ÉCOLE NORMALE SUPÉRIEURE DE CACHAN
Laboratoire Spécification et Vérification

Games and Automata: From Boolean to Quantitative Verification

Mémoire d'Habilitation à Diriger des Recherches

Laurent DOYEN
CNRS

December 2011

ÉCOLE NORMALE SUPÉRIEURE DE CACHAN
Laboratoire Spécification et Vérification

Games and Automata:
From Boolean to Quantitative Verification

Mémoire d'Habilitation à Diriger des Recherches

Laurent DOYEN
CNRS

Jury:

- Rajeev Alur (rapporteur)
- Ahmed Bouajjani (président)
- Alain Finkel
- Jean Goubault-Larrecq
- Rupak Majumdar
- Joël Ouaknine
- Moshe Y. Vardi (rapporteur)
- Igor Walukiewicz (rapporteur)

Date de soutenance: 13 mars 2012

Acknowledgments

The work presented in this manuscript has been carried out in the group of Tom Henzinger at École Polytechnique Fédérale de Lausanne (Oct. 2006 - Dec. 2008), in the group of Jean-François Raskin at Université Libre de Bruxelles (Jan. 2009 - Sep. 2009), and in the Laboratoire Spécification et Vérification at ENS Cachan under the auspices of Alain Finkel (since Oct. 2009). Great collaboration with Krishnendu Chatterjee has also been possible through several visits to IST Austria.

I would like to warmly thank Tom, Jean-François, Alain, and Krishnendu, as well as all my other co-authors:

- Dietmar Berwanger
- Thomas Brihaye
- Luboš Brim
- Véronique Bruyère
- Jakub Chaloupka
- Aldric Degorre
- Martin De Wulf
- Marc Ducobu
- Herbert Edelsbrunner
- Gilles Geeraerts
- Raffaella Gentilini
- Hugo Gimbert
- Barbara Jobstmann
- Axel Legay
- Nicolas Maquet
- Nicolas Markey
- Thierry Massart
- Dejan Ničković
- Joël Ouaknine
- Tatjana Petrov
- Sangram Raje
- Philippe Rannou
- Julien Reichert
- Mahsa Shirmohammadi
- Rohit Singh
- Szymon Toruńczyk
- James Worrell

I am also grateful to Rajeev Alur, Ahmed Bouajjani, Jean Goubault-Larrecq, Rupak Majumdar, Joël Ouaknine, Moshe Vardi, and Igor Walukiewicz who have readily agreed to serve as members of the jury.

Aux yeux qui brillent.

Table of Contents

Preamble

This manuscript presents my results in three main research directions followed after my PhD thesis (June 2006) on:

(*i*) practical algorithmic analysis of finite-state automata (antichain algorithms),

(*ii*) games for control and synthesis (including partial-observation and quantitative games), and

(*iii*) theory of quantitative verification (quantitative languages, weighted automata, and beyond).

The most important definitions and key ideas that we have developed towards these results are given. Examples and tables summarize the key insights and results. The intention is to give informal and accessible crisp exposition of the techniques, and to provide a concise summary of the results. The full proofs and details can be found in our papers and technical reports. All results in this thesis have been published (see the publication list, page 99) although we omit systematic references. The results of other authors are explicitly referenced. Furthermore, my own work is referenced in plain numerical style (i.e., [1], [2], etc.) while other works are referenced in alpha-numerical style (e.g., [ACC$^+$10]).

We briefly present in the introductory chapter the main results obtained in other lines of research that are not otherwise covered by this manuscript.

Chapter 1

Introduction

Patience et longueur de temps
Font plus que force ni que rage.

Jean de La Fontaine, *Le lion et le rat*.

1.1 Context

The background of this thesis is the problem of computer-aided verification. The goal is to automatically construct a proof of correctness of a system design. This problem involves three main dimensions: (*i*) a mathematical (i.e., precise and well-defined) *model* of the system, (*ii*) a mathematical *specification* of the requirements for the system, and (*iii*) a mathematical definition of the *correctness relation* between models and specifications. While these dimensions are intertwined and their frontier may not be that sharp in practical cases, it is convenient to make this distinction for the theoretical study of the verification problem.

For example, in a concurrent program where several threads execute in parallel with shared memory and lock-based resource access, the model could be the state-transition graph of the concurrent program, and the specification be deadlock-avoidance defined by a set of blocking states (e.g., states where two threads are mutually waiting for a lock to be released by the other). Typically, the correctness relation in this case is defined as a property of the set of executions of the program, stating that all executions have to avoid the set of blocking states.

In some applications, it would make sense to consider that the model to verify is just one thread of the program, and that the other threads are a specification of the environment in which this thread is executed. Correctness would then require that the parallel composition of the thread model with its environment satisfies the deadlock-free property. This view is particularly useful when the different threads are developed separately and independently. The environment may even be non-deterministic, and the specification require that deadlock is avoided in *all* possible implementations of the environment.

Concerning the third dimension (the correctness relation), while the simplest and most common specifications reduce to *safety* properties (such as deadlock-avoidance), it is also useful to consider *liveness* properties (e.g., to ensure that every lock request be eventually granted). More generally, the semantics of the correctness relation is often more complex. It may sometimes be viewed as part of the specification (e.g., given by a monitor that tracks the program execution), or sometimes as part of the correctness relation (e.g., logic-based description of the correctness semantics).

Finally, this three-dimensional view needs to be extended with a parametric dimension, where each parameter may be universally or existentially quantified. A parameter can be the model itself, with existential quantification: "does there exist a model that satisfies the given specification ? If yes, provide such a model". Framed in this way, the verification question is called the *synthesis problem.* Parameters may also occur as unknown value of constants (typically in the model), or as non-determinism (typically in the specification). Assignments of values to the symbolic constants, and schedulers (or policies) to resolve the non-determinism may be existentially quantified (design parameters) or universally quantified (adversarial parameters).

From the above discussion, splitting the verification questions according to these three (or even four) dimensions may appear arbitrary or artificial. This is partly true. However, this view identifies the foundational milestones that need to be addressed by the theoretical study of the verification question. One may consider various classes of models (finite automata, pushdown automata, counter systems, etc.), various classes of specifications (automata, logics, formal languages, etc.), and various notions of correctness (trace inclusion, (bi)simulation, etc.). The results presented in this thesis are a contribution to this field, and are mostly about new algorithmic solutions to the verification question in games and automata theory, complexity results, and expressiveness comparison.

In the first part of this thesis, we consider finite-state models (such as directed graphs, automata, and games), linear-time logic specifications (LTL), and classical notions of correctness defined as Boolean questions (with Yes/No answer).

For example, the *satisfiability* question for LTL specifications asks, given an LTL formula φ, whether there exists an execution that satisfies φ, and the *model-checking* question asks whether all executions of a given finite-state model satisfy the formula φ. The same questions can be asked for automata-based specifications. In the context of (two-player) games, the *synthesis* question asks whether there exists a strategy for one player such that for all strategies of the other player, the resulting execution satisfies a given specification (or objective). In the automata-based approach to verification [VW86], satisfiability questions typically reduce to emptiness of automata, model-checking questions reduce to language inclusion, and synthesis questions reduce to realizability, or simply game solving.

The automata-based approach is essential for the design and analysis of computation systems [VW86, VW94]. While finite-state graphs and automata are relatively simple models, they allow to capture important properties of complex systems (circuits, networks, hierarchical designs, embedded systems, etc.). For example, the automata on finite words are attractive as a model of computation for several reasons: they are robust, in the sense that their expressive power is equivalent to most of their variants (e.g., with or without silent transitions, deterministic or not, etc.) as well as to other natural formalisms like regular expressions or alternating automata; they are *closed* under all Boolean operations, which is important for compositional design; their decision problems (emptiness, universality, language inclusion) are *decidable*. The situation is similar for automata on infinite words, although the verification problems are in general more difficult to solve in practice.

Games and the realizability problem have been studied for long as a fundamental theoretical problem [Chu62, BL69, Rab72, GH82], and for their connection with logic [Rab69, EJ91]. Important theoretical results include the determinacy of two-player games with Borel objective [Mar98] (if one player does not have a winning strategy for a given objective, then the other player has a winning strategy for the complementary objective), and the deep connection of mu-calculus and parity games [EJ91]. Tight connections exist between games and automata [GTW02], and of special interest in this context is the fact that language emptiness of finite automata is a special case of one-player perfect-information game, while language inclusion (as well as language universality) can be viewed as a two-player game with partial observation [38].

In the second part of this thesis, we present a line of research that aims at extending the traditional verification from a Boolean question to a quantitative question. Traditional specifications describe the set of admissible behaviors, and a model satisfies a specification if all its behaviors are admissible, which is a Yes/No question. This classical approach can be extended to quantitative specifications that describe not only the set of admissible implementations but also the *value* of each behavior, in terms of weight or cost. For example, in a lock-placement problem for concurrent programs, a typical requirement is that the locks are used in such a way that deadlocks are avoided, and that all lock requests are eventually granted. However, this leaves room for unrealistic and inefficient program implementations that for example acquire and release all locks all together for each computation step, or schedulers that grant lock requests with arbitrarily long delays. While designers would naturally tend to minimize the number of locks, and try to reduce the delays, an automatic solver may come up with such inefficient solutions. A quantitative specification that assigns a cost to lock usage, and a solver that minimizes the cost of a solution reflect the fact that among two correct programs, we prefer one that avoids unnecessary locks. In the same way, the Boolean requirement that all lock requests are eventually granted can be refined by the quantitative requirement that the (maximal, or average) waiting time for a grant should be minimized.

The theoretical study of quantitative verification leads us to consider quantitative generalizations of languages and automata, games with quantitative objective, and either perfect information (corresponding the quantitative version of language emptiness) or partial information (corresponding the quantitative version of language inclusion and universality), as well as games with combination of Boolean and quantitative objectives.

Several previous works have suggested to use quantitative specifications [dAHM03, HK97] or quantitative logics [DGJP03, LLM05, MM02], and have defined notions of correctness based on quantitative simulation and metrics [dAFS04, DGJP99]. The quantitative approach to verification and synthesis presented in this thesis was also inspired by the work of Henzinger et al. [CdHS03, CHJ05, BCHJ09, Hen10].

1.2 Content

In Chapter 2, we present an overview of the *antichain algorithms* that we have introduced in [39] with Martin De Wulf and Jean-François Raskin for solving partial-observation games. Antichain algorithms are new efficient algorithms for solving classical problems in game theory, as well as in logics and automata theory: emptiness, universality, and language inclusion for finite and Büchi automata, LTL satisfiability and model-checking, partial-observation games, and evaluation of quantified Boolean formulas. The implementation of antichain algorithms in our tools Alaska [45] and Alpaga [44] show dramatic performance improvements (by orders of magnitude) over state-of-the-art tools.

Recently, it has come to our knowledge that analogous ideas have been outlined independently in [RHN00] for universality of finite word automata, and in [TH03] for language containment of finite tree automata. In this chapter, we present a more general view of antichain algorithms, and we present a range of applications beyond finite words and finite trees. Most of the material presented in Chapter 2 appeared in [9, 12, 25] and consists of a survey of the results in [6, 34, 36, 37, 38, 39, 44, 45].

In Chapter 3, we study algorithmic problems for *games with quantitative objective*, motivated by applications in modeling of embedded systems with resource constraints. We consider two-player games played on a weighted graph, with energy and mean-payoff objectives. In an energy game, the weights represent resource consumption and the objective of the game is to maintain the sum of weights always nonnegative. In a mean-payoff game, the objective is to optimize the limit-average usage of the resource.

First, we present a new pseudo-polynomial algorithm for deciding the winner in energy games, and since energy games are log-space equivalent to mean-payoff games, and memoryless optimal strategies exist, we obtain a new algorithm for mean-payoff games, improving the best known worst-case complexity of [ZP96] for this problem from $O(|E| \cdot |Q|^2 \cdot W)$ to $O(|E| \cdot |Q| \cdot W)$ where E is the set of edges in the game, Q is the set of states, and W is the largest weight (in absolute value) in the game.

Second, we show that partial-observation games are much more complicated: infinite memory may be required for mean-payoff games, whereas finite memory suffice for energy games. We show that the problem of deciding the winner in partial-observation energy games is undecidable but co-r.e., while partial-observation mean-payoff games are neither r.e. nor co-r.e.

Third, when the energy objective is combined with a parity objective we show that exponential memory is required in general, still the problem of deciding the winner in energy parity games is in NP $\cap$ coNP. We present a conceptually simple algorithm to solve energy parity games, and we show that it can be used to solve mean-payoff parity games by establishing polynomial-time equivalence of the two problems. This equivalence is not obvious because winning strategies in mean-payoff parity games require infinite memory in general, while finite memory suffices in energy parity games. It follows that mean-payoff parity games can be solved in NP $\cap$ coNP.

Finally, we consider generalized mean-payoff and energy games which replace individual weights by tuples, and require that the limit average (resp., running sum) of each coordinate be (resp., remain) nonnegative. We give an optimal coNP-complete bound for solving generalized energy games, while the previously best known upper bound was EXPSPACE, and no non-trivial lower bound was known.

The results in Chapter 3 have been published in [2, 23, 18, 22].

In Chapter 4, we present *quantitative generalizations of classical languages*, which assign to each word a real number instead of a Boolean value, and have applications in modeling resource-constrained computation. We use weighted automata (finite automata with transition weights) to define several natural classes of quantitative languages over finite and infinite words; in particular, the real value of an infinite run is computed as the maximum, limsup, liminf, limit average, or discounted sum of the transition weights.

First, we define the classical decision problems of automata theory (emptiness, universality, language inclusion, and language equivalence) in the quantitative setting and study their computational complexity. As the decidability of the language-inclusion problem remains open for some classes of weighted automata, we introduce a notion of quantitative simulation that is decidable and implies language inclusion.

Second, we give a complete characterization of the expressive power of the various classes of weighted automata. In particular, we show that most classes of weighted automata cannot be determinized.

Third, for quantitative languages L_1 and L_2, we consider the operations $\max(L_1, L_2)$, $\min(L_1, L_2)$, and $1 - L_1$, which generalize the Boolean operations on languages, as well as the sum $L_1 + L_2$. We establish the closure properties of all classes of quantitative languages with respect to these four operations.

Finally, we introduce a new class of quantitative languages, defined by *mean-payoff automaton expressions*, which is robust and decidable: it is closed under the four pointwise operations, and we show that all decision problems are decidable for this class. Mean-payoff automaton expressions subsume deterministic mean-payoff automata, and we show that they have expressive power incomparable to nondeterministic and alternating mean-payoff automata. We also present for the first time an algorithm to compute the distance between two quantitative languages, and in our case the quantitative languages are given as mean-payoff automaton expressions. The material presented in Chapter 4 appeared in [4, 5, 19, 27, 33].

1.3 Other Contributions

We briefly outline other research directions and results that are not presented in this manuscript.

Timed and hybrid systems. The algorithmic verification of timed and hybrid systems was the topic of my phD thesis. Timed and hybrid automata extend finite-state automata with variables that evolve continuously with time, according to differential equations (or differential constraints).

I have kept a sharp interest in real-time systems, and contributed the following recent results. In [30] we study the realizability problem for specifications of reactive systems expressed in real-time linear temporal logics. The logics we consider are subsets of MITL (Metric Interval Temporal Logic), a logic for which the satisfiability and validity problems are decidable, a necessary condition for the realizability problem to be decidable. On the positive side, we show that the realizability of LTL extended with past real-time formulas is decidable in 2EXPTIME, with a matching lower bound. On the negative side, we show that a simple extension of this decidable fragment with future real-time formulas leads to undecidability. In particular, our results imply that the realizability problem is undecidable for ECL (Event Clock Logic), and therefore also for MITL.

In [13] we investigate the time-bounded version of the reachability problem for hybrid automata. The problem asks, given an hybrid automaton, a target state q, and a rational time bound T, whether the automaton can reach the state q *within* T time units. In contrast to the classical (unbounded) reachability problem, we show that the timed-bounded version is decidable for rectangular hybrid automata provided only non-negative rates are allowed for the continuous variables. We also show that the problem becomes undecidable if either diagonal constraints or both negative and positive rates are allowed.

Markov decision processes (MDP). With Tom Henzinger and Jean-François Raskin, we have developed a new algorithm to decide the problem of probabilistic trace equivalence between labeled Markov chains [8]. Two Markov chains are equivalent if they produce every finite trace with the same probability. The decidability of this question for MDPs (say with existentially quantified scheduler) is open.

We have obtained several results on partially-observable MDPs. In [20], we present a complete picture of the computational complexity of the qualitative analysis problem for partially-observable MDPs with parity objectives and its subclasses: safety, reachability, Büchi, and coBüchi objectives. We establish several upper and lower bounds that were not known in the literature. We also give optimal bounds (matching upper and lower bounds) for the memory required by pure and randomized observation-based strategies for each class of objectives. In [21], we present a complete characterization for the classes of stochastic games and MDPs where randomness is not helpful in: (a) the transition function (probabilistic transition can be simulated by deterministic transition); and (b) strategies (pure strategies are as powerful as randomized strategies). As a consequence of our characterization we obtain new undecidability results for these games, as for example that deciding the existence of a strategy that is winning with probability 1 (i.e., almost-surely) in turn-based partial-observation games with coBüchi objective is undecidable even in the special case where the opponent player has perfect observation.

With Thierry Massart and our student Mahsa Shirmohammadi, we consider in [16, 47] a new semantics for probabilistic automata where on an input word the automaton produces a sequence of probability distributions over states. An infinite word is accepted if the produced sequence is synchronizing, i.e. the sequence of the highest probability in the distributions tends to 1. We show that this semantics generalizes the classical notion of synchronizing words for deterministic automata. We consider the emptiness problem, which asks whether some word is accepted by a given probabilistic automaton, and the universality problem, which asks whether all words are accepted. We provide reductions to establish the PSPACE-completeness of the two problems.

Partial-observation games. We mention some results about partial-observation games in Chapter 2 (Section 2.4.2) and in Chapter 3 (Section 3.4). Partial-observation games are an active research field of ours. Several results have been obtained, and two surveys have appeared on this topic [1, 17]. Other results include the tool Alpaga which solves parity games with partial observation, and constructs winning strategies. The implementation is based on the algorithm presented in [31] and the tool itself was presented in [44]. Finally, with Dietmar Berwanger we have shown that under partial observation, parity games are polynomially reducible to safety games, showing that these games are EXPTIME-complete even for a safety objective [32]. The existence of such a reduction is a major open question for perfect observation.

Interfaces and sequential circuits. Interfaces describe the possible interactions of a component (e.g., a subroutine) with its environment (e.g., the calling program), specifying an input assumption that the environment is expected to satisfy (generalizing the classical pre-condition) and an output guarantee provided by the component to the environment (generalizing the classical post-condition).

General theories of interfaces have been proposed, in both static (or combinatorial) and dynamic (or temporal/sequential) frameworks [dAH01b, dAH01a, LX01]. Such theories support incremental design and independent implementability. Incremental design means that the compatibility checking of interfaces can proceed for partial system descriptions, without knowing the interfaces of all components. Independent implementability means that compatible interfaces can be refined separately, maintaining compatibility.

In [35], we show that these interface theories provide no formal support for component reuse, meaning that the same component cannot be used to implement several different interfaces in a design. For example, different interfaces for the same component may refer to different aspects such as functionality, timing, and power consumption. We add a new operation to interface theories in order to support such reuse. We give both combinatorial and sequential examples for interface theories with component reuse.

In this line of work, we studied in [24] the robustness of digital components embedded in an environment that can provide inaccurate input data to the component. Intuitively, a component is robust if the presence of a small change in the input sequence does not result in a drastic change in the output sequence. We introduce a definition of robustness for sequential circuits as a form of continuity, and we characterize the class of sequential circuits that are robust according to our definition. We present an algorithm to decide whether a sequential circuit is robust or not.

Chapter 2

Antichain Algorithms

Etre raisonnable, toujours, en toutes circonstances ?!
Il faudrait être fou...

Raymond Devos, *Un jour sans moi*.

2.1 Introduction

Finite state-transition systems play a central role in the design and verification of program and circuit models. One of the essential model-checking questions is the *reachability problem* which asks, given an initial state s and a final state s', if there exists a (finite) path from s to s'. For reactive (non-terminating) programs, the *repeated reachability problem* asks, given an initial state s and a final state s', if there exists an infinite path from s that visits s' infinitely often.

The (repeated) reachability problem underlies important verification questions. For example, in the automata-based approach to model-checking [VW86, VW94], the correctness of a program A with respect to a specification B (where A and B are finite automata) is defined by the language inclusion $L(A) \subseteq L(B)$, that is all traces of the program (executions) should be traces of the specification. The language inclusion problem is equivalent to the *emptiness problem* "is $L(A) \cap L^c(B)$ empty ?" where $L^c(B)$ is the complement of $L(B)$. If G is a transition system (or an automaton) defined as the product of A with an automaton B^c obtained by complementation of B, then the emptiness problem can be viewed as a reachability question on G for automata on finite words, and as a repeated reachability question for Büchi automata on infinite words. Note that complementation procedures resort to exponential subset constructions [MS72, Saf88, KV01]. Therefore, while the (repeated) reachability problem, which is NLogSpace-complete, can be solved in linear time in the size of G, the language inclusion problem, which is PSpace-complete, requires exponential time (in the size of B). In practice, implementations for finite words give reasonably good re-

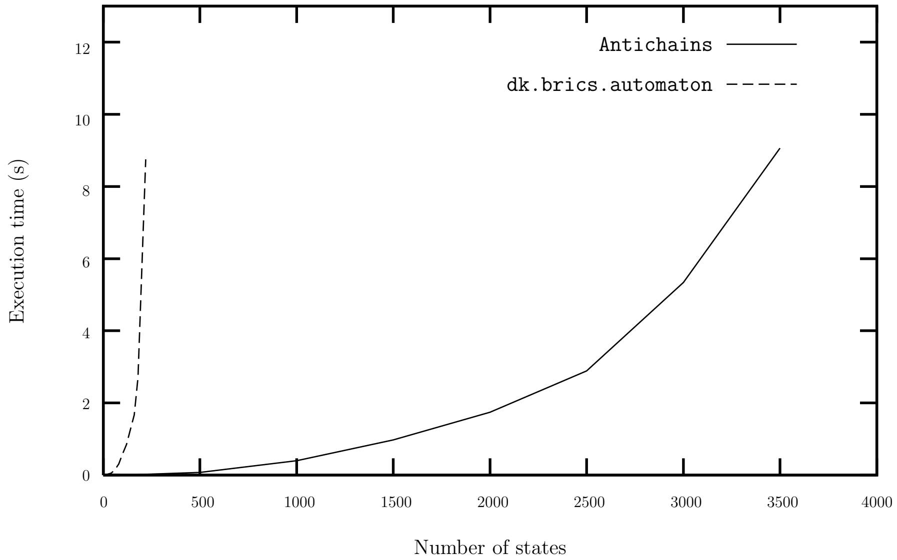


Figure 2.1: Average execution time for the antichain and classical algorithms.

sults (see e.g. [TV05]), while the complementation constructions for infinite words are difficult to implement (even if they have been improved recently [Sch09]) and automata with more than around ten states are intractable [GKSV03, TV07].

