35).

[CT12] Krishnendu Chatterjee and Mathieu Tracol. “Decidable Problems for Probabilistic Automata on Infinite Words”. In: *LICS*. 2012, pp. 185–194 (cited on p. 121).

[Chu57] Alonzo Church. “Applications of recursive arithmetic to the problem of circuit synthesis”. In: 1 (1957), pp. 3–50 (cited on p. 10).

[Col10] Thomas Colcombet. “Factorization Forests for Infinite Words and Applications to Countable Scattered Linear Orderings”. In: *Theoretical Computer Science* 411.4-5 (2010), pp. 751–764. DOI: `10.1016/j.tcs.2009.10.013`. URL: `http://dx.doi.org/10.1016/j.tcs.2009.10.013` (cited on p. 139).

[Col13a] Thomas Colcombet. “Fonctions Régulières de Coût”. Habilitation Thesis. 2013 (cited on pp. 15, 50, 53).

[Col11] Thomas Colcombet. “Green’s Relations and Their Use in Automata Theory”. In: *LATA*. 2011, pp. 1–21 (cited on pp. 118, 139).

[Col13b] Thomas Colcombet. “Regular Cost Functions, Part I: Logic and Algebra over Words”. In: *Logical Methods in Computer Science* 9.3 (2013). DOI: `10.2168/LMCS-9(3:3)2013`. URL: `http://dx.doi.org/10.2168/LMCS-9(3:3)2013` (cited on pp. 12, 20, 49, 111).

[Col09] Thomas Colcombet. “The Theory of Stabilisation Monoids and Regular Cost Functions”. In: *ICALP (2)*. 2009, pp. 139–150 (cited on pp. 12, 20, 27, 49, 111, 112, 139).

[CKLV13] Thomas Colcombet, Denis Kuperberg, Christof Löding, and Michael Vanden Boom. “Deciding the weak definability of Büchi definable tree languages”. In: *CSL*. 2013, pp. 215–230. DOI: 10.4230/LIPIcs.CSL.2013.215. URL: http://dx.doi.org/10.4230/LIPIcs.CSL.2013.215 (cited on p. 14).

[CKL10] Thomas Colcombet, Denis Kuperberg, and Sylvain Lombardy. “Regular Temporal Cost Functions”. In: *ICALP (2)*. 2010, pp. 563–574 (cited on p. 54).

[CL10] Thomas Colcombet and Christof Löding. “Regular Cost Functions over Finite Trees”. In: *LICS*. 2010, pp. 70–79 (cited on p. 12).

[CL08a] Thomas Colcombet and Christof Löding. “The Nesting-Depth of Disjunctive μ-Calculus for Tree Languages and the Limitedness Problem”. In: *CSL*. 2008, pp. 416–430. DOI: 10.1007/978-3-540-87531-4_30. URL: http://dx.doi.org/10.1007/978-3-540-87531-4_30 (cited on pp. 12, 53).

[CL08b] Thomas Colcombet and Christof Löding. “The Non-deterministic Mostowski Hierarchy and Distance-Parity Automata”. In: *ICALP (2)*. 2008, pp. 398–409 (cited on pp. 12, 14, 50).

[Con92] Anne Condon. “The Complexity of Stochastic Games”. In: *Information and Computation* 96.2 (1992), pp. 203–224. DOI: 10.1016/0890-5401(92)90048-K. URL: http://dx.doi.org/10.1016/0890-5401(92)90048-K (cited on pp. 14, 60).

[CL89] Anne Condon and Richard J. Lipton. “On the Complexity of Space Bounded Interactive Proofs (Extended Abstract)”. In: *FOCS*. 1989, pp. 462–467. DOI: 10.1109/SFCS.1989.63519. URL: http://dx.doi.org/10.1109/SFCS.1989.63519 (cited on pp. 17, 86).

[DJW97] Stefan Dziembowski, Marcin Jurdziński, and Igor Walukiewicz. “How Much Memory is Needed to Win Infinite Games?” In: *LICS*. 1997, pp. 99–110. DOI: 10.1109/LICS.1997.614939. URL: http://dx.doi.org/10.1109/LICS.1997.614939 (cited on pp. 13, 36, 37, 78).

[Elg61] Calvin C. Elgot. “Decision problems of finite automata design and related arithmetics”. In: *Transactions of the AMS* 98 (1961), pp. 21–51 (cited on p. 10).