Recently, dramatic performance improvements have been obtained by so-called *antichain algorithms* for the reachability and repeated reachability problems on the subset construction and its variants for infinite words [38, BHH$^+$08, 6, FV10, ACC$^+$11]. Figure 2.1 shows the execution time of an antichain algorithm for deciding universality of a family of randomly generated finite automata of increasing size, as compared to the best previously known implementation for this problem [38].

The main idea of antichain algorithms is to exploit the special structure of the subset constructions. As an example, consider the classical subset construction for the complementation of automata on finite words. States of the complement automaton are sets of states of the original automaton, that we call *cells* and denote by s_i. Set inclusion between cells is a partial order that turns out to be a simulation relation for the complement automaton: if $s_2 \subseteq s_1$ and there is a transition from s_1 to s_3, then there exists a transition from s_2 to some $s_4 \subseteq s_3$. This structural property carries over to the sets of cells manipulated by reachability algorithms: if $s_2 \subseteq s_1$ and a final cell can be reached from s_1, then a final cell can also be reached from s_2. Therefore, in a breadth-first search algorithm with backward state traversal, if s_1 is visited by the algorithm, then s_2 is visited simultaneously: the algorithm manipulates $\subseteq$-downward closed sets of cells that can be canonically and compactly represented by the *antichain* of their $\subseteq$-maximal elements. Antichains serve as a symbolic data-structure on which efficient symbolic operations can be defined. Antichain algorithms have been implemented for automata on finite words [38], on finite trees [BHH$^+$08], on infinite words [6, FV09], and for other applications where exponential constructions are involved such as model-checking of linear-time logic [34], partial-observation games [9, 31], and synthesis of linear-time specifications [FJR09]. They outperform explicit and BDD-based algorithms by orders of magnitude [45, 44, FJR09].

In Section 2.3, we present an abstract theory to justify the correctness of antichain algorithms. For backward state traversal algorithms, we first show that forward simulation relations (such as set inclusion in the above example) are required to maintain closed sets in the algorithms. This corresponds to view antichains as a suitable symbolic data-structure to represent closed sets (as above). Then, we develop a different view in which antichains are sets of *promising states* in the (repeated) reachability analysis. This view is justified by mean of backward simulation relations. In our example, it turns out that set inclusion is also a backward simulation which implies that if $s_2 \subseteq s_1$ and s_2 is reachable, then s_1 is also reachable. Therefore, an algorithm which traverses the state space in a backward fashion need not to explore the predecessors of s_2 if s_1 has been visited previously by the algorithm. We say that s_1 is more promising[1] than s_2. As a consequence, the algorithms can safely drop non-$\subseteq$-maximal cells, hence keeping $\subseteq$-maximal cells only. While the two views coincide when set inclusion is used for finite automata, we show that provably better antichain algorithms are obtained when coarser (hence improved) simulation relations are used: fixed points can be reached in fewer iterations, and the antichains that are manipulated are smaller. Dual results are obtained for forward state traversal algorithms.

In Section 2.4, we revisit classical problems of automata theory: the universality problem for nondeterministic automata, the emptiness problem for alternating automata on finite and infinite words, and the emptiness of a product of automata. In such applications, the transition systems to explore are of exponential size and thus they are not constructed prior to the reachability analysis, but explored on-the-fly. And consequently, simulation relations needed by the antichain algorithms should be given without any computation on the transition system itself (which is the case of set inclusion for the subset construction). However, we show that by computing a simulation relation on the original automaton, coarser simulation relations can be induced on the exponential constructions. On the way, we introduce a new notion of backward simulation for alternating automata. Finally, we mention applications of antichain algorithms for solving partial-observation games, and evaluating quantified Boolean formulas.

[1]Note that this is not a heuristic: if s_1 is more promising that s_2, then the exploration of the predecessors of s_2 can be omitted without spoiling the correctness of the analysis.

2.2 Preliminaries

Relations A *pre-order* over a finite set V is a binary relation $\preceq \subseteq V \times V$ which is reflexive and transitive. If $v_1 \preceq v_2$, we say that v_1 is smaller than v_2 (or v_2 is greater than v_1). A pre-order $\preceq'$ is *coarser* than $\preceq$ if for all $v_1, v_2 \in V$, if $v_1 \preceq v_2$, then $v_1 \preceq' v_2$. The $\preceq$-*upward closure* of a set $S \subseteq V$ is the set $\mathsf{Up}(\preceq, S) = \{v_1 \in V \mid \exists v_2 \in S : v_2 \preceq v_1\}$ of elements that are greater than some element in S. A set S is $\preceq$-*upward-closed* if it is equal to its $\preceq$-upward closure, and $\mathsf{Min}(\preceq, S) = \{v_1 \in S \mid \forall v_2 \in S : v_2 \preceq v_1 \rightarrow v_1 \preceq v_2\}$ denotes the minimal elements of S. Note that $\mathsf{Min}(\preceq, S) \subseteq S \subseteq \mathsf{Up}(\preceq, S)$. Analogously, define the $\preceq$-*downward closure* $\mathsf{Down}(\preceq, S) = \{v_1 \in V \mid \exists v_2 \in S : v_1 \preceq v_2\}$ of S, and $\mathsf{Max}(\preceq, S)$ the set of maximal elements[2] of S.

A set $S \subseteq V$ is a *quasi-antichain* if for all $v_1, v_2 \in S$, either v_1 and v_2 are $\preceq$-incomparable, or $v_1 \preceq v_2$ and $v_2 \preceq v_1$. The sets $\mathsf{Min}(\preceq, S)$ and $\mathsf{Max}(\preceq, S)$ are quasi-antichains. A *partial order* is a pre-order which is antisymmetric. For partial orders, the sets $\mathsf{Min}(\preceq, S)$ and $\mathsf{Max}(\preceq, S)$ are *antichains*, i.e., sets of pairwise $\preceq$-incomparable elements. By abuse of language, we call antichains the sets of minimal (or maximal) elements even if the pre-order is not a partial order, and denote by $\mathcal{A}$ the set of antichains over 2^V.

Antichains can be used as a symbolic data-structure for $\preceq$-upward-closed sets. An $\preceq$-upward-closed set S is represented by $\widetilde{S} = \mathsf{Min}(\preceq, S)$. Operations on antichains are defined as follows. The membership question "given v and S, is $v \in S$?" becomes "given v and $\widetilde{S}$, is there $\tilde{v} \in \widetilde{S}$ such that $\tilde{v} \preceq v$?"; the emptiness question is unchanged as $S = \varnothing$ iff $\widetilde{S} = \varnothing$; the relation of set inclusion $S_1 \subseteq S_2$ becomes $\widetilde{S}_1 \sqsubseteq \widetilde{S}_2$ defined by $\forall v_1 \in \widetilde{S}_1 \cdot \exists v_2 \in \widetilde{S}_2 : v_2 \preceq v_1$. If $\langle V, \preceq \rangle$ is a semi-lattice with least upper bound lub, then $\langle \mathcal{A}, \sqsubseteq \rangle$ is a complete lattice (the *lattice of antichains*) where the intersection $S_1 \cap S_2$ is represented by $\widetilde{S}_1 \sqcap \widetilde{S}_2 = \mathsf{Min}(\preceq, \{\mathsf{lub}(v_1, v_2) \mid v_1 \in \widetilde{S}_1 \wedge v_2 \in \widetilde{S}_2\})$ and the union $S_1 \cup S_2$ is represented by $\widetilde{S}_1 \sqcup \widetilde{S}_2 = \mathsf{Min}(\preceq, \widetilde{S}_1 \cup \widetilde{S}_2)$. Analogous definitions exist for antichains of $\preceq$-downward-closed sets if $\langle V, \preceq \rangle$ is a semi-lattice with greatest lower bound. Other operations mixing $\preceq$-upward-closed sets and $\preceq$-downward-closed set can be defined over antichains (such as mixed set inclusion, or emptiness of mixed intersection).

Simulation relations Let $G = (V, E, \mathsf{Init}, \mathsf{Final})$ be a transition system with finite set of states V, transition relation $E \subseteq V \times V$, initial states $\mathsf{Init} \subseteq V$, and final states $\mathsf{Final} \subseteq V$. We define two notions of *simulation* [Mil71]:

- a pre-order $\preceq_{\mathsf{f}}$ over V is a *forward simulation* for G ("$v_2 \preceq_{\mathsf{f}} v_1$" reads v_2 forward simulates v_1) if for all $v_1, v_2, v_3 \in V$, if $v_2 \preceq_{\mathsf{f}} v_1$ and $E(v_1, v_3)$, then there exists $v_4 \in V$ such that $v_4 \preceq_{\mathsf{f}} v_3$ and $E(v_2, v_4)$;

[2] We also denote this set by $\mathsf{Max}(\succeq, S)$, and we equally say that a set is $\preceq$-downward-closed or $\succeq$-downward-closed, etc.

- a pre-order $\succeq_{\mathsf{b}}$ over V is a *backward simulation* for G, ("$v_2 \succeq_{\mathsf{b}} v_1$" reads v_2 backward simulates v_1), if for all $v_1, v_2, v_3 \in V$, if $v_2 \succeq_{\mathsf{b}} v_1$ and $E(v_3, v_1)$, then there exists $v_4 \in V$ such that $v_4 \succeq_{\mathsf{b}} v_3$ and $E(v_4, v_2)$.

The notations $\preceq_{\mathsf{f}}$ and $\succeq_{\mathsf{b}}$ are inspired by the fact that in the subset construction for finite automata, $\subseteq$ is a forward simulation and $\supseteq$ is a backward simulation. Note that a forward simulation for G is a backward simulation for the transition system with transition relation $E^{-1} = \{(v_1, v_2) \mid (v_2, v_1) \in E\}$.

We say that a simulation over V is *compatible* with a set $S \subseteq V$ if for all $v_1, v_2 \in V$, if $v_1 \in S$ and v_2 (forward or backward) simulates v_1, then $v_2 \in S$. Note that a forward simulation $\preceq_{\mathsf{f}}$ is compatible with S if and only if S is $\preceq_{\mathsf{f}}$-downward-closed, and a backward simulation $\succeq_{\mathsf{b}}$ is compatible with S if and only if S is $\succeq_{\mathsf{b}}$-upward-closed. In the sequel, we will be interested in simulation relations that are compatible with Init, or Final, or with both.

Fixpoint algorithms Let $G = (V, E, \mathsf{Init}, \mathsf{Final})$ be a transition system and let $S, S' \subseteq V$ be sets of states. The set of *predecessors* and *successors* of S in one step are denoted $\mathsf{pre}(S) = \{v_1 \mid \exists v_2 \in S : E(v_1, v_2)\}$ and $\mathsf{post}(S) = \{v_1 \mid \exists v_2 \in S : E(v_2, v_1)\}$ respectively. We denote by $\mathsf{pre}^*(S)$ the set $\bigcup_{i \geq 0} \mathsf{pre}^i(S)$ where $\mathsf{pre}^0(S) = S$ and $\mathsf{pre}^i(S) = \mathsf{pre}(\mathsf{pre}^{i-1}(S))$ for all $i \geq 1$, and by $\mathsf{pre}^+(S)$ the set $\bigcup_{i \geq 1} \mathsf{pre}^i(S)$. The operators post^* and post^+ are defined analogously. A finite *path* in G is a sequence $v_0 v_1 \ldots v_n$ of states such that $E(v_i, v_{i+1})$ for all $0 \leq i < n$. Infinite paths are defined analogously. We say that S' is *reachable* from S if there exists a finite path $v_0 v_1 \ldots v_n$ with $v_0 \in S$ and $v_n \in S'$.

The *reachability problem* for G asks if Final is reachable from Init, and the *repeated reachability problem* for G asks if there exists an infinite path starting from Init and passing through Final infinitely many times. To solve these problems, we can use the following classical fixpoint algorithms:

1. The *backward reachability algorithm* computes the sequence of sets:

   $\mathsf{B}(0) = \mathsf{Final}$ and $\mathsf{B}(i) = \mathsf{B}(i-1) \cup \mathsf{pre}(\mathsf{B}(i-1))$ for all $i \geq 1$.

2. The *backward repeated reachability algorithm* computes the sequence of sets:

   $\mathsf{BB}(0) = \mathsf{Final}$ and $\mathsf{BB}(i) = \mathsf{pre}^+(\mathsf{BB}(i-1)) \cap \mathsf{Final}$ for all $i \geq 1$.

3. The *forward reachability algorithm* computes the sequence of sets:

   $\mathsf{F}(0) = \mathsf{Init}$ and $\mathsf{F}(i) = \mathsf{F}(i-1) \cup \mathsf{post}(\mathsf{F}(i-1))$ for all $i \geq 1$.

4. The *forward repeated reachability algorithm* computes the sequence of sets:

   $\mathsf{FF}(0) = \mathsf{Final} \cap \mathsf{post}^*(\mathsf{Init})$ and $\mathsf{FF}(i) = \mathsf{post}^+(\mathsf{FF}(i-1)) \cap \mathsf{Final}$ for all $i \geq 1$.

The above sequences converge to a fixpoint because the operations involved are monotone. We denote by B^*, BB^*, F^*, and FF^* the respective fixpoints. Note that $\mathsf{B}^* = \mathsf{pre}^*(\mathsf{Final})$ and $\mathsf{F}^* = \mathsf{post}^*(\mathsf{Init})$.

Theorem 2.1 *Let $G = (V, E, \mathsf{Init}, \mathsf{Final})$ be a transition system. Then,*

(a) the answer to the reachability problem for G is Yes *if and only if $\mathsf{B}^* \cap \mathsf{Init}$ is nonempty if and only if $\mathsf{F}^* \cap \mathsf{Final}$ is nonempty;*

(b) the answer to the repeated reachability problem for G is Yes *if and only if BB^* is reachable from Init if and only if FF^* is nonempty.*

2.3 Antichain Fixpoint Algorithms

In this section, we show that the sets in the sequences B, BB, F, and FF can be replaced by antichains for well chosen pre-orders. Two views can be developed: when backward algorithms are combined with forward simulation pre-orders (or forward algorithms with backward simulations), antichains are *symbolic representations* of closed sets; when backward algorithms are combined with backward simulation pre-orders (or forward algorithms with forward simulations), antichains are sets of *promising states*. It may be surprising to consider algorithms for the reachability problem (which can be solved in linear time), based on simulation relations (which can be computed in quadratic time). However, such algorithms are useful for applications where the transition systems have a special structure for which simulation relations *need not to be computed*. For example, the relation of set inclusion is always a forward simulation in the subset construction for finite automata (see Section 2.4 for details and other applications). We develop these two views below.

2.3.1 Antichains as a symbolic representation

2.3.1.1 Backward reachability

First, we show that the sets computed by the backward algorithm B are closed for all forward simulations of the transition system G compatible with Final.

Lemma 2.2 *Let $G = (V, E, \mathsf{Init}, \mathsf{Final})$ be a transition system. A pre-order $\preceq_{\mathsf{f}}$ over V is a forward simulation in G if and only if $\mathsf{pre}(S)$ is $\preceq_{\mathsf{f}}$-downward-closed for all $\preceq_{\mathsf{f}}$-downward-closed sets $S \subseteq V$.*

Lemma 2.3 *Let $G = (V, E, \mathsf{Init}, \mathsf{Final})$ be a transition system. If $\preceq_{\mathsf{f}}$ is a forward simulation in G compatible with Final, then $\mathsf{B}(i)$ and $\mathsf{BB}(i)$ are $\preceq_{\mathsf{f}}$-downward-closed sets for all $i \geq 0$.*

Since the sets in the backward algorithms B and BB are $\preceq_{\mathsf{f}}$-downward-closed, we can use the antichain of their maximal elements as a symbolic representation, and adapt the fixpoint algorithms accordingly. Given a forward simulation $\preceq_{\mathsf{f}}$ in G compatible with Final, the antichain algorithm for backward reachability is as follows:

- $\widetilde{\mathsf{B}}(0) = \mathsf{Max}(\preceq_{\mathsf{f}}, \mathsf{Final})$;
- $\widetilde{\mathsf{B}}(i) = \mathsf{Max}(\preceq_{\mathsf{f}}, \widetilde{\mathsf{B}}(i-1) \cup \mathsf{pre}(\mathsf{Down}(\preceq_{\mathsf{f}}, \widetilde{\mathsf{B}}(i-1))))$, for all $i \geq 1$.

Lemma 2.4 *For all $i \geq 0$, $\widetilde{\mathsf{B}}(i) = \mathsf{Max}(\preceq_{\mathsf{f}}, \mathsf{B}(i))$ and $\mathsf{B}(i) = \mathsf{Down}(\preceq_{\mathsf{f}}, \widetilde{\mathsf{B}}(i))$.*

Corollary 2.5 *For all $i \geq 0$, $\mathsf{B}(i+1) = \mathsf{B}(i)$ if and only if $\widetilde{\mathsf{B}}(i+1) = \widetilde{\mathsf{B}}(i)$.*

Using Theorem 2.1, we get the following result.

Theorem 2.6 *$\mathsf{B}^* \cap \mathsf{Init} \neq \varnothing$ if and only if $\mathsf{Down}(\preceq_{\mathsf{f}}, \widetilde{\mathsf{B}}^*) \cap \mathsf{Init} \neq \varnothing$, and therefore the answer to the reachability problem for G is* Yes *if and only if $\mathsf{Down}(\preceq_{\mathsf{f}}, \widetilde{\mathsf{B}}^*) \cap \mathsf{Init} \neq \varnothing$.*

So the antichain algorithm for backward reachability computes exactly the same information as the classical algorithm and the two algorithms reach their fixpoint after exactly the same number of iterations. However, the antichain algorithms can be more efficient in practice if the symbolic representation by antichains is significantly more succinct and if the computations on the antichains can be done efficiently. In particular, the predecessors of the $\preceq_{\mathsf{f}}$-downward-closure of $\widetilde{\mathsf{B}}(i-1)$ needed to obtain $\widetilde{\mathsf{B}}(i)$ should be computed in a way that avoids constructing the $\preceq_{\mathsf{f}}$-downward-closure of $\widetilde{\mathsf{B}}(i-1)$. For applications of the antichain algorithms in automata theory (see also Section 2.4), it can be shown that this operation can be computed efficiently (see e.g. [38, 6]).

Remark Antichains as a data-structure have been used previously for representing the sets of backward reachable states in *well-structured transition systems* [ACJT96, FS01]. So, the sequence $\widetilde{\mathsf{B}}$ converges also when the underlying state space is infinite and $\preceq_{\mathsf{f}}$ is a well-quasi order.

2.3.1.2 Backward repeated reachability

Given a forward simulation $\preceq_{\mathsf{f}}$ in G compatible with Final, the antichain algorithm for repeated backward reachability is as follows:

- $\widetilde{\widetilde{\mathsf{BB}}}(0) = \mathsf{Max}(\preceq_{\mathsf{f}}, \mathsf{Final})$;
- $\widetilde{\widetilde{\mathsf{BB}}}(i) = \mathsf{Max}(\preceq_{\mathsf{f}}, \mathsf{pre}^+(\mathsf{Down}(\preceq_{\mathsf{f}}, \widetilde{\widetilde{\mathsf{BB}}}(i-1))) \cap \mathsf{Final})$, for all $i \geq 1$.

Note that a symbolic representation of $\mathsf{pre}^+(\mathsf{Down}(\preceq_\mathsf{f}, \widetilde{\mathsf{BB}}(i-1))$ can be computed using the antichain algorithm $\widetilde{\mathsf{B}}$ by taking $\widetilde{\mathsf{B}}(0) = \mathsf{Max}(\preceq_\mathsf{f}, \mathsf{pre}(\mathsf{Down}(\preceq_\mathsf{f}, \widetilde{\mathsf{BB}}(i-1))))$. Using Lemma 2.3, we get the following result and corollary.

Lemma 2.7 *$\widetilde{\mathsf{BB}}(i) = \mathsf{Max}(\preceq_\mathsf{f}, \mathsf{BB}(i))$ and $\mathsf{BB}(i) = \mathsf{Down}(\preceq_\mathsf{f}, \widetilde{\mathsf{BB}}(i))$ for all $i \geq 0$.*

Corollary 2.8 *For all $i \geq 0$, $\mathsf{BB}(i+1) = \mathsf{BB}(i)$ if and only if $\widetilde{\mathsf{BB}}(i+1) = \widetilde{\mathsf{BB}}(i)$.*

Using Theorem 2.1, we get the following result.

Theorem 2.9 *BB^* is reachable from Init if and only if $\mathsf{Down}(\preceq_\mathsf{f}, \widetilde{\mathsf{BB}}^*)$ is reachable from Init, and therefore the answer to the repeated reachability problem for G is* Yes *if and only if $\mathsf{Down}(\preceq_\mathsf{f}, \widetilde{\mathsf{BB}}^*)$ is reachable from Init.*

2.3.1.3 Forward algorithms

For the forward algorithms F and FF, results that are dual of Lemma 2.2 and Lemma 2.3 can be obtained, as well as antichain algorithms using backward simulations. The details can be found in [25].

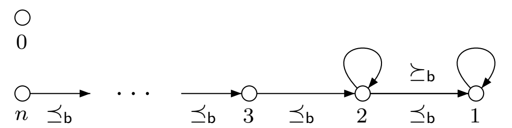
2.3.2 Antichains of promising states

We show that instead of using forward simulations to obtain backward antichain algorithms (or backward simulations for the forward algorithms), we can also use backward simulations to obtain new backward antichain algorithms (or forward simulations to obtain new forward antichain algorithms). These new antichain algorithms do not compute the same information as the original algorithms. In particular, we show that convergence is reached at least as soon as in the original algorithms, but it may be reached sooner. On this basis, we define in Section 2.4 new antichain algorithms that are provably better than the antichain algorithms of [38, 6].

2.3.2.1 Backward reachability

Let $\succeq_\mathsf{b}$ be a backward simulation relation compatible with Init. The *sequence of antichains of backward promising states* is defined as follows:

- $\widehat{\mathsf{B}}(0) = \mathsf{Max}(\succeq_\mathsf{b}, \mathsf{Final})$;
- $\widehat{\mathsf{B}}(i) = \mathsf{Max}(\succeq_\mathsf{b}, \widehat{\mathsf{B}}(i-1) \cup \mathsf{pre}(\widehat{\mathsf{B}}(i-1)))$, for all $i \geq 1$.



$\mathsf{B}(0) =$	$\{1\}$	$\widehat{B}(0) =$	$\{1\} = \widehat{\mathsf{B}}^{\natural}$
$\mathsf{B}(1) =$	$\{1, 2\}$	$\widehat{B}(1) =$	$\{1, 2\}$
$\mathsf{B}(2) =$	$\{1, 2, 3\}$	$\widehat{B}(2) =$	$\{1, 2\}$
...		...	

Figure 2.2: Backward reachability with $\mathsf{Final} = \{1\}$.

Note that while in the sequence $\widetilde{\mathsf{B}}$ we took the $\preceq_{\mathsf{f}}$-downward-closure of $\widetilde{\mathsf{B}}(i-1)$ before computing pre, this is not necessary here. And note that the original sets $\mathsf{B}(i)$ are $\preceq_{\mathsf{f}}$-downward-closed (and represented symbolically by $\widetilde{\mathsf{B}}(i)$), while they are not necessarily $\succeq_{\mathsf{b}}$-downward-closed (here, $\widehat{\mathsf{B}}(i) \subseteq \mathsf{B}(i)$ is a set of most promising states in $\mathsf{B}(i)$). The correctness of this algorithm is justified by monotonicity properties. De define $\sqsubseteq_{\mathsf{b}} \subseteq 2^V \times 2^V$ as follows: $S_1 \sqsubseteq_{\mathsf{b}} S_2$ if $\forall v_1 \in S_1 \cdot \exists v_2 \in S_2 : v_2 \succeq_{\mathsf{b}} v_1$. We write $S_1 \approx_{\mathsf{b}} S_2$ if $S_1 \sqsubseteq_{\mathsf{b}} S_2$ and $S_2 \sqsubseteq_{\mathsf{b}} S_1$.

Lemma 2.10 *The operators* pre*,* $\mathsf{Max}(\succeq_{\mathsf{b}}, \cdot)$*, and* $\cup$ *(and their composition) are* $\sqsubseteq_{\mathsf{b}}$*-monotone.*

Lemma 2.11 *For all* $i \geq 0$*,* $\widehat{\mathsf{B}}(i) \approx_{\mathsf{b}} \mathsf{B}(i)$*.*

Corollary 2.12 (Early convergence) *For all* $i \geq 0$*, (a) if* $\mathsf{B}(i+1) = \mathsf{B}(i)$*, then* $\widehat{\mathsf{B}}(i+1) \approx_{\mathsf{b}} \widehat{\mathsf{B}}(i)$*, and (b)* $\mathsf{B}(i) \cap \mathsf{Init} \neq \varnothing$ *if and only if* $\widehat{\mathsf{B}}(i) \cap \mathsf{Init} \neq \varnothing$*.*

Denote by $\widehat{\mathsf{B}}^{\natural}$ the value $\widehat{\mathsf{B}}(i)$ for the smallest $i \geq 0$ such that $\widehat{\mathsf{B}}(i) \approx_{\mathsf{b}} \widehat{\mathsf{B}}(i+1)$. Corollary 2.12 ensures that convergence (modulo $\approx_{\mathsf{b}}$) on the sequence $\widehat{\mathsf{B}}$ occurs at the latest when B converges. Also, as $\succeq_{\mathsf{b}}$ is compatible with Init, if $\mathsf{B}(i)$ intersects Init then we know that $\widetilde{\mathsf{B}}(i)$ also intersects Init. So, for both positive and negative instances of the reachability problem, we never need to compute more iterations in the $\widehat{\mathsf{B}}$ sequence than of in the B sequence. We establish the correctness of the $\widehat{\mathsf{B}}$ sequence to decide the reachability problem, and using Theorem 2.1, we get the following result.

Theorem 2.13 (Correctness) $\mathsf{B}^* \cap \mathsf{Init} \neq \varnothing$ *if and only if* $\widehat{\mathsf{B}}^{\natural} \cap \mathsf{Init} \neq \varnothing$*, and therefore the answer to the reachability problem for* G *is* YES *if and only if* $\widehat{\mathsf{B}}^{\natural} \cap \mathsf{Init} \neq \varnothing$*.*

Example 2.1 *Consider the transition system in Figure 2.2 where* $\mathsf{Final} = \{1\}$ *and* $\mathsf{Init} = \{0\}$*. The classical backward reachability algorithm computes the sequence* $\mathsf{B}(0) = \{1\}$*,* $\mathsf{B}(1) = \{1, 2\}$*, ...,* $\mathsf{B}(i) = \{1, 2, \dots, i+1\}$ *and converges to* $\{1, \dots, n\}$ *after* $O(n)$ *iterations. Consider the backward simulation* $\succeq_{\mathsf{b}}$ *as depicted on Figure 2.2. States 1 and 2 are mutually simulated by each other, and* $i \succeq_{\mathsf{b}} i+1$ *for all* $1 \leq i < n$*.*

The antichain algorithm for backward reachability based on $\succeq_{\mathsf{b}}$ *computes the sequence* $\widehat{B}(0) = \{1\}$, $\widehat{B}(1) = \{1, 2\}$ *and the algorithms halts since* $\widehat{B}(0) \approx_{\mathsf{b}} \widehat{B}(1)$, *i.e.* $\widehat{\mathsf{B}}^{\natural} = \widehat{\mathsf{B}}(0)$. *We get early convergence because state* 1 *is more promising than all other states, yet is not reachable from* Init.

2.3.2.2 Backward repeated reachability

Let $\succeq_{\mathsf{b}}$ be a backward simulation relation compatible with both Final and Init. Using such a relation, we define the *sequence of antichains of backward repeated promising states* as follows:

- $\widehat{\mathsf{BB}}(0) = \mathsf{Max}(\succeq_{\mathsf{b}}, \mathsf{Final})$;
- $\widehat{\mathsf{BB}}(i) = \mathsf{Max}(\succeq_{\mathsf{b}}, \mathsf{pre}^{+}(\widehat{\mathsf{BB}}(i-1)) \cap \mathsf{Final})$, for all $i \geq 1$.

Note that the computation of $S_i = \mathsf{pre}^{+}(\widehat{\mathsf{BB}}(i-1))$ can be replaced by algorithm $\widehat{\mathsf{B}}$ with $\widehat{\mathsf{B}}(0) = \mathsf{Max}(\succeq_{\mathsf{b}}, \mathsf{pre}(\widehat{\mathsf{BB}}(i-1)))$. This yields $\widehat{\mathsf{B}}^{\natural} \approx_{\mathsf{b}} S_i$ which is sufficient to ensure correctness of the algorithm. We have required that $\succeq_{\mathsf{b}}$ is compatible with Final to have the following property.

Lemma 2.14 *The operator* $\lambda S \cdot S \cap \mathsf{Final}$ *is* $\sqsubseteq_{\mathsf{b}}$*-monotone.*

Lemma 2.15 *For all* $i \geq 0$, $\widehat{\mathsf{BB}}(i) \approx_{\mathsf{b}} \mathsf{BB}(i)$.

Corollary 2.16 (Early convergence) *For all* $i \geq 0$, *if* $\mathsf{BB}(i+1) = \mathsf{BB}(i)$ *then* $\widehat{\mathsf{BB}}(i) \approx_{\mathsf{b}} \widehat{\mathsf{BB}}(i+1)$.

Denote by $\widehat{\mathsf{BB}}^{\natural}$ the value $\widehat{\mathsf{BB}}(i)$ for the smallest $i \geq 0$ such that $\widehat{\mathsf{BB}}(i) \approx_{\mathsf{b}} \widehat{\mathsf{BB}}(i+1)$. Using Theorem 2.1, we get the following result.

Theorem 2.17 (Correctness) BB^{*} *is reachable from* Init *if and only if* $\widehat{\mathsf{BB}}^{\natural}$ *is reachable from* Init, *and therefore the answer to the repeated reachability problem for* G *is* YES *if and only if* $\widehat{\mathsf{BB}}^{\natural}$ *is reachable from* Init.

2.3.2.3 Forward reachability and repeated reachability algorithm

As before, we refer to [25] for the forward algorithms.

Remark In antichain algorithms of promising states, if $\preceq^1$ is coarser than $\preceq^2$, then the induced relation $\approx^1$ on sets of states is coarser than $\approx^2$ which entails that convergence modulo $\approx^1$ occurs at the latest when convergence modulo $\approx^2$ occurs, and possibly earlier. This will be illustrated in Section 2.4.

2.4 Applications

In this section, we present applications of the antichain algorithms to solve classical (and computationally hard) problems in automata theory, partial-observation games, and QBF evaluation (quantified Boolean formulas).

2.4.1 Automata theory

We consider automata running on finite and infinite words.

An *alternating automaton* is a tuple $A = (Q, q_\iota, \Sigma, \delta, \alpha)$ where:

- Q is a finite set of states;
- $q_\iota \in Q$ is the initial state;
- Σ is a finite alphabet;
- $\delta : Q \times \Sigma \to 2^{2^Q}$ is the transition relation that maps each state q and letter σ to a set $\{C_1, \dots, C_n\}$ where each $C_i \subseteq Q$ is a *choice*;
- $\alpha \subseteq Q$ is the set of accepting states.

In an alternating automaton, the input word $w = \sigma_0\sigma_1 \dots$ over Σ is processed by two players in a turn-based game played in rounds. Each round starts in a state of the automaton, and the first round starts in q_ι. In round i, the first player makes a choice $C \in \delta(q_i, \sigma_i)$ where q_i is the state in round i and σ_i is the i^{th} letter of the input word. Then, the second player chooses a state $q_{i+1} \in C$, and the next round starts in q_{i+1}. A finite input word is accepted by A if the first player has a strategy to force an accepting state of A in the last round; an infinite input word is accepted by A if the first player has a strategy to force infinitely many rounds to be in an accepting state of A. A run of an alternating automaton corresponds to a fixed strategy of the first player, and contains all possible outcomes that are consistent this strategy of the first player.

Formally, a *run* of A over a (finite or infinite) word $w = \sigma_0\sigma_1 \dots$ is a directed acyclic graph $T_w = \langle N, \to \rangle$ where:

- $N = Q \times \mathbb{N}$ is the set of nodes. A node (q, i) represents the state q after the first i letters of the word w have been processed. Nodes of the form (q, i) with $q \in \alpha$ are called α-*nodes*;
- and $\to \subseteq V \times V$ is such that (i) if $(q, i) \to (q', i')$ then $i' = i + 1$ and (ii) for every $(q, i) \in V$, there exists $C \in \delta(q, \sigma_i)$ such that $C \subseteq \{q' \mid (q, i) \to (q', i + 1)\}$. We say that $(q', i + 1)$ is a *successor* of (q, i) if $(q, i) \to (q', i + 1)$, and we say that (q', i') is *reachable* from (q, i) if $(q, i) \to^* (q', i')$.

A run $T_w = \langle N, \rightarrow \rangle$ of A on an a finite word w is *accepting* if all nodes (q, i) with $i = |w|$ reachable from $(q_\iota, 0)$ are α-nodes; and a run $T_w = \langle N, \rightarrow \rangle$ of A on an a infinite word w is *accepting* if all paths from $(q_\iota, 0)$ visit α-nodes infinitely often (i.e., all paths satisfy a Büchi condition). A (finite or infinite) word w is *accepted* by A if there exists an accepting run on w. Alternating automata on finite words are called AFA, and alternating automata on infinite words are called ABW. The *language* of an AFA (resp., ABW) A is the set $L(A)$ of finite (resp., infinite) words accepted by A.

The *emptiness problem* for alternating automata is to decide if the language of a given alternating automaton (AFA or ABW) is empty. This problem is PSpace-complete for both AFA and ABW [MS72, SVW87]. For finite words, we also consider the *universality problem* which is to decide if the language of a given AFA with alphabet Σ is equal to Σ^*, and also PSpace-complete even for the special case of nondeterministic automata. A *nondeterministic automaton* (NFA) is an AFA such that $\delta(q, \sigma)$ is a set of singletons for all states q and letters σ.

We use antichain algorithms to solve the emptiness problem of AFA and ABW, as well as the universality problem for NFA, and the emptiness problem for NFA specified by a product of automata. In the case of NFA, it is more convenient to represent the transition relation as a function $\delta : Q \times \Sigma \rightarrow 2^Q$ where $\delta(q, \sigma) = \{q_1, \dots, q_n\}$ represents the set of singletons $\{\{q_1\}, \dots, \{q_n\}\}$.

2.4.1.1 Universality problem for NFA

Let $A = (Q, q_\iota, \Sigma, \delta, \alpha)$ be an NFA, and define the subset construction $G(A) = (V, E, \mathsf{Init}, \mathsf{Final})$ as follows: $V = 2^Q$, $E(v_1, v_2)$ if there exists $\sigma \in \Sigma$ such that $\delta(q, \sigma) \subseteq v_2$ for all $q \in v_1$, $\mathsf{Init} = \{v \in V \mid q_\iota \in v\}$, and $\mathsf{Final} = \{v \in V \mid v \subseteq Q \setminus \alpha\}$. A classical result shows that $L(A) \neq \Sigma^*$ if and only if Final is reachable from Init in $G(A)$, and thus we can solve the universality problem for A using antichain algorithms for the reachability problem on $G(A)$.

Antichains as symbolic representation Consider the relation $\preceq_{\mathsf{F}}$ on the states of $G(A)$ defined by $v_2 \preceq_{\mathsf{F}} v_1$ if and only if $v_2 \subseteq v_1$. Note that $\preceq_{\mathsf{F}}$ is a partial order.

Lemma 2.18 *$\preceq_{\mathsf{F}}$ is a forward simulation in $G(A)$ compatible with* Final.

The antichain algorithm for backward reachability is instantiated as follows:

- $\widetilde{\mathsf{B}}(0) = \mathsf{Max}(\subseteq, \mathsf{Final}) = \{Q \setminus \alpha\}$;
- $\widetilde{\mathsf{B}}(i) = \mathsf{Max}(\subseteq, \widetilde{\mathsf{B}}(i-1) \cup \mathsf{pre}(\mathsf{Down}(\subseteq, \widetilde{\mathsf{B}}(i-1))))$, for all $i \geq 1$.

Details about efficient computation of this sequence as well as experimental comparison with the classical algorithm based on determinization can be found in [38].

Antichains of promising states Consider the relation $\succeq_{\mathsf{B}}$ such that $v_2 \succeq_{\mathsf{B}} v_1$ if $v_2 \supseteq v_1$. Note that $v_2 \succeq_{\mathsf{B}} v_1$ iff $v_1 \preceq_{\mathsf{F}} v_2$.

Lemma 2.19 $\succeq_{\mathsf{B}}$ *is a backward simulation in* $G(A)$ *compatible with* Init.

The corresponding antichain algorithm for backward reachability is instantiated as follows:

- $\widehat{\mathsf{B}}(0) = \mathsf{Max}(\supseteq, \mathsf{Final}) = \{Q \setminus \alpha\}$;
- $\widehat{\mathsf{B}}(i) = \mathsf{Max}(\supseteq, \widehat{\mathsf{B}}(i-1) \cup \mathsf{pre}(\widehat{\mathsf{B}}(i-1))))$, for all $i \geq 1$.

It should be noted that $\widetilde{\mathsf{B}}(i) = \widehat{\mathsf{B}}(i)$, for all $i \geq 0$. In this particular case, the two views coincide due to the special structure of the transition system $G(A)$ (namely $\subseteq$ is a forward simulation and its inverse $\supseteq$ is a backward simulation).

In the rest of this section, we establish the existence of simulation relations for various constructions in automata theory, and we omit the instantiation of the corresponding antichain algorithms in the promising state view.

Coarser simulations We show that the algorithms based on antichains of promising states can be improved using coarser simulations (obtained by exploiting the structure of the NFA before subset construction). We illustrate this below for backward algorithms and coarser backward simulations.

We construct a backward simulation coarser than $\succeq_{\mathsf{B}}$, using a pre-order $\gg_{\mathsf{b}} \subseteq Q \times Q$ on the state space of A such that for all $\sigma \in \Sigma$, for all $q_1, q_2, q_3 \in Q$, if $q_2 \gg_{\mathsf{b}} q_1$, then

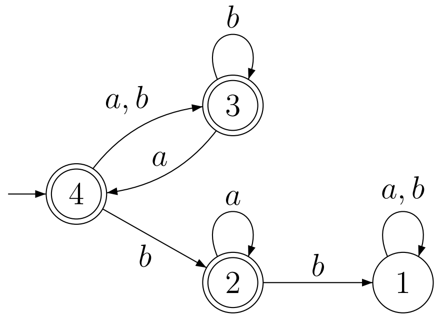
(*i*) if $q_1 = q_\iota$, then $q_2 = q_\iota$, and

(*ii*) if $q_1 \in \delta(q_3, \sigma)$, then there exists $q_4 \in Q$ such that $q_2 \in \delta(q_4, \sigma)$ and $q_4 \gg_{\mathsf{b}} q_3$.

Such a relation $\gg_{\mathsf{b}}$ is usually called a *backward simulation relation* for the NFA A, and a maximal backward simulation relation (which is unique) can be computed in polynomial time (see e.g. [HHK95]). Given $\gg_{\mathsf{b}}$, define the relation $\succeq_{\mathsf{B+}}$ on $G(A)$ as follows: $v_2 \succeq_{\mathsf{B+}} v_1$ if $\forall q_2 \notin v_2 \cdot \exists q_1 \notin v_1 : q_1 \gg_{\mathsf{b}} q_2$.

Lemma 2.20 $\succeq_{\mathsf{B+}}$ *is a backward simulation for* $G(A)$ *compatible with* Init.

Note that $\succeq_{\mathsf{B+}}$ is coarser than $\succeq_{\mathsf{B}}$ because $v_2 \supseteq v_1$ is equivalent to say that for all $q_2 \notin v_2$, there exists $q_1 \notin v_1$ such that $q_1 = q_2$ (which implies that $q_1 \gg_{\mathsf{b}} q_2$ since $\gg_{\mathsf{b}}$ is a pre-order). Therefore, the antichains in the antichain algorithm based on $\succeq_{\mathsf{B+}}$ are subsets of those based on $\succeq_{\mathsf{B}}$. By Corollary 2.12, the number of iterations of the algorithms based on $\succeq_{\mathsf{B+}}$ and $\succeq_{\mathsf{B}}$ is the same when $L(A) \neq \Sigma^*$, and Example 2.2 below shows that the algorithm based on $\succeq_{\mathsf{B+}}$ may converge faster when $L(A) = \Sigma^*$.



	using $\succeq_{\mathsf{B}}$	using $\succeq_{\mathsf{B+}}$
$\widehat{\mathsf{B}}(0) =$	$\{\{1\}\}$	$\{\{1\}\}$
$\widehat{\mathsf{B}}(1) =$	$\{\{1,2\}\}$	$\{\{1\},\{1,2\}\}$
$\widehat{\mathsf{B}}(2) =$	$\{\{1,2\}\}$	end
	end	

Figure 2.3: Improved antichain algorithm for the universality problem of NFA (Example 2.2).

Example 2.2 *Consider the nondeterministic finite automaton A with alphabet $\Sigma = \{a, b\}$ in Figure 2.3. Note that every word is accepted by A i.e., $L(A) = \Sigma^*$ (it suffices to always go to state 3 from state 4). The backward antichain algorithm applied to the subset construction $G(A)$ (using $\succeq_{\mathsf{B}}$) converges after 3 iterations, and the intersection of $\widehat{B}^{\natural} = \{\{1, 2\}\}$ with the initial states of $G(A)$ is empty. Now, let $\gg_{\mathsf{b}}$ be the maximal backward simulation relation for A. We have $3 \gg_{\mathsf{b}} 2$, $3 \gg_{\mathsf{b}} 1$, and $q \gg_{\mathsf{b}} q$ for all $q \in \{1, 2, 3, 4\}$. The induced relation $\succeq_{\mathsf{B+}}$ is such that $\{1\} \succeq_{\mathsf{B+}} \{1, 2\}$ and $\{1, 2\} \succeq_{\mathsf{B+}} \{1\}$. Therefore, $\widehat{\mathsf{B}}(0) \approx_{\mathsf{b}} \widehat{\mathsf{B}}(1)$ and the backward antichain algorithm based on $\succeq_{\mathsf{B+}}$ converges faster, namely after 2 iterations.*

2.4.1.2 Emptiness problem for AFA

In this section, we use a new definition of backward simulation for alternating automata on finite words to construct an induced backward simulation on the subset construction for AFA.

Let $A = (Q, q_\iota, \Sigma, \delta, \alpha)$ be an AFA. Define the subset construction $G(A) = (V, E, \mathsf{Init}, \mathsf{Final})$ where $V = 2^Q$, $E = \{(v_1, v_2) \in V \times V \mid \exists \sigma \in \Sigma \cdot \forall q \in v_1 \cdot \exists C \in \delta(q, \sigma) : C \subseteq v_2\}$, $\mathsf{Init} = \{v \in V \mid q_\iota \in v\}$, and $\mathsf{Final} = \{v \in V \mid q \subseteq \alpha\}$.

As before, it is easy to see that $L(A) \neq \varnothing$ if and only if Final is reachable from Init in $G(A)$, and the emptiness problem for A can be solved using antichain algorithms for the reachability problem in $G(A)$ e.g., using the relation $\succeq_{\mathsf{B}}$ such that $v_2 \succeq_{\mathsf{B}} v_1$ if $v_2 \supseteq v_1$ which is a backward simulation in $G(A)$ compatible with Init.

As in the case of the universality problem for NFA, the relation $\succeq_{\mathsf{B}}$ can be improved using an appropriate notion of backward simulation relation defined on the AFA A. We introduce such a new notion as follows. An *backward alternating simulation relation* for an alternating automaton $A = (Q, q_\iota, \Sigma, \delta, \alpha)$ is a pre-order $\gg_{\mathsf{b}}$ which is the reflexive closure of a relation $>_{\mathsf{b}}$ such that for all $\sigma \in \Sigma$, for all $q_1, q_2, q_3 \in Q$, if $q_2 >_{\mathsf{b}} q_1$, then

(*i*) if $q_1 = q_\iota$, then $q_2 = q_\iota$, and (*ii*) if there exists $C \in \delta(q_3, \sigma)$ such that $q_1 \in C$, then there exists $q_4 \in Q$ such that (*a*) $q_2 \in C'$ for all $C' \in \delta(q_4, \sigma)$, and (*b*) $q_4 >_{\mathsf{b}} q_3$.