[EJ88] E. Allen Emerson and Charanjit S. Jutla. “The Complexity of Tree Automata and Logics of Programs (Extended Abstract)”. In: *FOCS*. 1988, pp. 328–337 (cited on pp. 11, 13, 33).

[EJ91] E. Allen Emerson and Charanjit S. Jutla. “Tree Automata, Mu-Calculus and Determinacy (Extended Abstract)”. In: *FOCS*. 1991, pp. 368–377 (cited on pp. 11, 13).

[FMS13] Alessandro Facchini, Filip Murlak, and Michal Skrzypczak. “Rabin-Mostowski Index Problem: A Step beyond Deterministic Automata”. In: *LICS*. 2013, pp. 499–508. DOI: 10.1109/LICS.2013.56. URL: http://dx.doi.org/10.1109/LICS.2013.56 (cited on p. 14).

[Gim07] Hugo Gimbert. “Jeux Positionnels”. PhD thesis. Université Paris 7, 2007 (cited on pp. 13, 33).

[GO14] Hugo Gimbert and Youssouf Oualhadj. "Deciding the Value 1 Problem for sharp-acyclic Partially Observable Markov Decision Processes". In: *SOFSEM*. 2014, pp. 281–292. DOI: `10.1007/978-3-319-04298-5_25`. URL: `http://dx.doi.org/10.1007/978-3-319-04298-5_25` (cited on p. 138).

[GO10] Hugo Gimbert and Youssouf Oualhadj. "Probabilistic Automata on Finite Words: Decidable and Undecidable Problems". In: *ICALP (2)*. 2010, pp. 527–538 (cited on pp. 18, 80, 85–87, 106, 121, 123).

[GTW02] Erich Grädel, Wolfgang Thomas, and Thomas Wilke, eds. *Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001]*. Vol. 2500. Lecture Notes in Computer Science. Springer, 2002. ISBN: 3-540-00388-6 (cited on pp. 22, 26).

[GH82] Yuri Gurevich and Leo Harrington. "Trees, Automata, and Games". In: *STOC*. 1982, pp. 60–65 (cited on pp. 11, 13).

[Has90] Kosaburo Hashiguchi. "Improved Limitedness Theorems on Finite Automata with Distance Functions". In: *Theoretical Computer Science* 72.1 (1990), pp. 27–38 (cited on p. 12).

[KMOWW11] Stefan Kiefer, Andrzej S. Murawski, Joël Ouaknine, Björn Wachter, and James Worrell. "Language Equivalence for Probabilistic Automata". In: *CAV*. 2011, pp. 526–540. DOI: `10.1007/978-3-642-22110-1_42`. URL: `http://dx.doi.org/10.1007/978-3-642-22110-1_42` (cited on p. 16).

[KMOWW13] Stefan Kiefer, Andrzej S. Murawski, Joël Ouaknine, Björn Wachter, and James Worrell. "On the Complexity of Equivalence and Minimisation for Q-weighted Automata". In: *Logical Methods in Computer Science* 9.1 (2013). DOI: `10.2168/LMCS-9(1:8)2013`. URL: `http://dx.doi.org/10.2168/LMCS-9(1:8)2013` (cited on p. 16).

[KMOWW12] Stefan Kiefer, Andrzej S. Murawski, Joël Ouaknine, Björn Wachter, and James Worrell. "On the Complexity of the Equivalence Problem for Probabilistic Automata". In: *FOSSACS*. 2012. DOI: `10.1007/978-3-642-28729-9_31`. URL: `http://dx.doi.org/10.1007/978-3-642-28729-9_31` (cited on p. 16).

[Kir06] Daniel Kirsten. "A Burnside Approach to the Finite Substitution Problem". In: *Theory of Computing Systems* 39.1 (2006), pp. 15–50. DOI: `10.1007/s00224-005-1255-x`. URL: `http://dx.doi.org/10.1007/s00224-005-1255-x` (cited on p. 12).

[Kir05] Daniel Kirsten. "Distance Desert Automata and the Star-Height Problem". In: *ITA* 39.3 (2005), pp. 455–509 (cited on pp. 12, 111, 116, 117, 120, 121, 139).

[Kle56] Stephen Cole Kleene. "Representation of Events in Nerve Nets and Finite Automata". In: *Annals of Mathematical Studies* 34 (1956), pp. 3–41 (cited on p. 10).

[Kop06] Eryk Kopczyński. "Half-Positional Determinacy of Infinite Games". In: *ICALP (2)*. 2006, pp. 336–347 (cited on pp. 37, 45, 78).

[Kop09] Eryk Kopczyński. "Half-Positional Determinacy of Infinite Games". PhD thesis. University of Warsaw, 2009 (cited on pp. 13, 33, 37, 45, 78).

[Kuč11] A. Kučera. *Turn-Based Stochastic Games.* Lectures in Game Theory for Computer Scientists. Cambridge University Press, 2011 (cited on p. 60).

[KPV09] Orna Kupferman, Nir Piterman, and Moshe Y. Vardi. “From Liveness to Promptness”. In: *Formal Methods in System Design* 34.2 (2009), pp. 83–103. DOI: 10.1007/s10703-009-0067-z. URL: http://dx.doi.org/10.1007/s10703-009-0067-z (cited on p. 14).

[KV00] Orna Kupferman and Moshe Y. Vardi. “An Automata-Theoretic Approach to Reasoning about Infinite-State Systems”. In: *CAV.* 2000, pp. 36–52. DOI: 10.1007/10722167_7. URL: http://dx.doi.org/10.1007/10722167_7 (cited on p. 31).

[KV01] Orna Kupferman and Moshe Y. Vardi. “Weak Alternating Automata are not that Weak”. In: *ACM Transactions on Computational Logic* 2.3 (2001), pp. 408–429. DOI: 10.1145/377978.377993. URL: http://doi.acm.org/10.1145/377978.377993 (cited on p. 72).

[Leu91] Hing Leung. “Limitedness Theorem on Finite Automata with Distance Functions: An Algebraic Proof”. In: *Theoretical Computer Science* 81.1 (1991), pp. 137–145 (cited on p. 12).

[LPW08] David A. Levin, Yuval Peres, and Elizabeth L. Wilmer. *Markov Chains and Mixing Times.* American Mathematical Society, 2008 (cited on p. 98).

[Mar75] Donald A. Martin. “Borel determinacy”. In: *Annals of Mathematics* 102(2) (1975), pp. 363–371 (cited on p. 24).

[McN66] Robert McNaughton. “Testing and Generating Infinite Sequences by a Finite Automaton”. In: *Information and Control* 9.5 (1966), pp. 521–530. DOI: 10.1016/S0019-9958(66)80013-X. URL: http://dx.doi.org/10.1016/S0019-9958(66)80013-X (cited on p. 10).

[MH84] Satoru Miyano and Takeshi Hayashi. “Alternating Finite Automata on omega-Words”. In: *Theoretical Computer Science* 32 (1984), pp. 321–330. DOI: 10.1016/0304-3975(84)90049-5. URL: http://dx.doi.org/10.1016/0304-3975(84)90049-5 (cited on p. 56).

[MS87] David E. Muller and Paul E. Schupp. “Alternating Automata on Infinite Trees”. In: *Theoretical Computer Science* 54 (1987), pp. 267–276. DOI: 10.1016/0304-3975(87)90133-2. URL: http://dx.doi.org/10.1016/0304-3975(87)90133-2 (cited on pp. 11, 13, 48, 49).

[MS95] David E. Muller and Paul E. Schupp. “Simulating Alternating Tree Automata by Nondeterministic Automata: New Results and New Proofs of the Theorems of Rabin, McNaughton and Safra”. In: *Theoretical Computer Science* 141.1&2 (1995), pp. 69–107. DOI: 10.1016/0304-3975(94)00214-4. URL: http://dx.doi.org/10.1016/0304-3975(94)00214-4 (cited on pp. 11, 13, 48, 49).

[Paz71] Azaria Paz. *Introduction to Probabilistic Automata.* Academic Press, 1971 (cited on pp. 17, 86).

[Pin09] Jean-Éric Pin. “Profinite Methods in Automata Theory”. In: *STACS.* 2009, pp. 31–50. DOI: 10.4230/LIPIcs.STACS.2009.1856. URL: http://dx.doi.org/10.4230/LIPIcs.STACS.2009.1856 (cited on p. 139).

[Rab69] Michael O. Rabin. “Decidability of Second-Order Theories and Automata on Infinite Trees”. In: *Transactions of the AMS* 141 (1969), pp. 1–23 (cited on pp. 10, 11).