It can be shown that a unique maximal backward simulation relation exists for AFA (because the union of two backward simulation relations is again a backward simulation relation), and it can be computed in polynomial time using analogous fixpoint algorithms for computing standard simulation relations [HHK95], e.g. the fixpoint iterations defined by $R_0 = \{(q_1, q_2) \in Q \times Q \mid q_1 = q_\iota \to q_2 = q_\iota\}$ and $R_i = \{(q_1, q_2) \in R_{i-1} \mid \forall q_3 \in Q : (\exists C \in \delta(q_3, \sigma) : q_1 \in C) \to \exists q_4 \in Q : (\forall C' \in \delta(q_4, \sigma) : q_2 \in C') \land (q_3, q_4) \in R_{i-1}\}$ for all $i \geq 1$. Note that for so-called *universal finite automata* (UFA) which are AFA where $\delta(q, \sigma)$ is a singleton for all $q \in Q$ and $\sigma \in \Sigma$, our definition of backward alternating simulation coincides with ordinary backward simulation for the dual of the UFA (which is an NFA with transition relation $\delta'(q, \sigma) = \big\{q \in C \mid \delta(q, \sigma) = \{C\}\big\}$).

As before, given a backward alternating simulation relation $\gg_{\mathsf{b}}$ for A, we define the relation $\succeq_{\mathsf{B+}}$ on $G(A)$ as follows: $v_2 \succeq_{\mathsf{B+}} v_1$ if $\forall q_2 \notin v_2 \cdot \exists q_1 \notin v_1 : q_1 \gg_{\mathsf{b}} q_2$.

Lemma 2.21 $\succeq_{\mathsf{B+}}$ *is a backward simulation in* $G(A)$ *compatible with* Init.

2.4.1.3 Emptiness problem for ABW

The emptiness problem for ABW can be solved using a subset construction due to Miyano and Hayashi [MH84, 34, 6].

Given an ABW $A = (Q, q_\iota, \Sigma, \delta, \alpha)$, define the Miyano-Hayashi transition system $\mathsf{MH}(A) = (V, E, \mathsf{Init}, \mathsf{Final})$ where $V = 2^Q \times 2^Q$, $\mathsf{Init} = \{\langle s, \varnothing\rangle \mid q_\iota \in s \subseteq V\}$ $\mathsf{Final} = 2^Q \times \{\varnothing\}$, and for all $v_1 = \langle s_1, o_1\rangle$, and $v_2 = \langle s_2, o_2\rangle$, we have $E(v_1, v_2)$ if there exists $\sigma \in \Sigma$ such that $\forall q \in s_1 \cdot \exists C \in \delta(q, \sigma) : C \subseteq s_2$, and either (*i*) $o_1 \neq \varnothing$ and $\forall q \in o_1 \cdot \exists C \in \delta(q, \sigma) : C \subseteq o_2 \cup (s_2 \cap \alpha)$, or (*ii*) $o_1 = \varnothing$ and $o_2 = s_2 \setminus \alpha$.

A classical result shows that $L(A) \neq \varnothing$ if and only if there exists an infinite path from Init in $\mathsf{MH}(A)$ that visits Final infinitely many times. Therefore, the emptiness problem for ABW can be reduced to the repeated reachability problem, and we can use an antichain algorithm (e.g., based on forward simulation) for repeated reachability to solve it. We construct a forward simulation for $\mathsf{MH}(A)$ using a classical notion of alternating simulation.

A pre-order $\ll_{\mathsf{f}} \subseteq Q \times Q$ is an *alternating forward simulation relation* [AHKV98] for an alternating automaton A if for all $\sigma \in \Sigma$, for all $q_1, q_2, q_3 \in Q$, if $q_2 \ll_{\mathsf{f}} q_1$, then

(*i*) if $q_1 \in \alpha$, then $q_2 \in \alpha$, and

(*ii*) for all $C_1 \in \delta(q_1, \sigma)$, there exists $C_2 \in \delta(q_2, \sigma)$ such that for all $q_4 \in C_2$, there exists $q_3 \in C_1$ such that $q_4 \ll_{\mathsf{f}} q_3$.

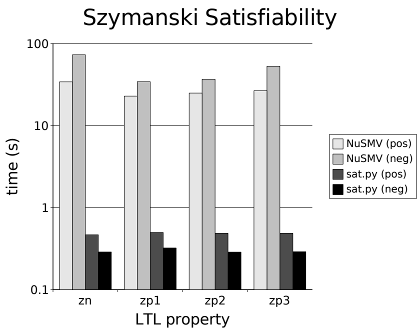


Figure 2.4: Experimental comparison of antichain algorithms and the tool NuSMV for LTL satisfiability.

Given a forward alternating simulation relation $\ll_{\mathsf{f}}$ for A, define the relation $\preceq_{\mathsf{F}^+}$ on $\mathsf{MH}(A)$ such that $\langle s_2, o_2\rangle \preceq_{\mathsf{F}^+} \langle s_1, o_1\rangle$ if the following conditions hold: (a) $\forall q_2 \in s_2 \cdot \exists q_1 \in s_1 : q_2 \ll_{\mathsf{f}} q_1$, (b) $\forall q_2 \in o_2 \cdot \exists q_1 \in o_1 : q_2 \ll_{\mathsf{f}} q_1$, and (c) $o_1 = \varnothing$ if and only if $o_2 = \varnothing$.

Lemma 2.22 $\preceq_{\mathsf{F}^+}$ *is a forward simulation in* $\mathsf{MH}(A)$ *compatible with* Final.

The result of Lemma 2.22 is particularly useful for the satisfiability and model-checking problem for LTL (Linear Temporal Logic) because LTL formulas can be translated in polynomial time to ABW. However, the efficient implementation of antichain algorithms for LTL requires some more care due to the fact that the produced ABW have a symbolic alphabet (transitions are labeled by formulas of propositional logic) and that forward exploration algorithms are much more efficient in practice. Details and performance evaluation can be found in [34, 45]. An example of the performance improvements obtained by antichain algorithm is shown in Figure 2.4 for a set of four LTL formulas describing liveness properties for the Szymanski mutual exclusion protocol and for a variant of this protocol due to Pnueli [STV05]. We compare our prototype with NuSMV [CCGR00] on these four formulas (pos) and their negation (neg).

2.4.1.4 Emptiness problem for a product of NFA

Consider NFAs $A_i = (Q_i, q_\iota^i, \Sigma \cup \{\tau_i\}, \delta_i, \alpha_i)$ for $1 \leq i \leq n$ where $\tau_1, \dots, \tau_n$ are internal actions, and Σ is a shared alphabet. The *synchronized product* $A_1 \otimes A_2 \otimes \cdots \otimes A_n$ is the transition system $(V, E, \mathsf{Init}, \mathsf{Final})$ where

- $V = Q_1 \times Q_2 \times \cdots \times Q_n$;
- $E(v_1, v_2)$ if $v_1 = (q_1^1, q_1^2, \dots, q_1^n)$, $v_2 = (q_2^1, q_2^2, \dots, q_2^n)$ and either $q_2^i \in \delta_i(q_1^i, \tau_i)$ for all $1 \leq i \leq n$, or there exists $\sigma \in \Sigma$ such that $q_2^i \in \delta_i(q_1^i, \sigma)$ for all $1 \leq i \leq n$;

- $\mathsf{Init} = \{(q_\iota^1, q_\iota^2, \ldots, q_\iota^n)\}$;
- $\mathsf{Final} = \alpha_1 \times \alpha_2 \times \cdots \times \alpha_n$.

For each $i = 1 \ldots n$, let $\ll_{\mathsf{f}}^i \subseteq Q_i \times Q_i$ be a forward simulation relation for A_i. Define the relation $\preceq_{\mathsf{F}}$ such that $(q_2^1, q_2^2, \ldots, q_2^n) \preceq_{\mathsf{F}} (q_1^1, q_1^2, \ldots, q_1^n)$ if $q_2^i \ll_{\mathsf{f}}^i q_1^i$ for all $1 \leq i \leq n$.

Lemma 2.23 $\preceq_{\mathsf{F}+}$ *is a forward simulation in* $A_1 \otimes \cdots \otimes A_n$ *compatible with* Final.

2.4.2 Partial-observation games

We consider partial-observation games defined as follows.

A *partial-observation game* (or simply a *game*) is a tuple $G = \langle Q, q_\iota, \Sigma, \Delta, \mathsf{Obs}\rangle$, where Q is a finite set of states, $q_\iota \in Q$ is the initial state, Σ is a finite alphabet, $\Delta \subseteq Q \times \Sigma \times Q$ is a labeled transition relation which is total, i.e. for all $q \in Q$ and $\sigma \in \Sigma$, there exists $q' \in Q$ such that $(q, \sigma, q') \in \Delta$; and $\mathsf{Obs} \subseteq 2^Q$ is a set of *observations* that partition the state space. For each state $q \in L$, we denote by $\mathsf{obs}(q)$ the unique observation $o \in \mathsf{Obs}$ such that $q \in o$; and for $s \subseteq L$ and $\sigma \in \Sigma$, we denote by $\mathsf{post}_\sigma^G(s) = \{q' \in Q \mid \exists q \in s : (q, \sigma, q') \in \Delta\}$ the set of σ-successors of s. A game with *perfect observation* is such that $\mathsf{Obs} = \{\{q\} \mid q \in Q\}$, i.e. every state is observable. We omit the set Obs in perfect-observation games.

Games are played in rounds in which player 1 chooses an action $\sigma \in \Sigma$, and player 2 chooses a σ-successor of the current state. However, player 1 does not see the states chosen by player 2, but only their observation.

The first round starts in the initial state q_ι. A *play* in G is an infinite sequence $\pi = q_0\sigma_0q_1\sigma_1\ldots$ such that $q_0 = q_\iota$ and $(q_i, \sigma_i, q_{i+1}) \in \Delta$ for all $i \geq 0$. The *prefix* up to q_n of the play π is denoted by $\pi(n)$. The set of plays in G is denoted $\mathsf{Plays}(G)$ and the set of corresponding prefixes is written as $\mathsf{Prefs}(G)$. The *observation sequence* of π is the sequence $\mathsf{obs}(\pi) = \mathsf{obs}(q_0)\sigma_0\mathsf{obs}(q_1)\sigma_1\ldots$

A *strategy* (for player 1) in G is a function $\alpha : \mathsf{Prefs}(G) \to \Sigma$ such that $\alpha(\rho) = \alpha(\rho')$ for all prefixes $\rho, \rho' \in \mathsf{Prefs}(G)$ with $\mathsf{obs}(\rho) = \mathsf{obs}(\rho')$. A play $\rho = q_0\sigma_0q_1 \cdots \in \mathsf{Plays}(G)$ is *consistent* with α if $\sigma_i = \alpha(\pi(i))$ for all $i \geq 0$.

An *objective* for G is a set φ of infinite sequences of states and actions, that is, $\varphi \subseteq (Q \times \Sigma)^\omega$. A strategy α of player 1 is *winning* for φ if $\pi \in \varphi$ for all plays π consistent with α. We only consider *observable* objectives such that if $\rho \in \varphi$ and $\mathsf{obs}(\rho) = \mathsf{obs}(\rho')$, then $\rho' \in \varphi$. We consider *reachability objectives* defined by a set $\mathcal{T} \subseteq \mathsf{Obs}$ of target observations, and *parity objectives* defined by a priority function $p : \mathsf{Obs} \to \mathbb{N}$ that requires that the minimal priority visited infinitely often be even. Formally, $\mathsf{Reach}_G(\mathcal{T}) = \{\pi = q_0\sigma_0q_1 \cdots \mid \exists i \geq 0 : \mathsf{obs}(q_i) \in \mathcal{T}\}$, and $\mathsf{Parity}_G(p) = \{\pi =$

$q_0\sigma_0 q_1 \cdots \mid \min\{p(\mathsf{obs}(q)) \mid \forall i \cdot \exists j \geq i : q_j = q\}$ is even $\}$. When the game G is clear form the context, we omit the subscript in the objective name.

We are interested in deciding there exists a winning strategy for player 1 in a given game G.

The standard solution for deciding the existence of a winning strategy for player 1 in a given game G with reachability objective is to construct from G an equivalent game of perfect observation $G^{\mathcal{K}}$ obtained using a subset construction [Rei84]. Each state in $G^{\mathcal{K}}$ is a set of states of G that represents the *knowledge* of player 1, i.e. the set of states in which the game can be currently. In the worst case, the size of $G^{\mathcal{K}}$ is exponentially larger than the size of G.

Given a partial-observation game $G = \langle Q, q_\iota, \Sigma, \Delta, \mathsf{Obs}\rangle$, we define the *knowledge-based subset construction* of G as the following game structure of perfect information:

$$G^{\mathcal{K}} = \langle S, \{q_\iota\}, \Sigma, \Delta^{\mathcal{K}}\rangle,$$

where $S = \{s \in 2^Q \backslash \{\varnothing\} \mid \exists o \in \mathsf{Obs} : s \subseteq o\}$, and $(s_1, \sigma, s_2) \in \Delta^{\mathcal{K}}$ iff there exists an observation $o \in \mathsf{Obs}$ such that $s_2 = \mathsf{Post}^G_\sigma(s_1) \cap o$ and $s_2 \neq \varnothing$. Notice that for all $s \in S$ and all $\sigma \in \Sigma$, there exists a set $s' \in S$ such that $(s, \sigma, s') \in \Delta^{\mathcal{K}}$.

Given a reachability objective φ defined on G by a set $\mathcal{T}$ of target observations, we construct a reachability objective $\varphi^{\mathcal{K}}$ on $G^{\mathcal{K}}$ using the same set $\mathcal{T}$, which is well defined since every state in $s \in S$ is naturally associated to a unique observation o such that $s \subseteq o$. A result of [9] shows that player 1 has a winning strategy in a partial-observation game G with reachability objective φ if and only if player 1 has a winning strategy in the perfect-observation game $G^{\mathcal{K}}$ with the reachability objective $\varphi^{\mathcal{K}}$.

The techniques developed in this chapter can be extended from reachability analysis in simple transition systems to reachability in games (i.e., alternating reachability) using alternating simulation relations instead of classical simulations, and the fixpoint algorithms for reachability and repeated reachability can also be generalized to the parity condition [9]. A symbolic solution for partial-observation game with parity objective can be obtained by evaluating a μ-calculus formula over $G^{\mathcal{K}}$ using antichains as a symbolic representation of $\subseteq$-downward-closed sets, using the following lemma.

Lemma 2.24 $\subseteq$ *is an alternating forward simulation in* $G^{\mathcal{K}}$ *compatible with all priority sets* $\{s \in S \mid p(o) = k$ *where* o *is the unique observation such that* $s \subseteq o\}$ *(for* $k \in \mathbb{N}$*).*

We have chosen the simplest definition of partial-observation games, with the assumptions that the observations form a partition of the state space, that the objective is observable, and that only player 1 has partial observation. These assumptions can be dropped without drastically changing the results and techniques presented here. Overlapping observations are addressed in [9], non-observable parity objectives are addressed in [17], and partial observation for player 2 has no effect on the existence of winning strategies for player 1 [9].

2.4.3 QBF - Quantified Boolean Formulas

In verification and automata theory, a typical PSPACE-complete problem is the universality problem for nondeterministic finite automata (see Section 2.4.1.1). Since antichain algorithms are really efficient in practice on this problem (Figure 2.1), it is natural to explore analogous ideas for the problem of evaluating the truth value of a quantified Boolean formula (QBF), one of the most popular PSPACE-complete problems.

We present below the main ingredients to obtain antichain algorithms by exploiting the underlying structure of QBF to define *subsumption relations*, yielding compact symbolic representations, as well as sound pruning of the search space. Algorithms and preliminary performance evaluation can be found in [12].

Notations and QBF problem Let $V = \{x_1, x_2, \ldots, x_m\}$ be a set of m Boolean variables, we use $X, X_1, X_2, \ldots$ to denote subsets of V. A *literal* ℓ is either a variable $x \in V$ or the negation $\bar{x}$ of a variable $x \in V$, and a *clause* c is a disjunction of literals, or equivalently a set of literals. We use notations such as $\ell \in c$, $\bar{x} \in c$, etc. A CNF formula is a conjunction of clauses, or equivalently a set of clauses. The empty CNF formula is denoted by 1, and the empty clause by 0. In the figures we use the notations $\varnothing$ and $\perp$ instead of 1 and 0 respectively. Given a set $X \subseteq V$ and a CNF formula ψ over V, we denote by $\pi_X(\psi)$ the projection of ψ over X, with $\pi_X(\psi) = \{\pi_X(c) \mid c \in \psi\}$ and $\pi_X(c) = \{l \in c \mid (l = x \vee l = \overline{x})\text{such that } x \in X\}$.

A *quantified Boolean formula* (QBF) is an expression $\mathsf{Q}_1 X_1 \cdot \mathsf{Q}_2 X_2 \cdots \mathsf{Q}_n X_n \cdot \psi$ where each $\mathsf{Q}_i \in \{\exists, \forall\}$ for $1 \le i \le n$, the sets $X_1, \ldots, X_n$ (called *blocks*) form a partition of V, and ψ is a CNF formula over V. We also write $\mathsf{Q}_1 x_1 \cdot \mathsf{Q}_2 x_2 \cdots \mathsf{Q}_n x_n \cdot \psi$ when each block X_i contains one variable ($X_i = \{x_i\}$). Since ψ is in CNF we assume w.l.o.g. that the last block is *existential* (i.e., $\mathsf{Q}_n = \exists$). The truth value of a QBF formula is defined as usual. The QBF *evaluation problem* is to decide whether a given QBF formula is true or false. This problem is PSPACE-complete [Pap94].

A *valuation* for $X \subseteq V$ is a function $v : X \to \{0, 1\}$. The domain of v is $\mathsf{dom}(v) = X$. If $X = \{x_1, \ldots, x_k\}$, a valuation $v : X \to \{0, 1\}$ can be identified with a word $a_1 a_2 \cdots a_k \in \{0, 1\}^{|X|}$ such that $a_l = v(x_l)$ for all $1 \le l \le k$. The empty word ϵ corresponds to $\mathsf{dom}(v) = \varnothing$. Given a partition $P = X_1 \cup X_2 \cup \cdots \cup X_n$ of V, let $X_{\le i}$ be the set of variables $X_1 \cup X_2 \cdots \cup X_i$ (with $X_{\le 0} = \varnothing$), and let $X_{\ge i} = V \setminus X_{\le i-1}$. Given the valuations $v : X_{\le i-1} \to \{0, 1\}$ and $w : X_i \to \{0, 1\}$, let vw be the valuation identified with the concatenation of the words representing v and w.

A clause c is *satisfied* by a valuation v (written $v \models c$) if there exists $x \in \mathsf{dom}(v)$ such that either $x \in c$ and $v(x) = 1$, or $\bar{x} \in c$ and $v(x) = 0$. Given a CNF formula ψ, we denote by $\mathsf{sat}_v(\psi)$ the set of clauses $c \in \psi$ such that $v \models c$. We denote by $\psi[v]$ the CNF formula obtained by replacing in ψ each variable $x \in \mathsf{dom}(v)$ by its value $v(x)$. Formula $\psi[v]$ is supposed to be simplified using the laws $c \vee 1 = 1$, $c \vee 0 = c$ with c

being a clause, and $\varphi \wedge 1 = \varphi$, $\varphi \wedge 0 = 0$ with φ being a CNF formula.

Let ψ be an unsatisfiable CNF formula. An *unsatisfiable core* ψ' of ψ is any subset of clauses of ψ, minimal for the inclusion, such that ψ' is still unsatisfiable.

QBF problem as a game It is classical to view the QBF evaluation problem as reachability in an And-Or graph, or equivalently as a two-player reachability game [Pap94]. For the formula $f = \mathsf{Q}_1 x_1 \cdot \mathsf{Q}_2 x_2 \cdots \mathsf{Q}_m x_m \cdot \psi$ over $V = \{x_1, x_2, \dots, x_m\}$, the game is played in m rounds (numbered $1, \dots, m$) by the existential player $\mathsf{P}_\exists$ and the universal player $\mathsf{P}_\forall$. In round i, the truth value of the variable x_i is chosen by player $\mathsf{P}_{\mathsf{Q}_i}$. After m rounds, the players have constructed a valuation $v : V \to \{0, 1\}$, and player $\mathsf{P}_\exists$ wins if $\psi[v] = 1$ (all clauses are satisfied by v), otherwise player $\mathsf{P}_\forall$ wins. It is easy to see that $\mathsf{P}_\exists$ has a winning strategy in this game iff the formula f is true. Note that instead of having one round for each variable, we can also consider a game with one round for each block of variables, such that the blocks correspond to quantifier alternations in f. The players then choose a valuation for all the variables in the block at once, and the number of rounds is equal to the number of quantifier alternations. As the algorithms proposed in this paper are based on this game metaphor, we present the And-Or graph on which the game is played.

Let $P = X_1 \cup X_2 \cup \cdots \cup X_n$ be a partition of $V = \{x_1, x_2, \dots, x_m\}$, and let $f = \mathsf{Q}_1 X_1 \cdot \mathsf{Q}_2 X_2 \cdots \mathsf{Q}_n X_n \cdot \psi$ be a QBF formula over V. We define the *And-Or graph* $G_f = (S, S_\exists, S_\forall, s_0, E, F)$ where:

- $S = \{\psi[v] \mid \mathsf{dom}(v) = X_{\leq i-1},\ \text{for } i, 1 \leq i \leq n+1\}$;
- $S_\exists = \{\psi[v] \mid \mathsf{dom}(v) = X_{\leq i-1} \wedge \mathsf{Q}_i = \exists,\ \text{for } i, 1 \leq i \leq n\}$ is the set of $\mathsf{P}_\exists$ nodes;
- $S_\forall = \{\psi[v] \mid \mathsf{dom}(v) = X_{\leq i-1} \wedge \mathsf{Q}_i = \forall,\ \text{for } i, 1 \leq i \leq n\}$ is the set of $\mathsf{P}_\forall$ nodes;
- $s_0 = \psi$ is the initial node;
- $E = \{(\psi[v], \psi[vw]) \mid \mathsf{dom}(v) = X_{\leq i-1} \wedge \mathsf{dom}(w) = X_i,\ \text{for } i, 1 \leq i \leq n\}$ is the set of edges;
- $F = \{\psi[v] \in S \mid \psi[v] = 1\}$ is the set of final nodes.

The set S is naturally partitioned into *levels* as follows: $S = \mathsf{Level}_1 \cup \mathsf{Level}_2 \cup \cdots \cup \mathsf{Level}_{n+1}$ where $\mathsf{Level}_i = \{\psi[v] \mid \mathsf{dom}(v) = X_{\leq i-1}\}$ for each $1 \leq i \leq n+1$. The objective of player $\mathsf{P}_\exists$ is to reach the set F of nodes $\psi[v]$ such that all clauses of ψ are satisfied by v. The game starts in node s_0 and player P_Q ($\mathsf{Q} \in \{\exists, \forall\}$) chooses the successor of node s if $s \in S_\mathsf{Q}$. Thus if $s = \psi[v] \in S_\exists$ and $\mathsf{dom}(v) = X_{\leq i-1}$, then player $\mathsf{P}_\exists$ chooses one of the $2^{|X_i|}$ possible successors of s in E, corresponding to a valuation $w : X_i \to \{0, 1\}$. A node s is *winning* for player $\mathsf{P}_\exists$ if he has a strategy to force reaching a node in F from s, no matter the choices of $\mathsf{P}_\forall$; otherwise it is *losing*. We denote by W the set of winning nodes for player $\mathsf{P}_\exists$, and by $L = S \setminus W$ the set of losing nodes for $\mathsf{P}_\exists$. We say that $\mathsf{P}_\exists$ is *winning the game* if $s_0 \in W$. In the sequel, we use the notations W_i (resp., L_i) to denote $W \cap \mathsf{Level}_i$ (resp., $L \cap \mathsf{Level}_i$).

Lemma 2.25 *A QBF formula f is true iff player $\mathsf{P}_\exists$ is winning the game G_f.*

Note that in the graph G_f, each node $\psi[v]$ with $\mathsf{dom}(v) = X_{\leq i-1}$ can be associated with the formula $\mathsf{Formula}(\psi[v]) \equiv \mathsf{Q}_i X_i \cdots \mathsf{Q}_n X_n \cdot \psi[v]$, and we can strengthen the previous lemma as follows.

Lemma 2.26 *Given a QBF formula f, the set of winning nodes in the graph G_f is $W = \{\psi[v] \in S \mid \mathsf{Formula}(\psi[v])$ is true$\}$, and the set of losing nodes is $L = \{\psi[v] \in S \mid \mathsf{Formula}(\psi[v])$ is false$\}$.*

Structure in the And-Or graph and antichains The antichain algorithm to solve the game played on G_f exploits the following *subsumption* relation on QBF formulas. We write $f_1 \sqsubseteq f_2$ if $f_1 = \mathsf{Q}_i X_i \cdots \mathsf{Q}_n X_n \cdot \psi_1$ and $f_2 = \mathsf{Q}_i X_i \cdots \mathsf{Q}_n X_n \cdot \psi_2$ are two QBF formulas with the same quantifier prefix, and $\psi_1 \subseteq \psi_2$. Intuitively, f_1 is more promising than f_2 for player $\mathsf{P}_\exists$ because all strategies that are winning from ψ_2 are also winning from ψ_1.

Lemma 2.27 *Suppose that $f_1 \sqsubseteq f_2$. If f_2 is true, then f_1 is true; and if f_1 is false, then f_2 is false.*

As a direct consequence of Lemmas 2.26 and 2.27, we obtain the next corollary.

Corollary 2.28 *In the graph G_f, for all nodes $s_1, s_2 \in \mathsf{Level}_i$ such that $s_1 \subseteq s_2$, i.e. $\mathsf{Formula}(s_1) \sqsubseteq \mathsf{Formula}(s_2)$, if $s_2 \in W_i$, then $s_1 \in W_i$; and if $s_1 \in L_i$, then $s_2 \in L_i$.*

Hence, W_i is $\subseteq$-downward closed and L_i is $\subseteq$-upward closed. The set of $\subseteq$-maximal elements of W_i, noted $\lceil W_i \rceil$, is an *antichain* for the partial order $\subseteq$ that canonically and compactly represents W_i. Similarly, the set of $\subseteq$-minimal elements of L_i, noted $\lfloor L_i \rfloor$, is an antichain that canonically and compactly represents L_i. A prototype implementation of an antichain algorithm based on such compact representations is presented in [12]. Although the finely tuned state-of-the-art QBF solvers [GNT04, LB10, NPPT09] remain faster on most examples, the preliminary results are promising and show the feasibility of the approach.

2.5 Tools

Since their introduction in 2006, the antichain algorithms we have presented have been implemented in the tools Alaska [45] for applications in logics and automata, and Alpaga [44] for applications in partial-observation games. We briefly present these tools below. The antichain approach has been further developed by other teams for LTL synthesis (Acacia [FJR09, FJR10], Unbeast [Ehl10]) and automata-based verification [BHH+08, ACH+10, FV10].

- Our tool Alaska implements antichain algorithms to solve the emptiness problem for alternating finite automata (AFA) and alternating Büchi automata (ABW). Using the well-known translation from LTL to alternating automata, Alaska solves the satisfiability, validity, and model-checking problems for LTL over finite and infinite words.

  While several tools (notably NuSMV [CCGR00], and Spin [RH04]) have addressed the satisfiability and model-checking problems for LTL [RV07], Alaska uses new antichain algorithms that are often more efficient, especially over large LTL formulas. Moreover, to the best of our knowledge, Alaska was the first publicly available tool that provides a direct interface to efficient algorithms for the emptiness of ABW and AFA. Alaska is intended as an open and documented library of antichain-based verification algorithms.

  Promising experimental results have been obtained by Alaska as illustrated by the example of Section 2.4.1.3. Detailed performance evaluation and comparison can be found in [34, 45].

- Our tool Alpaga solves parity games with partial observation. Given the description of a game, the tool determines whether the partial-observation player has a winning strategy for the parity objective and, if this is the case, it constructs such a winning strategy.

  Alpaga implements antichain algorithms based on symbolic fixed-point computations using a compact representation of sets [9], and compact representation of strategies [31]. The symbolic algorithm evaluates a μ-calculus fixpoint formula defined using the *controllable predecessor* operator that returns the states from which a player can force the play into a given target set in one round. As no polynomial algorithms is known for computing this operator, we propose a new symbolic implementation based on BDDs to avoid a naive enumerative procedure.

  To the best of our knowledge, this is the first implementation of a tool for solving parity games with partial observation. Details and performance evaluation can be found in [31, 44].

2.6 Perspectives

Currently, we are exploring the following applications of antichain algorithms in verification:

- with Marc Ducobu et al. [12], we continue the development of a solver for QBF, based on the game interpretation presented in Section 2.4.3. Subsumption is a new and simple idea that gives promising results, and should further be evaluated in QBF solving.
- we are exploring antichain algorithms for solving parity games, using a reduction to safety games [1]. The reduction is exponential, but it induces a partial order on the state space that can be used in principle to define antichain algorithms.
- a quivering research direction is the practical algorithmic solution of partial-observation games. We have presented tools and algorithms for solving partial-observation games with parity objectives (Section 2.4.2). What is missing is the efficient algorithmic solution of partial-observation games with stochastic transitions, randomized strategies, and partial observation for both players.

Chapter 3

Quantitative Games

Il n'y a que deux manières de gagner la partie: jouer cœur ou tricher.

Jean Cocteau, *Lettre à Jacques Maritain*.

3.1 Introduction

In classical model-checking, the aim is to formally verify that an implementation of a system (circuit, protocol, embedded software, etc.) satisfies a given specification. Fully automatic and exhaustive verification is typically undecidable, and therefore the formal analysis typically considers simplified model of the system (*abstraction*) for which certain properties can be expressed and are decidable (deadlock avoidance, reactivity, fairness, etc.).

We consider extensions of the classical model-checking framework, in two directions. First, all or some part of the implementation may not be given (such as synchronization policy in lock-based concurrent systems, or termination condition of a loop, etc.) and then the formal analysis should be aimed at synthesizing a correct implementation. Synthesis is naturally viewed as a *game*, the interactive generalization of automata where one player (the implementation) tries to satisfy an objective (the specification) no matter the behavior of the second player (the external uncontrollable environment). Second, the correctness of an implementation may not be formulated as a purely Yes/No question, but as a measure of its quality (such as response time, or energy consumption), and the properties of interest may not be expressed as classical qualitative properties (such as standard trace inclusion in ω-regular specifications). *Quantitative* notions of correctness, and quantitative properties generalize the classical theory of games and automata [Hen10].

In this chapter, we study *quantitative games* where the quantitative aspect is a natural model for the amount of resource (such as energy usage, or buffer size) used by an implementation, or for the quality measure or distance to correctness of a model.

Quantitative games for synthesis have been studied in several previous works (see e.g. [CdHS03, CdF+06, BCHJ09]). We consider the model of *energy games* [CdHS03, BFL+08] where the value of a run of the system is the sum of the weights along the run, reminiscent of the *energy* consumption (and recharging) of an embedded system. Energy games have been relatively little studied, as compared to mean-payoff and discounted games where the value of a run is the long-run average of the weights, or the discounted sum of the weights [EM79, ZP96]. Mean-payoff and discounted games have fascinating connections with parity games and simple stochastic games: parity games (and the model-checking problem for the modal mu-calculus [Koz83]) reduce to mean-payoff games [GH82, EJS93], which reduce to discounted games [ZP96], themselves reducing to simple stochastic games [Con92, ZP96]. All reductions are in polynomial time, and all these problems are in NP $\cap$ coNP. It is a long-standing open question to know whether these problems are in P.

It follows from the results of [BFL+08] that energy games are surprisingly log-space equivalent to mean-payoff games. Since energy games are conceptually simpler than mean-payoff games, their structure and algorithmic solution provide new insights on problems related to classical games (mainly mean-payoff games) as we show in this chapter.

In an energy game, given an initial credit $c \in \mathbb{N}$, the objective of player 1 is to maintain the sum of the weights (the energy level) always positive. The decision problem for energy games asks, given a weighted game graph and state q, if there exists an initial credit for which player 1 wins from q. It is known that memoryless strategies are sufficient for energy games, and that player 1 essentially needs to ensure that all cycles that can be formed by player 2 have nonnegative weight.

First, we present a direct fixpoint algorithm to compute the minimum initial credit when it exists. In combination with the log-space equivalence with mean-payoff games, it improves on the algorithmic solutions to solve mean-payoff games.

Second, we consider mean-payoff and energy games with *partial observation*. In applications of games for controller synthesis, the players may not have a complete knowledge of the history of the execution (e.g., because the private variables of other players are not visible, or because the sensors have poor accuracy). The log-space equivalence between mean-payoff and energy games breaks in the setting of partial-observation games, and we show that winning strategies may require infinite memory for mean-payoff games, while finite memory is sufficient for energy games, in contrast with games of perfect observation where memoryless strategies suffice in both cases. We show that under partial observation, energy games with fixed initial credit are decidable, while both energy games with unknown initial credit, and mean-payoff games are undecidable.

Third, we consider *energy parity games* as a model for the design of reactive systems that work in resource-constrained environment. The desired system must respect both a quantitative specification (e.g., constraining resource usage such as power consumption) and a qualitative specification (constraining the functional requirement). Only recently objectives combining both qualitative and quantitative specifications have been considered [CdHS03, CHJ05, BCHJ09]. The qualitative specification is a parity condition, a canonical way to express the ω-regular objectives [Tho97], and the quantitative specification is the energy condition The main algorithmic question is to decide if there exists an initial credit (or initial energy level) such that one player has a strategy to maintain the level of energy positive while satisfying the parity condition.

Energy parity games generalize both parity games and energy games. We obtain the following results: (a) exponential-memory strategies are sufficient to win energy parity games, and exponential memory may be required. Strategies of the opponent need no memory at all (memoryless spoiling strategies exist); (b) while memoryless strategies are not sufficient, we show that the problem of deciding the existence of an initial credit which is sufficient to win an energy parity game is (perhaps surprisingly) in NP $\cap$ coNP; (c) a conceptually simple algorithmic solution to compute the minimum initial credit in energy parity games, and (d) the polynomial equivalence with mean-payoff parity games, which provides as a corollary that the problem of deciding the winner in mean-payoff parity games is also in NP $\cap$ coNP, and a new algorithm for solving mean-payoff parity games with essentially the same complexity as in [CHJ05], but with a conceptually simpler solution.

Finally, we study a multi-dimensional generalization of energy games as a model of reactive systems with several resources. The main result shows that the decision problem of the existence of an initial credit vector is coNP-complete. In multi-dimensional games, the log-space equivalence with mean-payoff games breaks as well.

3.2 Definitions

Game graphs. A *game graph* $G = \langle Q, E\rangle$ consists of a finite set Q of states partitioned into player-1 states Q_1 and player-2 states Q_2 (i.e., $Q = Q_1 \cup Q_2$), and a set $E \subseteq Q \times Q$ of edges such that for all $q \in Q$, there exists (at least one) $q' \in Q$ such that $(q, q') \in E$. A *player*-1 *game* is a game graph where $Q_1 = Q$ and $Q_2 = \varnothing$. The subgraph of G induced by $S \subseteq Q$ is the graph $G \upharpoonright S = \langle S, E \cap (S \times S)\rangle$ (which is not a game graph in general); the subgraph induced by S is a game graph if for all $s \in S$ there exist $s' \in S$ such that $(s, s') \in E$.

Given a set of states $U \subseteq Q$, we denote by $\mathsf{pre}(U)$ the set of states having a successor in U, i.e. $\mathsf{pre}(U) = \{q \mid \exists q' \in U : (q, q') \in E\}$, and by $\mathsf{post}(U)$ the set of successors of states in U, i.e. $\mathsf{post}(U) = \{q' \mid \exists q \in U : (q, q') \in E\}$. For singleton $U = \{q\}$ we write $\mathsf{pre}(q)$ and $\mathsf{post}(q)$.

Plays and strategies. A game on G starting from a state $q_0 \in Q$ is played in rounds as follows. If the game is in a player-1 state, then player 1 chooses the successor state from the set of outgoing edges; otherwise the game is in a player-2 state, and player 2 chooses the successor state. The game results in a *play* from q_0, i.e., an infinite path $\rho = q_0 q_1 \dots$ such that $(q_i, q_{i+1}) \in E$ for all $i \geq 0$. The prefix of length n of ρ is denoted by $\rho(n) = q_0 \dots q_n$. A *strategy* for player 1 is a function $\sigma : Q^* Q_1 \to Q$ such that $(q, \sigma(\rho \cdot q)) \in E$ for all $\rho \in Q^*$ and $q \in Q_1$. An *outcome* of σ from q_0 is a play $q_0 q_1 \dots$ such that $\sigma(q_0 \dots q_i) = q_{i+1}$ for all $i \geq 0$ such that $q_i \in Q_1$. Strategy and outcome for player 2 are defined analogously.

Finite-memory strategies. A strategy uses *finite-memory* if it can be encoded by a deterministic transducer $\langle M, m_0, \alpha_u, \alpha_n \rangle$ where M is a finite set (the memory of the strategy), $m_0 \in M$ is the initial memory value, $\alpha_u : M \times Q \to M$ is an update function, and $\alpha_n : M \times Q_1 \to Q$ is a next-move function. The *size* of the strategy is the number $|M|$ of memory values. If the game is in a player-1 state q and m is the current memory value, then the strategy chooses $q' = \alpha_n(m, q)$ as the next state and the memory is updated to $\alpha_u(m, q)$. Formally, $\langle M, m_0, \alpha_u, \alpha_n \rangle$ defines the strategy α such that $\alpha(\rho \cdot q) = \alpha_n(\hat{\alpha}_u(m_0, \rho), q)$ for all $\rho \in Q^*$ and $q \in Q_1$, where $\hat{\alpha}_u$ extends α_u to sequences of states as expected. A strategy is *memoryless* if $|M| = 1$. For a finite-memory strategy σ, let G_σ be the graph obtained as the product of G with the transducer defining σ, where $(\langle m, q \rangle, \langle m', q' \rangle)$ is a transition in G_σ if $m' = \alpha_u(m, q)$ and either $q \in Q_1$ and $q' = \alpha_n(m, q)$, or $q \in Q_2$ and $(q, q') \in E$. In G_σ, the expression *reachable from* q stands for *reachable from* $\langle q, m_0 \rangle$.

Objectives. An *objective* for G is a set $\varphi \subseteq Q^\omega$. Let $p : Q \to \mathbb{N}$ be a *priority function* and $w : E \to \mathbb{Z}$ be a *weight function*[1] where positive numbers represent rewards. We denote by W the largest weight (in absolute value) according to w. The *energy level* of a prefix $\gamma = q_0 q_1 \dots q_n$ of a play is $\mathsf{EL}(w, \gamma) = \sum_{i=0}^{n-1} w(q_i, q_{i+1})$, and the *mean-payoff value* of a play $\rho = q_0 q_1 \dots$ is is either $\overline{\mathsf{MP}}(w, \pi) = \limsup_{n \to \infty} \frac{1}{n} \cdot \mathsf{EL}(w, \pi(n))$ or $\underline{\mathsf{MP}}(w, \pi) = \liminf_{n \to \infty} \frac{1}{n} \cdot \mathsf{EL}(w, \pi(n))$.

In the sequel, when the weight function w is clear from context we will omit it and simply write $\mathsf{EL}(\gamma)$ and $\mathsf{MP}(\rho)$. We denote by $\mathsf{Inf}(\rho)$ the set of states that occur infinitely often in ρ. We consider the following objectives:

- *Safety and reachability objectives.* Given a set $\mathcal{T} \subseteq Q$ of target states, the *reachability objective* requires that the play visit the set $\mathcal{T}$: $\mathsf{Reach}(\mathcal{T}) = \{\rho = q_0 q_1 \dots \in \mathsf{Plays}(G) \mid \exists i \geq 0 : q_i \in \mathcal{T}\}$, the dual *safety objective* requires the play to stay within the set $\mathcal{T}$: $\mathsf{Safe}(\mathcal{T}) = \{\rho = q_0 q_1 \dots \in \mathsf{Plays}(G) \mid \forall i \geq 0 : q_i \in \mathcal{T}\}$.

- *Parity objectives.* The *parity* objective $\mathsf{Parity}_G(p) = \{\rho \in \mathsf{Plays}(G) \mid \min\{p(q) \mid q \in \mathsf{Inf}(\rho)\}$ is even $\}$ requires that the minimum priority visited infinitely often be even. The special cases of *Büchi* and *coBüchi* objectives correspond to the case with two priorities, $p : Q \to \{0, 1\}$ and $p : Q \to \{1, 2\}$ respectively.

[1] Sometimes we take the freedom to use rational weights (i.e., $w : E \to \mathbb{Q}$), while we always assume that weights are integers encoded in binary for complexity results.

- *Energy objectives.* Given an initial credit $c_0 \in \mathbb{N} \cup \{\infty\}$, the *energy* objective $\mathsf{PosEnergy}_G(c_0) = \{\rho \in \mathsf{Plays}(G) \mid \forall n \geq 0 : c_0 + \mathsf{EL}(\rho(n)) \geq 0\}$ requires that the energy level be always positive.

- *Mean-payoff objectives.* Given a threshold $\nu \in \mathbb{Q}$, the *mean-payoff* objective $\mathsf{MeanPayoff}_G(\nu) = \{\rho \in \mathsf{Plays}(G) \mid \underline{\mathsf{MP}}(\rho) \geq \nu\}$ requires that the mean-payoff value be at least ν. In Section 3.4, we distinguish between objectives defined by $\underline{\mathsf{MP}}(\cdot)$ and $\overline{\mathsf{MP}}(\cdot)$ and with strict or non-strict threshold.

- *Combined objectives.* The *energy parity* objective $\mathsf{Parity}_G(p) \cap \mathsf{PosEnergy}_G(c_0)$ and the *mean-payoff parity* objective $\mathsf{Parity}_G(p) \cap \mathsf{MeanPayoff}_G(\nu)$ combine the requirements of parity and energy (resp., mean-payoff) objectives.

When the game G is clear form the context, we omit the subscript in objective names.

Winning strategies. A player-1 strategy σ is *winning*[2] in a state q for an objective φ if $\rho \in \varphi$ for all outcomes ρ of σ from q. For energy and energy parity objectives with unspecified initial credit, we also say that a strategy is winning if it is winning for some finite initial credit.

Decision problems. We are interested in the following decision problems.

The *initial credit problem* asks, given an energy (parity) game $\langle G, p, w\rangle$ and a state q, whether there exists a finite initial credit $c_0 \in \mathbb{N}$ and a winning strategy for player 1 from q for the objective $\mathsf{PosEnergy}_G(c_0)$ (resp., $\mathsf{Parity}_G(p) \cap \mathsf{PosEnergy}_G(c_0)$). Given a fixed initial credit $c_0 \in \mathbb{N}$, the *fixed initial credit problem* asks whether there exists a winning strategy for player 1 from q for the objective $\mathsf{PosEnergy}_G(c_0)$.

The *minimum initial credit* in a state $q_0 \in Q$ is the least value of initial credit for which there exists a winning strategy for player 1 in q_0.

The *mean-payoff threshold problem* asks, given a mean-payoff (parity) game $\langle G, p, w\rangle$, a state q, and a threshold $\nu \in \mathbb{Q}$, to decide whether there exists a winning strategy σ for player 1 from q for the objective $\mathsf{MeanPayoff}_G(\nu)$ (resp., $\mathsf{Parity}_G(p) \cap \mathsf{MeanPayoff}_G(\nu)$). It is sufficient to consider this problem with threshold 0 because the answer for the instance $(\langle G, p, w\rangle,\ q,\ \nu)$ is the same as the answer for $(\langle G, p, w - \nu\rangle,\ q,\ 0)$ where $(w - \nu)(e) = w(e) - \nu$ for all edges $e \in E$.

The *optimal mean-payoff value* in a state $q_0 \in Q$ is the largest threshold for which there exists a winning strategy for player 1 in q_0.

A strategy for player 1 is *optimal* in a state q_0 if it is winning from q_0 with the minimum initial credit (resp., the optimal mean-payoff value). The related problem of *strategy synthesis*, which is to construct an optimal strategy, is discussed in [2].

[2] We also say that player-1 is winning, or that q is a winning state.

		Memory player 1	Memory player 2
perfect observation	Energy	memoryless [CdHS03, BFL+08]	
	Mean-Payoff	memoryless [EM79]	
partial observation	Energy	finite	counting
	Mean-Payoff	infinite	counting
perfect observation	Energy parity	$\lvert Q\rvert \cdot d \cdot W$	memoryless
	Mean-Payoff parity	infinite [CHJ05]	memoryless [CHJ05]
perfect obs.	Multi-energy	finite [BJK10]	memoryless [BJK10]

Table 3.1: Strategy complexity in energy and mean-payoff (parity) games.