[Rab63] Michael O. Rabin. “Probabilistic Automata”. In: *Information and Control* 6.3 (1963), pp. 230–245. DOI: `10.1016/S0019-9958(63)90290-0`. URL: `http://dx.doi.org/10.1016/S0019-9958(63)90290-0` (cited on pp. 11, 16, 82–84, 133).

[RS59] Michael O. Rabin and Dana Scott. “Finite Automata and Their Decision Problems”. In: *IBM Journal of Research and Development* 3.2 (1959), pp. 114–125. ISSN: 0018-8646. DOI: `10.1147/rd.32.0114` (cited on p. 10).

[Sch61] Marcel-Paul Schützenberger. “On the Definition of a Family of Automata”. In: *Information and Control* 4.2-3 (1961), pp. 245–270 (cited on pp. 11, 16).

[Sch56] Marcel-Paul Schützenberger. “Une Théorie Algébrique du Codage”. In: *Séminaire Dubreil-Pisot* (1956) (cited on p. 10).

[Ser06] Olivier Serre. “Contribution à l’Étude des Jeux sur des Graphes de Processus à Pile”. PhD thesis. Université Paris 7 - Denis Diderot, 2006, pp. 1–253 (cited on p. 31).

[Ser03] Olivier Serre. “Note on Winning Positions on Pushdown Games with ω-regular Conditions”. In: *Inf. Process. Lett.* 85.6 (2003), pp. 285–291 (cited on p. 31).

[Sha53] Lloyd S. Shapley. “Stochastic Games”. In: *National Academy of Science USA* 39 (1953), pp. 1095–1100 (cited on p. 13).

[Sim90] Imre Simon. “Factorization Forests of Finite Height”. In: *Theoretical Computer Science* 72.1 (1990), pp. 65–94 (cited on p. 111).

[Sim94] Imre Simon. “On Semigroups of Matrices over the Tropical Semiring”. In: *ITA* 28.3-4 (1994), pp. 277–294 (cited on pp. 12, 111, 112, 139).

[Tor11] Szymon Toruńczyk. “Languages of Profinite Words and the Limitedness Problem”. PhD thesis. University of Warsaw, 2011 (cited on pp. 112, 139).

[Tra62] Boris A. Trakhtenbrot. “Finite Automata and Monadic Second-Order Logic (in Russian)”. In: *Siberian Mathematical Journal* 3 (1962), pp. 103–131 (cited on p. 10).

[Tze92] Wen-Guey Tzeng. “A Polynomial-Time Algorithm for the Equivalence of Probabilistic Automata”. In: *SIAM Journal of Computation* 21.2 (1992), pp. 216–227 (cited on p. 16).

[Van11] Michael Vanden Boom. “Weak Cost Monadic Logic over Infinite Trees”. In: *MFCS*. 2011, pp. 580–591. DOI: `10.1007/978-3-642-22993-0_52`. URL: `http://dx.doi.org/10.1007/978-3-642-22993-0_52` (cited on pp. 15, 20, 53, 56, 70).

[Var98] Moshe Y. Vardi. “Reasoning about The Past with Two-Way Automata”. In: *ICALP*. 1998, pp. 628–641 (cited on p. 11).

[VW08] Moshe Y. Vardi and Thomas Wilke. “Automata: from logics to algorithms”. In: *Logic and Automata: History and Perspectives [in Honor of Wolfgang Thomas]*. 2008, pp. 629–736 (cited on p. 10).

[Wal01] Igor Walukiewicz. “Pushdown Processes: Games and Model-Checking”. In: *Information and Computation* 164.2 (2001), pp. 234–263 (cited on p. 11).

[Zie98] Wiesław Zielonka. “Infinite Games on Finitely Coloured Graphs with Applications to Automata on Infinite Trees”. In: *Theor. Comput. Sci.* 200.1-2 (1998), pp. 135–183 (cited on p. 13).

[Zim12] Martin Zimmermann. “Solving Infinite Games with Bounds”. PhD thesis. RWTH Aachen University, 2012 (cited on p. 14).

[ZP96] Uri Zwick and Mike Paterson. “The Complexity of Mean-Payoff Games on Graphs”. In: *Theoretical Computer Science* 158.1&2 (1996), pp. 343–359. DOI: `10.1016/0304-3975(95)00188-3`. URL: `http://dx.doi.org/10.1016/0304-3975(95)00188-3` (cited on p. 14).