		Computational complexity	Algorithmic complexity
perfect observation	Energy	NP ∩ coNP [CdHS03, BFL+08]	$O(\lvert E\rvert \cdot \lvert Q\rvert \cdot W)$
	Mean-Payoff	NP ∩ coNP [EM79]	$O(\lvert E\rvert \cdot \lvert Q\rvert \cdot W)$
partial observation	Energy	undecidable	
	Mean-Payoff	undecidable	
perfect observation	Energy parity	NP ∩ coNP	$O(\lvert E\rvert \cdot d \cdot \lvert Q\rvert^{d+2} \cdot W)$
	Mean-Payoff parity	NP ∩ coNP	$O(\lvert E\rvert \cdot d \cdot \lvert Q\rvert^{d+3} \cdot W)$
perfect obs.	Multi-energy	coNP-complete	

Table 3.2: Computational and algorithmic complexity of energy and mean-payoff (parity) games.

3.3 Energy and Mean-Payoff Games

We consider the decision problems for energy and mean-payoff games with perfect observation.

It is known from the works of Ehrenfeucht and Mycielski that memoryless optimal strategies exist for mean-payoff games [EM79]. From this result, it is easy to show that the decision problem for mean-payoff games lies in NP ∩ coNP, and that either definitions of mean-payoff (defined as lim sup or lim inf) are equivalent for perfect-observation games. Despite large research efforts [GKK88, ZP96, Pis99, Con93, DG06, LP07, BV07], no polynomial time algorithm is known for that problem. A pseudo-polynomial time algorithm has been proposed by Zwick and Paterson [ZP96] with complexity $O(|E| \cdot |Q|^2 \cdot W)$.

For energy games, it is known that the initial credit problem can be solved in NP ∩ coNP because memoryless strategies are sufficient to win such games [CdHS03, BFL+08]. The following characterizations establish the log-space equivalence of the decision problems for energy and mean-payoff games.

Lemma 3.1 ([EM79, BV07]) *A memoryless strategy σ for player 1 is winning from a state q in a mean-payoff game G with threshold ν if and only if all cycles reachable from q in G_σ have average weight at least ν.*

Lemma 3.2 ([EM79, LP07, BFL+08]) *A memoryless strategy σ for player 1 is winning from a state q in an energy game G if and only if all cycles reachable from q in G_σ are nonnegative.*

Corollary 3.3 ([BFL+08]) *Energy games are log-space equivalent to mean-payoff games. Given a game G, a state q, a weight function w, and a threshold ν, a strategy σ for player 1 is winning in the mean-payoff game (G, w) with threshold ν if and only if σ is winning in the energy game $(G, w - \nu)$.*

Once a memoryless winning strategy is fixed, the minimum initial credit in a winning state q can be computed as follows: for each state q', let $c(q')$ be the weight of the cheapest path from q to q'. By Lemma 3.2, the cheapest paths are acyclic and thus their weight is at most $\mathcal{M} = |V| \cdot W$. The minimum initial credit in q is $\max_{q'} c(q')$ and this value is at most $\mathcal{M}$ (if it is finite).

The algorithmic solution of energy game consists in computing the minimum initial credit $f(q)$ in each state q by a fixpoint iteration which computes successive (under)approximations of the minimum initial credit. Initially, let $f_0(q) = 0$ for all states $q \in Q$. Then, we update the function f_0 in each state q using the operator $\mathsf{Lift}(\cdot)$ defined by:

$$\mathsf{Lift}(f)(q) = \begin{cases} \max\{0, \min\{f(q') - w(q,q') \mid (q,q') \in E\}\} & \text{if } q \in Q_0 \\ \max\{0, \max\{f(q') - w(q,q') \mid (q,q') \in E\}\} & \text{if } q \in Q_1 \end{cases}$$

Intuitively, the value $f(q') - w(q,q')$ is the energy level needed in state q in order to take the edge (q,q') and ensure energy level at least $f(q')$ in the successor state q'. If $q \in Q_0$ is a player-0 state, then player 0 can choose the cheapest successor of q, therefore $\mathsf{Lift}(f)(q)$ computes $\min\{f(q') - w(q,q') \mid (q,q') \in E\}$; if $q \in Q_1$ is a player-1 state, then player 1 needs to survive all successors of q, therefore $\mathsf{Lift}(f)(q)$ computes $\max\{f(q') - w(q,q') \mid (q,q') \in E\}$.

In the fixpoint iterations $f_{i+1} = \mathsf{Lift}(f_i)$, whenever the value $f_i(q)$ in some state q gets larger than $\mathcal{M}$ we know by the above argument that the state q is not winning for player 0. Therefore, the values above $\mathcal{M}$ are equivalent, and we can stop the fixpoint iterations when the condition $\lceil f_i(q) \rceil_{\mathcal{M}} = \lceil f_{i+1}(q) \rceil_{\mathcal{M}}$ holds for all $q \in Q$ where:

$$\lceil k \rceil_{\mathcal{M}} = \begin{cases} k & \text{if } x \leq \mathcal{M} \\ \infty & \text{if } x \geq \mathcal{M} \end{cases}$$

This implies that the functions f_i have essentially a finite range ($\{0, \dots, \mathcal{M}\} \cup \{\infty\}$) and thus the fixpoint algorithm terminates. In a clever implementation of this algorithm, we need to update the value in a state q only when the value a successor q' of q has been updated previously. In fact, whenever the value in a state q' is updated,

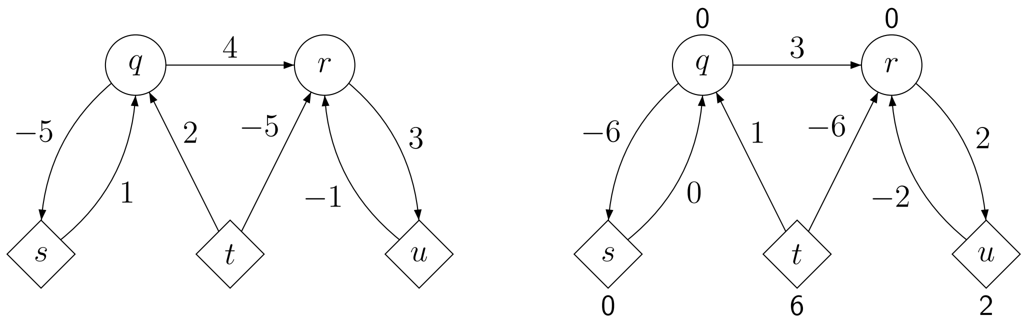


Figure 3.1: Solving mean-payoff games (on the left) using energy games (on the right).

we check which predecessors $q \in \mathsf{pre}(q')$ need to be updated and include them in a list. For $q \in Q_0$, the condition is that $f(q)+w(q,q') < \mathsf{Lift}(f)(q')$. For $q \in Q_1$, the condition is slightly more tricky because it is not sufficient that $f(q) + w(q,q') < \mathsf{Lift}(f)(q')$ as there may (or may not) be other states $q'' \in \mathsf{post}(q)$ such that $f(q) + w(q,q'') = f(q'')$. The algorithm keeps a counter for each state $q \in Q_1$ and decrement it when $f(q) + w(q,q') < \mathsf{Lift}(f)(q')$. The state q is inserted in the list when the counter is equal to 0. The implementation details [2] yield the following complexity analysis. The value of each state is updated at most $\mathcal{M} + 1$ times, and each update requires looking up the sets of predecessors and of successors, hence the time complexity is

$$\begin{aligned}
&\textstyle\sum_{q\in Q}(|\mathsf{pre}(q)| + |\mathsf{post}(q)|) \cdot (\mathcal{M} + 1) = \\
&O(|Q| \cdot W \cdot \textstyle\sum_{q\in Q}(|\mathsf{pre}(q)| + |\mathsf{post}(q)|)) \\
&= O(|E| \cdot |Q| \cdot W).
\end{aligned}$$

This improves the complexity of the algorithm proposed by Zwick and Paterson [ZP96] which has $O(|E| \cdot |Q|^2 \cdot W)$ complexity. The results of this section are summarized in the first two lines of Table 3.2.

Example 3.1 *Consider the mean-payoff game shown on the left of Figure 3.1, where round states controlled by player* 1*, and diamond states by player* 2*.*

Player 1 *can ensure a mean-payoff value* 1 *from all states by reaching the cycle* (ru)*. By Corollary 3.3, this game is equivalent to the energy game on the right of Figure 3.1 where the weights are decreasing by* 1*. Player* 1 *has a strategy to confine the play into the nonnegative cycle* (ru) *and win the energy game (the minimum initial credit is attached to each state on Figure 3.1). Therefore, the same strategy ensures mean-payoff value* 1 *in the original mean-payoff game.*

3.4 Partial-Observation Energy and Mean-Payoff Games

We recall the definition of partial-observation game from Section 2.4.2.

A *partial-observation game* is a tuple $G = \langle Q, q_\iota, \Sigma, \Delta, \mathsf{Obs}\rangle$, where Q is a finite set of states, $q_\iota \in Q$ is the initial state, Σ is a finite alphabet, $\Delta \subseteq Q \times \Sigma \times Q$ is a labeled transition relation which is total, i.e. for all $q \in Q$ and $\sigma \in \Sigma$, there exists $q' \in Q$ such that $(q, \sigma, q') \in \Delta$; and $\mathsf{Obs} \subseteq 2^Q$ is a set of *observations* that partition the state space. For each state $q \in L$, we denote by $\mathsf{obs}(q)$ the unique observation $o \in \mathsf{Obs}$ such that $q \in o$; and for $s \subseteq L$ and $\sigma \in \Sigma$, we denote by $\mathsf{post}^G_\sigma(s) = \{q' \in Q \mid \exists q \in s : (q, \sigma, q') \in \Delta\}$ the set of σ-successors of s. A game with *perfect observation* is such that $\mathsf{Obs} = \{\{q\} \mid q \in Q\}$, i.e. every state is observable. We omit the set Obs in perfect-observation games. Games with one observation (i.e., $\mathsf{Obs} = \{Q\}$) are called *blind* games.

Recall that partial-observation games are played in rounds in which player 1 chooses an action $\sigma \in \Sigma$, and player 2 chooses a σ-successor of the current state. For perfect-observation games, this model is equivalent to the model presented in Section 3.2 [1]. Related definitions of plays, prefixes, energy level, etc. are adapted from Section 2.4.2.

A *strategy* (for player 1) in a partial-observation game G is a function function $\sigma : \mathsf{Prefs}(G) \to \Sigma$ such that for all prefixes $\rho, \rho' \in \mathsf{Prefs}(G)$, if $\mathsf{obs}(\rho) = \mathsf{obs}(\rho')$, then $\alpha(\rho) = \alpha(\rho')$. We emphasize this property of strategies by calling them *observation-based.*

In the rest of this section, we consider mean-payoff objectives defined as lim sup or lim inf, and with strict or non-strict threshold, because our results crucially depend on this distinction.

Refined mean-payoff objectives. Given a threshold $\nu \in \mathbb{Q}$, and $\sim \in \{>, \geq\}$, the *mean-payoff* objectives are $\mathsf{MeanPayoffSup}^\sim(\nu) = \{\rho \in \mathsf{Plays}(G) \mid \overline{\mathsf{MP}}(\rho) \sim \nu\}$ and $\mathsf{MeanPayoffInf}^\sim(\nu) = \{\rho \in \mathsf{Plays}(G) \mid \underline{\mathsf{MP}}(\rho) \sim \nu\}$.

Example 3.2 *Consider the blind game of Figure 3.2. All states are indistinguishable (i.e., $\mathsf{Obs} = \{Q\}$), and an initial nondeterministic choice determines the state q_1 or q_1' in which the game loops forever.*

We claim that player 1 has an observation-based winning strategy for the objective $\mathsf{MeanPayoffSup}^{\geq}(0)$, but not for $\mathsf{MeanPayoffInf}^{\geq}(0)$. Note that in blind games, observation-based strategies can be viewed as infinite words. A winning strategy for the limsup version consists in playing sequences of a's and b's of increasing length in order to ensure a mean-payoff value $\overline{\mathsf{MP}}$ equal to 0 in both states q_1 and q_1'. For example, playing sequences of a's and b's such that the length of the i-th sequence is i times the length of the prefix played so far. This ensures that in q_1 and q_1', for all $i > 0$ there

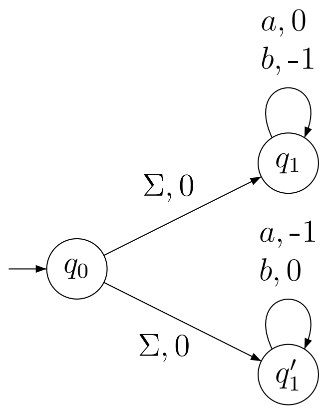


Figure 3.2: A blind mean-payoff game.

are infinitely many positions such that the average of the weights is greater than $-\frac{1}{i}$*, showing that the limsup is* 0 *in all outcomes.*

We show that for every word $w \in \{a, b\}^\omega$*, the mean-payoff value according to* $\underline{\mathsf{MP}}$ *is at most* $-\frac{1}{2}$*. Let* n_i *and* m_i *be the numbers of* a*'s and* b*'s in the prefix of length* i *of* w*. Either* $n_i \leq m_i$ *for infinitely many* i*'s, or* $n_i \geq m_i$ *for infinitely many* i*'s. In the first case, the average of the weights (in state* q_1*) is infinitely often at most* $-\frac{1}{2}$*. The same holds in the second case using state* q_1'*. Therefore the* $\liminf$ *of the weight averages is at most* $-\frac{1}{2}$*, and player* 1 *has no winning strategy for the mean-payoff objective defined using* $\liminf$ *and threshold* 0*.*

Note that infinite memory is required to achieve mean-payoff value 0 *(according to* $\overline{\mathsf{MP}}$*). Indeed, for all finite-memory strategies (which can be viewed as ultimately periodic words), the mean-payoff value of an outcome is* $\min\{-\frac{n}{n+m}, -\frac{m}{n+m}\} \leq -\frac{1}{2}$ *where* n *and* m *are the numbers of* a*'s and* b*'s in the cycle of the strategy.*

For partial-observation energy games, we show that when the initial credit is fixed the winner of the game can be decided, and finite-memory strategies are sufficient to win.

Theorem 3.4 *The fixed initial credit problem is decidable.*

It follows that the initial credit problem is r.e. (by enumerating the initial credit values). A reduction from the halting problem of 2-counter machines shows that the initial credit problem is undecidable (thus not co-r.e.).

Theorem 3.5 *The unknown initial credit problem for partial-observation energy games is undecidable, even for blind games.*

The proof of Theorem 3.4 relies an the construction of an equivalent perfect-observation game, and an argument based on well-quasi-order. The state space of the perfect-observation game is the set of functions $f : Q \to \mathbb{Z} \cup \{\perp\}$ which store as their *support* is $\mathsf{supp}(f) = \{q \in Q \mid f(q) \neq \perp\}$ the possible current states of the game G together with their worst-case energy level. Initially, if q_0 is the initial state, and c_0 is the initial credit, then we have $f_{c_0}(q_0) = c_0$ and $f_{c_0}(q) = \perp$ for all $q \neq q_0$. Functions are ordered by the relation $\preceq$ such that $f_1 \preceq f_2$ if $\mathsf{supp}(f_1) = \mathsf{supp}(f_2)$ and $f_1(q) \leq f_2(q)$ for all $q \in \mathsf{supp}(f_1)$.

For $\sigma \in \Sigma$ and $\gamma \in \mathsf{Obs}$, the (σ, γ)-successor of f_1 is the function f_2 such that $\mathsf{supp}(f_2) = \mathsf{post}^G_\sigma(\mathsf{supp}(f_1)) \cap \gamma$ and $f_2(q) = \min\{f_1(q') + w(q', \sigma, q) \mid q' \in \mathsf{supp}(f_1) \wedge (q', \sigma, q) \in \Delta\}$ for all $q \in \mathsf{supp}(f_2)$.

The perfect-information game has the form of an unraveling tree in which the leaves are the nodes with a $\preceq$-smaller ancestor. The game has a safety objective defined by the target states f such that $f(q) \geq 0$ for all $q \in Q$. Over the target states, the relation $\preceq$ is a *well-quasi-order*, and by König's Lemma [Kön36]) and Dickson's lemma [Dic13] this tree is finite, yielding decidability as well as existence of finite-memory winning strategies.

Corollary 3.6 *Finite-memory strategies are sufficient to win partial-observation energy games.*

Technical adaptations of the proof of Theorem 3.5 give the undecidability results for partial-observation mean-payoff games. However, the results do not hold for all mean-payoff objectives, and the proof that the problem is not r.e. requires two observations in the game, and it crucially relies on using non-strict inequality in the objective and on the mean-payoff value being defined using $\limsup$. The cases of $\liminf$ and blind games remain open.

Theorem 3.7 *The threshold problem for partial-observation mean-payoff games and objective* $\mathsf{MeanPayoffSup}^{>}(0)$ *(or* $\mathsf{MeanPayoffInf}^{>}(0)$*) is undecidable (it is not co-r.e.), even for blind games.*

Theorem 3.8 *The threshold problem for partial-observation mean-payoff games and objective* $\mathsf{MeanPayoffSup}^{\geq}(0)$ *is undecidable (it is not r.e.).*

It follows from Example 3.2 that infinite memory is required in general to win partial-observation mean-payoff games.

Corollary 3.9 *Finite-memory strategies are not sufficient in general to win partial-observation mean-payoff games.*

The results on partial-observation energy and mean-payoff games are summarized in Table 3.1 Table 3.2.

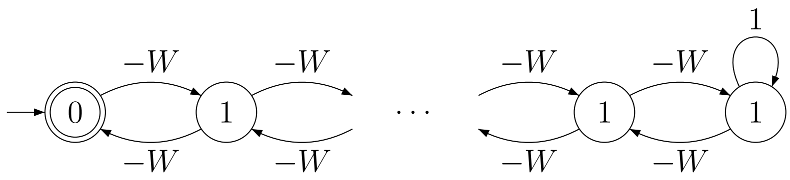


Figure 3.3: A family of 1-player energy parity games where player 1 needs memory of size $2 \cdot (n-1) \cdot W$ and initial credit $(n-1) \cdot W$. Edges are labeled by weights, states by priorities.

3.5 Energy Parity Games

The main result about energy parity games is that on the one hand exponential memory is required in general for winning strategies of player 1, and on the other hand the winner can be decided in NP ∩ coNP. The NP upper bound may look surprising, while the coNP upper bound follows from the fact that memoryless strategies are sufficient for player 2.

We provide some intuition of the results for energy Büchi games.

Theorem 3.10 (Strategy Complexity) *For all energy parity games, the following assertions hold: (1) winning strategies with memory of size $n \cdot d \cdot W$ exist for player 1; (2) memoryless winning strategies exist for player 2.*

Example 3.3 (Memory requirement) *In Figure 3.3, we present a family of player-1 games in which memory of size $2 \cdot (n-1) \cdot W + 1$ is necessary to win. Note that the weights are encoded in binary, and thus W is exponential in the input size. To satisfy the parity condition, the play has to visit the initial state infinitely often, and to maintain the energy positive, the play has to visit the state with the positive-weighted self-loop. Since the paths between these two state have weight $-(n-1) \cdot W$, it is easy to see that initial credit $(n-1) \cdot W$ is necessary, and the self-loop has to be taken $M = 2 \cdot (n-1) \cdot W$ times requiring memory of size $M + 1$.*

While exponential memory may be necessary to win, we show that winning strategies in energy Büchi games can be decomposed into two memoryless strategies: an *attractor* strategy which forces to reach a Büchi state, and a *good-for-energy* strategies defined below. The cycle decomposition of an infinite path ρ is defined as follows: we push the states of ρ onto a stack. Whenever we push a state already in the stack, a cycle γ_i is formed and we remove it from the stack. The *cycle decomposition* of ρ is the sequence $\gamma_1, \gamma_2, \ldots$ of cycles so obtained.

Good-for-energy strategy. A strategy σ for player 1 is *good-for-energy* in state q if for all outcomes $\rho = q_0 q_1 \ldots$ of σ such that $q_0 = q$, for all cycles γ in the cycle decomposition of ρ, either $\mathsf{EL}(\gamma) > 0$, or $\mathsf{EL}(\gamma) = 0$ and γ is even (i.e., $\min\{p(q) \mid q \in \gamma\}$ is even).

Attractor. The player-1 *attractor* of a given set $S \subseteq Q$ is the set of states from which player 1 can force to reach a state in S. It is defined as the limit $Attr_1(S)$ of the sequence $A_0 = S$, $A_{i+1} = A_i \cup \{q \in Q_1 \mid \exists (q, q') \in E : q' \in A_i\} \cup \{q \in Q_2 \mid \forall (q, q') \in E : q' \in A_i\}$ for all $i \geq 0$. The *attractor strategy* for player 1 is defined as expected.

Using results of [BSV04], we can show that memoryless good-for-energy strategies exist. It is well known that attractor strategies are memoryless. An NP algorithm for energy Büchi games consist in guessing the set of winning states as well as the good-for-energy and attractor strategies. A winning strategy for player 1 would play the good-for-energy strategy until a large energy level is reached (say $2 \cdot |Q| \cdot W$). Note that if the energy never reaches that value, then by definition of good-for-energy strategies the Büchi objectives is satisfied and player 1 wins. Otherwise, player 1 switches to the attractor strategy, and within $|Q|$ steps reaches a Büchi state, with decrease of energy level at most $|Q| \cdot W$. The remaining energy (at least $|Q| \cdot W$) is sufficient to start over. Analogous arguments give the general result for energy parity games.

Theorem 3.11 (Computational Complexity) *The initial credit problem for energy parity games can be solved in NP $\cap$ coNP.*

The structure of the proof naturally yields a fixpoint algorithm which generalizes the classical algorithm of McNaughton [McN93] and Zielonka [Zie98] for solving parity games [18].

Theorem 3.12 (Algorithmic Complexity) *The problem of deciding the existence of a finite initial credit for energy parity games can be solved in time $O(|E| \cdot d \cdot |Q|^{d+2} \cdot W)$.*

Finally, there is a tight relationship between energy parity games and mean-payoff parity games. The work in [CHJ05] shows that optimal strategies in mean-payoff parity games may require infinite memory in general, and whereas Theorem 3.10 shows that finite memory is sufficient in energy parity games, we show that energy parity games are polynomially equivalent to mean-payoff parity games, leading to NP $\cap$ coNP membership of the problem of deciding the winner in mean-payoff parity games, and leading to an algorithm for solving such games which is conceptually much simpler than the algorithm of [CHJ05], with essentially the same complexity (linear in the largest weight, and exponential in the number of states only).

Theorem 3.13 *The mean-payoff threshold problem for mean-payoff parity game can be decided in NP $\cap$ coNP.*

Theorem 3.14 *The problem of deciding the winner in mean-payoff parity games can be solved in time* $O(|E| \cdot d \cdot |Q|^{d+3} \cdot W \cdot (|Q| + 1))$.

The results on the complexity of energy and mean-payoff parity games are summarized in Table 3.1 Table 3.2.

3.6 Multi-Weighted Games

In a multi-weighted game, the edges of the graphs are labeled with k-dimensional vectors w of integer values, i.e., $w \in \mathbb{Z}^k$. The energy objective requires that, given an initial credit $v_0 \in \mathbb{N}^k$, the sum of v_0 and all the vectors labeling edges up to position i in the play is nonnegative, for all $i \in \mathbb{N}$. The mean-payoff objective is specified by a vector of threshold values $v \in \mathbb{Z}^k$ and requires that the mean values is at least v.

For multi-weighted energy games, we show that the unknown initial credit problem is coNP-complete. The result relies on the existence of memoryless strategies for player 2 [BJK10], and on the fact that checking whether a multi-weighted graph (in our case the graph obtained after fixing the strategy of player 2) contains a nonnegative cycle can be done in polynomial time [KS88]. The lower bound is obtained by a reduction from the complement of the 3SAT problem which is NP-complete [Pap94].

Consider a 3SAT formula ψ in CNF with clauses $C_1, C_2, \ldots, C_k$ over variables $\{x_1, x_2, \ldots, x_n\}$, where each clause consists of disjunctions of exactly three literals (a literal is a variable or its complement). Given the formula ψ, we construct a game graph as shown in Figure 3.4: from the initial state, player 1 chooses a clause, then player 2 chooses a literal that appears in the clause (i.e., makes the clause true). From every literal the next state is the initial state. In the multi-weight function there is a component for every literal. For edges from the initial state to the clause states, and from the clause states to the literals, the weight for every component is 0. For the edges from a literal y back to the initial state, the weight for the component of y is 1, the weight for the component of the complement of y is -1, and for all the other components the weight is 0.

Intuitively, if ψ is satisfiable, then player 2 wins by choosing in each clause a literal set to true by a satisfying assignment. Indeed, consider an arbitrary strategy for player 1: the resulting infinite play must visit some literal x infinitely often. The complement literal $\overline{x}$ is never visited, and every time literal x is visited, the component corresponding to $\overline{x}$ decreases by 1. It follows that the play is winning for player 2 for every finite initial credit.

On the other hand, if ψ is not satisfiable, then it suffices to show that there is no memoryless winning strategy for player 2. To every memoryless strategy for player 2 corresponds a (possibly partial) assignment of the variables. Since ψ is not satisfiable it follows that this assignment is conflicting (it assigns the same truth value to some

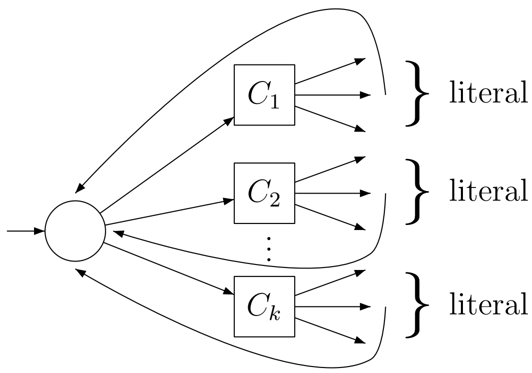


Figure 3.4: Game graph construction for a 3SAT formula (Lemma 3.15).

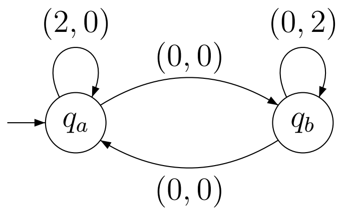


Figure 3.5: A multi-weighted mean-payoff game where infinite memory is necessary to win (Example 3.4).

literal x and its complement $\overline{x}_i$). Hence there must exist clauses C_i and C_j and variable x_k such that the strategy chooses the literal x_k in C_i and the complement $\overline{x}_k$ in C_j. The strategy for player 1 that alternates between clause C_i and C_j spoils the strategy of player 2 with initial credit 1 in each dimension.

Theorem 3.15 *The unknown initial credit and the mean-payoff threshold problems for multi-weighted two-player game structures are coNP-complete.*

Observe that our hardness proof works with weights restricted to the set $\{-1, 0, 1\}$.

If we restrict our attention to finite-memory strategies for multi-weighted games, then energy and mean-payoff games are equivalent.

Theorem 3.16 *For all multi-weighted two-player game structures G with dimension k, the answer to the unknown initial credit problem is* Yes *if and only if the answer to the mean-payoff threshold problem (for finite memory) with threshold vector $\{0\}^k$ is* Yes.

However, in general infinite memory is required to win generalized mean-payoff games.

Example 3.4 *Figure 3.5 shows a game where all states belong to player* 1*, and* (*a*) *for* $\underline{\mathsf{MP}}$*, player* 1 *can achieve a threshold vector* $(1, 1)$*, and* (*b*) *for* $\overline{\mathsf{MP}}$*, player* 1 *can achieve a threshold vector* $(2, 2)$*;* (*c*) *if we restrict player* 1 *to use a finite-memory strategy, then it is not possible to win the multi mean-payoff objective with threshold* $(1, 1)$ *(and thus also not with* $(2, 2)$*). To prove* (*a*)*, consider the strategy that visits* n *times* q_a *and then* n *times* q_b*, and repeats this forever with increasing value of* n*. This guarantees a mean-payoff vector* $(1, 1)$ *for* $\underline{\mathsf{MP}}$ *because in the long-run roughly half of the time is spent in* q_a *and roughly half of the time in* q_b*. To prove* (*b*)*, consider the strategy that alternates visits to* q_a *and* q_b *such that after the* n*th alternation, the self-loop on the visited state* q *(*$q \in \{q_a, q_b\}$*) is taken so many times that the average frequency of* q *gets larger than* $\frac{1}{n}$ *in the current finite prefix of the play. This is always possible and achieves threshold* $(2, 2)$ *for* $\overline{\mathsf{MP}}$*. Note that the above two strategies require infinite memory. To prove* (*c*)*, notice that finite-memory strategies produce an ultimately periodic play and therefore* $\underline{\mathsf{MP}}$ *and* $\overline{\mathsf{MP}}$ *coincide with* MP*. It is easy to see that such a play cannot achieve* $(1, 1)$ *because the periodic part would have to visit both* q_a *and* q_b *and then the mean-payoff vector* (v_1, v_2) *of the play would be such that* $v_1 + v_2 < 2$ *and thus* $v_1 = v_2 = 1$ *is impossible.*

Theorem 3.17 *In generalized mean-payoff games, infinite memory may be necessary to win (finite-memory strategies may not be sufficient).*

The mean-payoff threshold problem for generalized mean-payoff games with arbitrary (not necessarily finite-memory) strategies has been recently shown to be coNP-complete [VR11].

3.7 Conclusion and Perspectives

Studying the *energy objective* in games on graphs has provided new insights and a number of interesting results in the field. For example, the tight connection of energy games with mean-payoff games lead to an improved bound for solving mean-payoff games, and a new algorithm for solving mean-payoff parity games. A better understanding of the complexity of games on (multi-)counter systems is also arising.

Promising research directions to explore in the future include:

- solving the open questions related to energy games with partial-observation in order to get a complete picture. For instance, the proof of Theorem 3.8 requires two observations in the game. The question of blind games (with only one observation) is intriguing as it may still be decidable. To obtain a complete picture, we would need to solve the question of Theorem 3.7 for non-strict threshold, and the question of Theorem 3.8 for blind games, and for all combinations of strict and non-strict threshold, and mean-payoff defined as $\liminf$ or $\limsup$.

- in multi-weighted games, a recent result shows that in two dimensions, energy games are solvable in polynomial time [Cha10]. The question of the existence of a polynomial-time algorithm for solving energy games in three dimensions, or more generally in fixed dimension, remains open.

- the extension of the two-player games, to stochastic games is an active direction of research. The questions arising in the various forms of energy stochastic games are technical and seem to be really challenging.

- with Thierry Massart and our PhD student Mahsa Shirmohammadi, we are studying new semantics for probabilistic automata, which gives a new class of quantitative objectives in stochastic games (see also [47, 16] and the paragraph on MDPs in the introduction of this manuscript).

- with Dietmar Berwanger and our PhD student Julien Reichert, we plan to study the decidability frontier of various extensions of games with counters, and to explore connections with other quantitative frameworks such as regular cost functions, distance automata, and their game extension [CL08, Col09].

Chapter 4

Quantitative Languages

> Je suis anarchiste au point de toujours traverser dans les clous pour ne pas avoir à discuter avec la maréchaussée.
>
> Georges Brassens.

4.1 Introduction

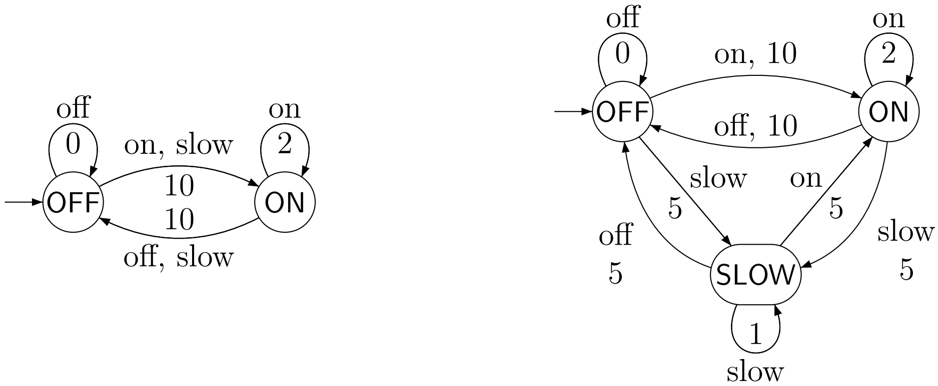
The automata-theoretic approach to verification is Boolean. To check that a system satisfies a specification, we construct a finite automaton A to model the system and a finite (usually nondeterministic) automaton B for the specification. The language $L(A)$ of A contains all behaviors of the system, and $L(B)$ contains all behaviors allowed by the specification. The language of an automaton A can be seen as a Boolean function L_A that assigns 1 (or true) to words in $L(A)$, and 0 (or false) to words not in $L(A)$. The verification problem "does the system satisfy the specification?" is then formalized as the language-inclusion problem "is $L(A) \subseteq L(B)$?", or equivalently, "is $L_A(w) \leq L_B(w)$ for all words w?". We investigate a natural generalization of this framework: a *quantitative language* L is a function that assigns a real-numbered value $L(w)$ to each (finite or infinite) word w. With quantitative languages, systems and specifications can be formalized more accurately. For example, a system may use a varying amount of some resource (e.g., memory consumption, or power consumption) depending on its behavior, and a specification may assign a maximal amount of available resource to each behavior, or fix the long-run average available use of the resource. The quantitative language-inclusion problem "is $L_A(w) \leq L_B(w)$ for all words w?" can then be used to check, say, if for each behavior, the peak power used by the system lies below the bound given by the specification; or if for each behavior, the long-run average response time of the system lies below the specified average response requirement.

In the Boolean automaton setting, the value of a word w in $L(A)$ is the maximal value of a run of A over w (if A is nondeterministic, then there may be many runs of

A over w), and the value of a run is a function that depends on the class of automata: for automata over finite words, the value of a run is true if the last state of the run is accepting; for Büchi automata, the value is true if an accepting state is visited infinitely often; etc. To define quantitative languages, we use automata with weights on transitions. We again set the value of a word w as the maximal value of all runs over w, and the value of a run r is a function of the (finite or infinite) sequence of weights that appear along r. This approach is well-known from the theory of weighted automata and in this work we consider several new ways for computing the values of runs.

We consider functions, such as Max and Sum of weights for finite runs, and Sup, LimSup, LimInf, limit average, and discounted sum of weights for infinite runs. For example, peak power consumption can be modeled as the maximum of a sequence of weights representing power usage; energy use can be modeled as the sum; average response time as the limit average [CCH+05, CdHS03]. Quantitative languages have also been used to specify and verify reliability requirements: if a special symbol $\perp$ is used to denote failure and has weight 1, while the other symbols have weight 0, one can use a limit-average automaton to specify a bound on the rate of failure in the long run [CGH+08, BCHJ09]. Alternatively, the discounted sum can be used to specify that failures happening later are less important than those happening soon [dAHM03]. It should be noted that LimSup and LimInf automata generalize Büchi and coBüchi automata, respectively. Functions such as limit average (or mean payoff) and discounted sum are classical in game theory [Sha53]; they have been studied extensively as quantitative objectives in the branching-time context of games played on graphs [EM79, Con92, CdHS03, Gim06]. The linear-time setting of automata and languages provides a uniform way to describe quantitative specifications (e.g., quantitative objectives as monitors in games) using the above functions, and allows to compare their expressive power and study their reducibility relationship. It is therefore natural to consider the same functions in the linear-time context of automata and languages that have been widely studied in the branching-time context of games.

Example 4.1 *We illustrate the use of limit-average automata to model the energy consumption of a motor. Energy-aware design has emerged as an important topic in the recent years, and our work could be used in that direction, as illustrated by the example. Since we consider energy consumption in the long-run, it is natural to accumulate the weights as limit-average (the total energy consumed is the sum of the amounts of consumed energy). The automaton A in Figure 4.1(a) specifies the maximal allowed energy consumption to maintain the motor on or off, and the maximal consumption for a mode change. The specification abstracts away that a mode change can occur smoothly with the slow command. A refined specification B is given in Figure 4.1(b) where the effect of slowing down is captured by a third state. One can check that B refines A, i.e. $L_B(w) \leq L_A(w)$ for all words $w \in \{on, off, slow\}^\omega$, hence the limit-average consumption in B always satisfies the bound specified by A.*



(a) Limit-average automaton A.

(b) Limit-average automaton B.

Figure 4.1: Specifications for the energy consumption of a motor (B refines A).

We make the following remarks about this example. First, to check that B refines A, it would not be sufficient to check locally that transitions in B have smaller weight than corresponding transitions in A. For instance, the word $slow \cdot on^\omega$ visits the sequences of weights $10, 2, 2, 2, \dots$ in A and $5, 5, 2, 2, \dots$ in B, the second weight in the sequences being larger in B than in A. The algorithmic problem of deciding whether refinement holds is discussed in Section 4.3 (for instance, Theorem 4.3 shows that refinement can be decided in polynomial time for deterministic limit-average automata). Second, if we would assign to an infinite run the supremum of its weights instead of the limit-average, then the automaton B would still refine A as the word off^ω would have value 0 in both A and B, and for all other words w, we would have $L_A(w) = 10$ and $L_B(w) \leq 10$. Now, if we assign weight 4 to the transition from ON to OFF in A, then the refinement of A by B still holds if we use Sup-automata (by exactly the same argument as before), but it fails for limit-average (e.g., for $w = (on \cdot off)^\omega$, we have $L_B(w) = 10$ and $L_A(w) = 7 < 10$).

We attempt a systematic study of quantitative languages defined by weighted automata. The main novelties concern quantitative languages of infinite words, and especially those that have no Boolean counterparts (i.e., limit-average and discounted-sum languages). Therefore we focus on infinite words in this manuscript, and quantitative languages of finite words are studied in [5].

In the first part of this chapter, we consider generalizations of the Boolean *decision problems* of emptiness, universality, language inclusion, and language equivalence. The quantitative emptiness problem asks, given a weighted automaton A and a rational number ν, whether there exists a word w such that $L_A(w) \geq \nu$. This problem can be reduced to a one-player game with a quantitative objective and is therefore solvable in polynomial time for all value functions considered here. The quantitative universality problem asks whether $L_A(w) \geq \nu$ for all words w. This problem can be formulated as a two-player partial-observation game: one player chooses input letters and the other player chooses the successor states, and the first player, whose goal is to construct a word w such that $L_A(w) < \nu$, is not allowed to see the state chosen by the second player. The problem is PSPACE-complete for simple functions like Sup, LimSup, and LimInf, it is undecidable for limit-average (see Section 3.4), and we do not know if it is decidable or not for discounted-sum automata (the corresponding partial-observation games are not known to be decidable either). The same situation holds for the quantitative language-inclusion and language-equivalence problems, which ask, given two weighted automata A and B, if $L_A(w) \leq L_B(w)$ (resp., $L_A(w) = L_B(w)$) for all words w. Therefore we introduce a notion of quantitative simulation between weighted automata, which generalizes Boolean simulation relations, is decidable, and implies language inclusion. Simulation corresponds to a weaker version of the above game, where the first player has perfect information about the state of the game. In particular, this implies that quantitative simulation can be decided in NP $\cap$ coNP for limit-average and discounted-sum automata.

In the second part of this chapter, we present a complete characterization of the *expressive power* of the various classes of weighted automata, by comparing the classes of quantitative languages they define. The complete picture relating the expressive powers of weighted automata is shown in Figure 4.3 and Table 4.2. For instance, the results for LimSup and LimInf are analogous to the special Boolean cases of Büchi and coBüchi (nondeterminism is strictly more expressive for LimSup, but not for LimInf). In the limit-average and discounted-sum cases, nondeterministic automata are strictly more expressive than their deterministic counterparts. Also, one of our results shows that nondeterministic limit-average automata are not as expressive as deterministic Büchi automata (and vice versa). Note that deterministic Büchi languages are complete for the second level of the Borel hierarchy [Tho97], and deterministic limit-average languages are complete for the third level [Cha07].

In the third part of this chapter, we study the *closure properties* of quantitative languages. It is natural and convenient to decompose a specification or a design into several components, and to apply composition operators to obtain a complete specification. We consider a natural generalization of the classical operations of union, intersection, and complement of Boolean languages. We define the *maximum*, *minimum*, and *sum* of two quantitative languages L_1 and L_2 as the quantitative language that assigns $\max(L_1(w), L_2(w))$, $\min(L_1(w), L_2(w))$, and $L_1(w) + L_2(w)$ to each word w. The *complement* L^c of a quantitative language L is defined by $L^c(w) = 1 - L(w)$ for all words w. The sum is a natural way of composing two automata if the weights represent costs (e.g., energy consumption). We give another example in Section 4.2 to illustrate the composition operators and the use of quantitative languages as a specification framework.

We give a complete picture of the closure properties of the various classes of quantitative languages over infinite words under maximum, minimum, complement and sum (see Table 4.3). For instance, (non)deterministic limit-average automata are not closed under sum and complement, while nondeterministic discounted-sum automata are closed under sum but not under complement. All other classes of weighted automata are closed under sum. The closure properties of Sup-, LimSup-, and LimInf-automata are obtained as a direct extension of the results for Boolean finite automata, while for limit-average and discounted-sum automata, the proofs require the analysis of the structure of the automata cycles and properties of the solutions of polynomials with rational coefficients. Note that the quantitative language-inclusion problem "is $L_A(w) \leq L_B(w)$ for all words w?" reduces to closure under sum and complement, because it is equivalent to the question of the non-existence of a word w such that $L_A(w) + L_B^c(w) > 1$, an *emptiness* question which is decidable for all classes of quantitative languages (Theorem 4.1). Also note that deterministic limit-average and discounted-sum automata are not closed under maximum, which implies that nondeterministic automata are strictly more expressive in these cases (because the maximum can be obtained by an initial nondeterministic choice).

In the last part of this chapter, we introduce a new class of quantitative languages, defined by *mean-payoff automaton expressions*, which is *robust* and *decidable*, i.e., a class which is closed under the four pointwise operations of max, min (which generalize union and intersection respectively), numerical complement, and sum, and for which all the quantitative decision problems are decidable (emptiness, universality, language inclusion, and equivalence), and that can express all natural examples of quantitative languages defined using the mean-payoff measure [ADMW09, 27, CGH+08]. Mean-payoff automaton expressions subsume deterministic LimAvg-automata, and we show that they have expressive power incomparable to nondeterministic and alternating LimAvg-automata. We also present for the first time an algorithm to compute distance between two quantitative languages given as mean-payoff automaton expressions.

In our attempt to obtain such a class, we have studied alternating weighted automata [26], and probabilistic weighted automata [28] with so-called almost-sure and positive semantics. We do not give details of these works here as none of the class of quantitative languages defined by these automata is both robust and decidable.

Related works. In the literature, there is a wealth of results on weighted automata on finite and infinite words. We now describe the differences with the setting presented in this chapter. The lattice automata of [KL07] map finite words to values from a finite lattice. Roughly speaking, the value of a run is the meet (greatest lower bound) of its transition weights, and the value of a word w is the join (least upper bound) of the values of all runs over w. This corresponds to Min and Inf automata in our setting, and for infinite words, the Büchi lattice automata of [KL07] are analogous to our LimSup automata. However, the other classes of weighted automata (Sum, limit-average, discounted-sum) cannot be defined using operations on finite lattices. The complexity of the emptiness and universality problems for lattice automata is given in [KL07] (and implies our results for LimSup automata), while their generalization of language inclusion differs from ours. They define the *implication value* $v(A, B)$ of two lattice automata A and B as the meet over all words w of the join of $\neg L_A(w)$ and $L_B(w)$, while we would define implication value as $v(A, B) = \min_w(L_B(w) - L_A(w))$ since min is the meet operation (and defining negation as multiplication by -1, but using $+$ instead of join), and say that B refines A if $v(A, B) \geq 0$.

In classical weighted automata [Sch61, Moh97] and semiring automata [KS86] (i.e., finite automata with transition weights in a semiring structure), the value of a finite word is defined using the two algebraic operations $+$ and $\cdot$ of a semiring as the sum of the product of the transition weights of the runs over the word. In that case, quantitative languages are called *formal power series* for finite words [Sch61, KS86], and ω-*series* for infinite words [CK94, DK03, ÉK04]. Over infinite words, weighted automata with discounted sum were first investigated in [DK03]. Note that the quantitative languages studied in this chapter use operations over rational numbers that do not form a semiring. Researchers have also considered other quantitative generalizations of languages over finite words [DG07], over trees [DKR08], and using finite lattices [GC03]. However, these works do not address the quantitative decision problems, nor do they compare the relative expressive power and closure properties of weighted automata over infinite words, as we do here. The work of [CK94] studies the decision problems for weighted automata but for different notion of behaviors (different value functions). The works of [Kar05, SSMH04] consider quantitative counting properties, and the works of [KR03] consider the cardinality properties in monadic second order logic, but the value functions we consider are different from the counting and cardinality properties. The works [SSM03, DZL03] consider numerical properties of documents such as XML that are very different from the quantitative properties we consider. In [CCH$^+$05], a quantitative generalization of languages is defined by discrete functions (the value of a word is an integer) and the decision problems only involve the extremal value of a language, which corresponds to emptiness.

In models that use transition weights as probabilities, such as probabilistic *Rabin automata* [Paz71], one does not consider values of individual infinite runs (which would usually have a value, or measure, of 0), but only measurable sets of infinite runs (where basic open sets are defined as extensions of finite runs). Our quantitative setting is orthogonal to the probabilistic framework: we assign quantitative values (e.g., peak power consumption, average response time, failure rate) to individual infinite behaviors, not probabilities to finite behaviors.

4.2 Quantitative Languages

First, we recall the classical automata-theoretic description of Boolean languages, and introduce an automata-theoretic description of several classes of quantitative languages. We consider Boolean languages $L \subseteq \Sigma^\omega$ of infinite words over a finite alphabet Σ. We have also investigated languages of finite words in [5, 27]. Alternatively, we can view a Boolean language as a function in $[\Sigma^\omega \to \{0, 1\}]$.

We recall the definition of (nondeterministic finite) automaton. A *(finite) automaton* is a tuple $A = \langle Q, q_I, \Sigma, \delta \rangle$ where:

- Q is a finite set of states, and $q_I \in Q$ is the initial state;
- Σ is a finite alphabet;
- $\delta \subseteq Q \times \Sigma \times Q$ is a finite set of labeled transitions.

The automaton A is *total* if for all $q \in Q$ and $\sigma \in \Sigma$, there exists $(q, \sigma, q') \in \delta$ for at least one $q' \in Q$. The automaton A is *deterministic* if for all $q \in Q$ and $\sigma \in \Sigma$, there exists $(q, \sigma, q') \in \delta$ for exactly one $q' \in Q$. We sometimes call automata *nondeterministic* to emphasize that they are not necessarily deterministic.

A *run* of A over a finite (resp., infinite) word $w = \sigma_1 \sigma_2 \dots$ is a finite (resp., infinite) sequence $r = q_0 \sigma_1 q_1 \sigma_2 \dots$ of states and letters such that (i) $q_0 = q_I$, and (ii) $(q_i, \sigma_{i+1}, q_{i+1}) \in \delta$ for all $0 \leq i < |w|$. When the run r is finite, we denote by $\mathsf{Last}(r)$ the last state in r. When r is infinite, we denote by $\mathsf{Inf}(r)$ the set of states that occur infinitely many times in r. The prefix of length i of an infinite run r is the prefix of r that contains the first i states.

Given a set $F \subseteq Q$ of final (or accepting) states, the *finite-word language* defined by the pair $\langle A, F \rangle$ is $L^{\mathrm{f}}_A = \{w \in \Sigma^* \mid$ there exists a run r of A over w such that $\mathsf{Last}(r) \in F\}$. The *infinite-word languages* defined by $\langle A, F \rangle$ are as follows: if $\langle A, F \rangle$ is interpreted as a Büchi automaton, then $L^{\mathrm{b}}_A = \{w \in \Sigma^\omega \mid$ there exists a run r of A over w such that $\mathsf{Inf}(r) \cap F \neq \varnothing\}$, and if $\langle A, F \rangle$ is interpreted as a coBüchi automaton, then $L^{\mathrm{c}}_A = \{w \in \Sigma^\omega \mid$ there exists a run r of A over w such that $\mathsf{Inf}(r) \subseteq F\}$. In the sequel, we assume that the set F is given with the description of the finite automaton A, and we often omit the superscripts in the notation L^{f}_A, L^{b}_A, and L^{c}_A, assuming that automata have a type (finite-word, Büchi, or coBüchi) that determines which language it defines.

Boolean decision problems. We recall the classical decision problems for automata, namely, emptiness, universality, language inclusion and language equivalence. Given a finite automaton A, the *Boolean emptiness problem* asks whether $L_A = \varnothing$, and the *Boolean universality problem* asks whether $L_A = \Sigma^*$ (for finite-word language) or $L_A = \Sigma^\omega$ (for infinite-word language). Given two finite automata A and B, the *Boolean language-inclusion problem* asks whether $L_A \subseteq L_B$, and the *Boolean language-equivalence problem* asks whether $L_A = L_B$. It is well-known that for both finite- and infinite-word languages, the emptiness problem is solvable in polynomial time, while the universality, inclusion, and equivalence problems are PSPACE-complete [MS72, SVW87].

Weighted automata. A *quantitative language* L over a finite alphabet Σ is a mapping $L : \Sigma^\omega \to \mathbb{R}$, where $\mathbb{R}$ is the set of real numbers.

A *weighted automaton* is a tuple $A = \langle Q, q_I, \Sigma, \delta, \gamma \rangle$ where:

- $\langle Q, q_I, \Sigma, \delta \rangle$ is a total finite automaton, and
- $\gamma : \delta \to \mathbb{Q}$ is a *weight* function, where $\mathbb{Q}$ is the set of rational numbers.

Given a finite (resp., infinite) run $r = q_0\sigma_1 q_1\sigma_2 \dots$ of A over a finite (resp., infinite) word $w = \sigma_1\sigma_2 \dots$, let $\gamma(r) = v_0 v_1 \dots$ be the sequence of weights defined by $v_i = \gamma(q_i, \sigma_{i+1}, q_{i+1})$ for all $0 \leq i < |w|$.

Given a *value function* $\mathsf{Val} : \mathbb{Q}^+ \to \mathbb{R}$ (resp., $\mathsf{Val} : \mathbb{Q}^\omega \to \mathbb{R}$), the Val-automaton A defines the quantitative language L_A such that for all words $w \in \Sigma^+$ (resp., $w \in \Sigma^\omega$), we have $L_A(w) = \sup\{\mathsf{Val}(\gamma(r)) \mid r \text{ is a run of } A \text{ over } w\}$. We assume that $\mathsf{Val}(v)$ is bounded when the numbers in v are taken from a finite set (namely, the set of weights in A), and since weighted automata are total, $L_A(w)$ is not infinite. All value functions we consider in this chapter satisfy this boundedness assumption.

For infinite words, we consider the following classical value functions from $\mathbb{Q}^\omega$ to $\mathbb{R}$. Given an infinite sequence $v = v_0 v_1 \dots$ of rational numbers taken from a finite set V (i.e., $v_i \in V$ for all $i \geq 0$), define

- $\mathsf{Sup}(v) = \sup\{v_n \mid n \geq 0\}$;
- $\mathsf{LimSup}(v) = \limsup\limits_{n\to\infty} \; v_n = \lim\limits_{n\to\infty} \sup\{v_i \mid i \geq n\}$;
- $\mathsf{LimInf}(v) = \liminf\limits_{n\to\infty} \; v_n = \lim\limits_{n\to\infty} \inf\{v_i \mid i \geq n\}$;
- $\mathsf{LimAvg}(v) = \liminf\limits_{n\to\infty} \; \dfrac{1}{n} \cdot \sum\limits_{i=0}^{n-1} v_i$;
- given a discount factor $0 < \lambda < 1$, $\mathsf{Disc}_\lambda(v) = \sum\limits_{i=0}^{\infty} \lambda^i \cdot v_i$.

Intuitively for a sequence $v = v_0 v_1 \dots$ of rational numbers from the finite set V the Sup function chooses the maximal number that appear in v; the LimSup function chooses the maximal number that appear infinitely often in v; the LimInf function chooses the maximal number ℓ such that from some point on all numbers that are visited are at least ℓ; the LimAvg functions gives the long-run average of the numbers in v; and the Disc_λ gives the discounted sum of the numbers in v. For decision problems, we always assume that the discount factor λ is a rational number. Note that $\mathsf{LimAvg}(v)$ is defined using lim inf and is therefore well-defined; all results of this chapter hold also if the limit average of v is defined instead as $\limsup_{n\to\infty} \frac{1}{n} \cdot \sum_{i=0}^{n-1} v_i$. One could also consider the value function $\inf\{v_n \mid n \geq 0\}$ and obtain results analogous to the Sup value function.

Note that for Boolean automata, if we assign value 1 to the accepting runs (either those that end up in an accepting state, or visit an accepting state infinitely often, or eventually visit accepting states only) and value 0 to the other runs, then the function L_A would be the characteristic function of the Boolean language defined by A. Hence, the sup operator is a natural generalization to the quantitative setting of the way nondeterminism is dealt with in Boolean automata. Other definitions can be considered [26], like choosing inf instead of sup which would correspond to the so-called *universal* automata in the Boolean case [KV01].

In the sequel, we denote by n the number of states and by m the number of transitions of a given automaton. We assume that rational numbers are given as pairs of integers, encoded in binary. All time bounds we give in this chapter assume that the largest size of an integer in the input is a constant p. Without this assumption, most complexity results would involve a polynomial factor in p, as they require polynomially many operations of addition, multiplication, and comparison of rational numbers, which are quadratic in p.

Quantitative decision problems and distance. We now present quantitative generalizations of the classical decision problems for automata. Given two quantitative languages L_1 and L_2 over Σ, we write $L_1 \sqsubseteq L_2$ if $L_1(w) \leq L_2(w)$ for all words $w \in \Sigma^+$ (resp. $w \in \Sigma^\omega$). Given a weighted automaton A and a rational number $\nu \in \mathbb{Q}$, the *quantitative emptiness problem* asks whether there exists a word $w \in \Sigma^+$ (resp., $w \in \Sigma^\omega$) such that $L_A(w) \geq \nu$, and the *quantitative universality problem* asks whether $L_A(w) \geq \nu$ for all words $w \in \Sigma^+$ (resp., $w \in \Sigma^\omega$). Given two weighted automata A and B, the *quantitative language-inclusion problem* asks whether $L_A \sqsubseteq L_B$, and the *quantitative language-equivalence problem* asks whether $L_A = L_B$, that is, whether $L_A(w) = L_B(w)$ for all $w \in \Sigma^+$ (resp., $w \in \Sigma^\omega$). All results in this chapter also hold for the decision problems defined above with inequalities replaced by strict inequalities. Finally, the *distance* between L_1 and L_2 is $D_{\sup}(L_1, L_2) = \sup_{w\in\Sigma^\omega}|L_1(w) - L_2(w)|$. It measures how close is an implementation L_1 as compared to a specification L_2.

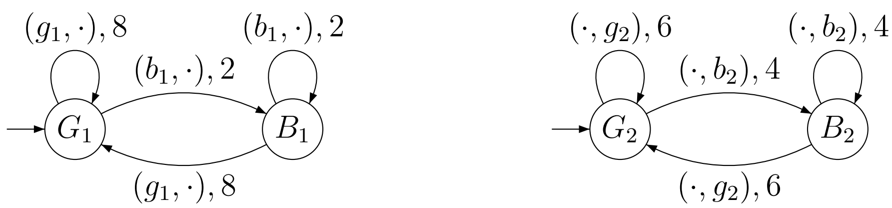
Expressiveness. A class $\mathcal{C}$ of weighted automata *can be reduced* to a class $\mathcal{C}'$ of weighted automata if for every $A \in \mathcal{C}$ there exists $A' \in \mathcal{C}'$ such that $L_A = L_{A'}$. In particular, a class of weighted automata *can be determinized* if it can be reduced to its deterministic counterpart. All reductions presented in this chapter are constructive: when $\mathcal{C}$ can be reduced to $\mathcal{C}'$, we always show how to construct an automaton $A' \in \mathcal{C}'$ that defines the same quantitative language as a given automaton $A \in \mathcal{C}$. We say that the *cost* of a reduction is $O(f(n, m))$ if for all automata $A \in \mathcal{C}$ with n states and m

transitions, the constructed automaton $A' \in \mathcal{C}'$ has at most $O(f(n,m))$ many states. For all reductions we present, the size of the largest transition weight in A' is linear in the size p of the largest weight in A (however, the time needed to compute these weights may be quadratic in p).

Composition. Given two quantitative languages L and L' over Σ, and a rational number c, we denote by $\max(L, L')$ (resp., $\min(L, L')$, $L + L'$, $k + L$, and kL) the quantitative language that assigns $\max\{L(w), L'(w)\}$ (resp., $\min\{L(w), L'(w)\}$, $L(w)+L'(w)$, $k + L(w)$, and $k \cdot L(w)$) to each word $w \in \Sigma^+$ (or $w \in \Sigma^\omega$). We say that $k + L$ is the *shift by* k of L and that kL is the *scale by* k of L. The language $L^c = 1 - L$ is called the *complement* of L. The max, min and complement operators for quantitative languages generalize respectively the union, intersection and complement operator for Boolean languages. For instance, De Morgan's laws hold (the complement of the max of two languages is the min of their complement, etc.) and complementing twice leave languages unchanged.

Example 4.2 *Consider an investment of 100 dollars that can be made in two banks A_1 and A_2 as follows: (a) 100 dollars to bank A_1, (b) 100 dollars to bank A_2, or (c) 50 dollars to bank A_1 and 50 dollars to bank A_2. The banks can be either in a good state (denoted G_1, G_2) or in a bad state (denoted B_1, B_2). If it is in a good state, then A_1 offers 8% reward while A_2 offers 6% reward. If it is in a bad state, then A_1 offers 2% reward while A_2 offers 4% reward. The change of state is triggered by the input symbols b_1, b_2 (from a good to a bad state) and g_1, g_2 (from a bad to a good state). The rewards received earlier weight more than rewards received later due to inflation represented by the discount factor. The automata A_1 and A_2 in Figure 4.2 specify the behavior of the two banks for an investment of 100 dollars, where the input alphabet is $\{g_1, b_1\} \times \{g_2, b_2\}$ (where the notation $(g_1, \cdot)$ represents the two letters (g_1, g_2) and (g_1, b_2), and similarly for the other symbols). If 50 dollars are invested in each bank, then we obtain automata C_1 and C_2 from A_1 and A_2 where each reward is halved. The combined automaton is obtained as the composition of C_1 and C_2 under the sum operator.*

Notation. Classes of weighted automata over infinite words are denoted with acronyms of the form xy where x is either N(ondeterministic), D(eterministic), or N/D (when deterministic and nondeterministic automata have the same expressiveness), and y is one of the following: Sup, Lsup (LimSup), Linf (LimInf), Lavg (LimAvg), or Disc. For Büchi and coBüchi condition, we use BW and CW respectively.



(a) 100 dollars invested in bank A_1.

(b) 100 dollars invested in bank A_2.

Figure 4.2: The discounted-sum automaton models of two banks.

4.3 The Complexity of Quantitative Decision Problems

We study the complexity of the quantitative decision problems for weighted automata over infinite words. The results are summarized in Table 4.1. We present a quantitative generalization of the notion of *simulation* as an approximation of language inclusion (simulation implies language inclusion) and we show that quantitative simulation can be decided with a lower complexity as compared to quantitative language inclusion.

4.3.1 Quantitative Emptiness, Universality, Language Inclusion and Equivalence

Emptiness. The quantitative emptiness problem can be solved by a reduction to the problem of finding the maximal value of an infinite path in a graph. This is decidable because pure memoryless strategies for resolving nondeterminism exist for all quantitative objectives that we consider [FV97, Kar78, And06].

Theorem 4.1 *The quantitative emptiness problem is solvable in $O(m+n)$ time for* Sup-*,* LimSup-*, and* LimInf-*automata; in $O(n \cdot m)$ time for* LimAvg-*automata; and in $O(n^2 \cdot m)$ time for* Disc-*automata.*

Language inclusion. The following theorem relies on the analogous results for finite automata (since Büchi and coBüchi automata are special cases of respectively LimSup- and LimInf-automata, with weights in $\{0,1\}$).

Theorem 4.2 *The quantitative language-inclusion problem is PSPACE-complete for* Sup-*,* LimSup-*, and* LimInf-*automata.*

It follows from the results of Chapter 3 that the quantitative language-inclusion problem is undecidable for LimAvg-automata. We do not know if this problem is

	$\exists$	$\forall$	$\subseteq$	$=$	simulation
Sup	PTime	PSpace	PSpace	PSpace	PTime
LimSup	PTime	PSpace	PSpace	PSpace	NP $\cap$ coNP
LimInf	PTime	PSpace	PSpace	PSpace	NP $\cap$ coNP
LimAvg	PTime	undec.	undec.	undec.	NP $\cap$ coNP
Deterministic LimAvg	PTime	PTime	PTime	PTime	PTime
Disc_λ	PTime	?	?	?	NP $\cap$ coNP
Deterministic Disc_λ	PTime	PTime	PTime	PTime	PTime

Table 4.1: Complexity upper bound for the quantitative decision problems ($\exists$) emptiness, ($\forall$) universality, ($\subseteq$) inclusion, ($=$) equivalence, and quantitative simulation.

decidable for Disc-automata. The special cases of deterministic automata can be solved using a product construction.

Theorem 4.3 *The quantitative language-inclusion problems $L_A \sqsubseteq L_B$ for* LimAvg- *and* Disc-*automata are decidable in polynomial time when B is deterministic. The problem is undecidable for nondeterministic* LimAvg-*automata.*

For Disc-automata, we have the following result which follows from the fact that for a negative instance of the problem (i.e., if there exists a word $w \in \Sigma^\omega$ such that $L_A(w) > L_B(w)$) there exists a finite word $w \in \Sigma^*$ such that $L_A(w \cdot w') > L_B(w \cdot w')$ for all words $w' \in \Sigma^\omega$. This follows from the fact that finite prefixes provide arbitrarily accurate approximations of infinite discounted sums, and therefore after a sufficiently long finite prefix of w, the separation of $L_A(w)$ and $L_B(w)$ can be established.

Theorem 4.4 *The quantitative language-inclusion problem for* Disc-*automata is co-r.e.*

Universality and language equivalence. All of the above results about language inclusion also hold for quantitative universality and quantitative language equivalence.

4.3.2 Quantitative Simulation

As the quantitative language-inclusion problem for limit-average automata is undecidable, and as it remains open for discounted-sum automata, we introduce a notion of *quantitative simulation* as a decidable approximation of language inclusion for weighted automata. The quantitative language-inclusion problem can be viewed as a partial-observation game, and we define the quantitative simulation problem as exactly the same game, but with perfect observation. For quantitative objectives, perfect-observation games can be solved much more efficiently than partial-observation games, and in some cases the solution of partial-observation games with quantitative objectives is not known (e.g., for discounted-sum games), or undecidable (e.g., for for limit-average games). For example, perfect-observation games with limit-average and discounted-sum objectives can be decided in NP $\cap$ coNP, whereas the solution for such partial-observation games is not known. Second, quantitative simulation implies quantitative language inclusion, because it is always easier to win a game when observation is not partial. Hence, as in the case of finite automata and Boolean languages, quantitative simulation can be used as a conservative and efficient approximation of quantitative language inclusion.

Language-inclusion game. Let A and B be two weighted automata with weight function γ_1 and γ_2, respectively, for which we want to check if $L_A \sqsubseteq L_B$. The language-inclusion game is played by a *challenger* and a *simulator*, for infinitely many rounds. The goal of the simulator is to prove that $L_A \sqsubseteq L_B$, while the challenger has the opposite objective. The position of the game in the initial round is $\langle q_I^1, q_I^2 \rangle$ where q_I^1 and q_I^2 are the initial states of A and B, respectively. In each round, if the current position is $\langle q_1, q_2 \rangle$, first the challenger chooses a letter $\sigma \in \Sigma$ and a state q_1' such that $(q_1, \sigma, q_1') \in \delta_1$, and then the simulator chooses a state q_2' such that $(q_2, \sigma, q_2') \in \delta_2$. The position of the game in the next round is $\langle q_1', q_2' \rangle$. The outcome of the game is a pair (r_1, r_2) of runs of A and B, respectively, over the same infinite word. The simulator wins the game if $\mathsf{Val}(\gamma_2(r_2)) \geq \mathsf{Val}(\gamma_1(r_1))$. To make this game equivalent to the language-inclusion problem, we require that the challenger cannot observe the state of B in the position of the game.

Simulation game. The simulation game is the language-inclusion game without the restriction on the vision of the challenger, that is, the challenger is allowed to observe the full position of the game. Formally, given $A = \langle Q_1, q_I^1, \Sigma, \delta_1, \gamma_1 \rangle$ and $B = \langle Q_2, q_I^2, \Sigma, \delta_2, \gamma_2 \rangle$, a *strategy* τ for the challenger is a function from $(Q_1 \times Q_2)^+$ to $\Sigma \times Q_1$ such that for all $\pi \in (Q_1 \times Q_2)^+$, if $\tau(\pi) = (\sigma, q)$, then $(\mathsf{Last}(\pi_{|Q_1}), \sigma, q) \in \delta_1$, where $\pi_{|Q_1}$ is the projection of π on Q_1^+. A strategy τ for the challenger is *blind* if $\tau(\pi) = \tau(\pi')$ for all sequences $\pi, \pi' \in (Q_1 \times Q_2)^*$ such that $\pi_{|Q_1} = \pi'_{|Q_1}$. The set of *outcomes* of a challenger strategy τ is the set of pairs (r_1, r_2) of runs such that if $r_1 = q_0\sigma_1 q_1 \sigma_2 \dots$ and $r_2 = q_0'\sigma_1 q_1' \sigma_2 \dots$, then $q_0 = q_I^1$, $q_0' = q_I^2$, and for all $i \geq 0$, we have $(\sigma_{i+1}, q_{i+1}) = \tau((q_0, q_0') \dots (q_i, q_i'))$ and $(q_i', \sigma_{i+1}, q_{i+1}') \in \delta_2$. A strategy τ for the challenger is *winning* if $\mathsf{Val}(\gamma_1(r_1)) > \mathsf{Val}(\gamma_2(r_2))$ for all outcomes (r_1, r_2) of τ.

Theorem 4.5 *For all value functions and weighted automata A and B, we have $L_A \sqsubseteq L_B$ if and only if there is no blind winning strategy for the challenger in the language-inclusion game for A and B.*

Given two weighted automata A and B, there is a *quantitative simulation* of A by B if there exists no (not necessarily blind) winning strategy for the challenger in the simulation game for A and B. We note that for the special cases of Büchi and coBüchi automata, quantitative simulation coincides with *fair simulation* [HKR97].

Corollary 4.6 *For all value functions and weighted automata A and B, if there is a quantitative simulation of A by B, then $L_A \sqsubseteq L_B$.*

Given two weighted automata A and B, the *quantitative simulation problem* asks if there is a quantitative simulation of A by B. The results of the next theorem are summarized in Table 4.1.

Theorem 4.7 *The quantitative simulation problem for* Sup*-automata is solvable in polynomial time. The quantitative simulation problem is in NP $\cap$ coNP for* LimSup*-,* LimInf*-,* LimAvg*-, and* Disc*-automata.*

4.4 The Expressive Power of Weighted Automata

We study the expressiveness of the different classes of weighted automata over infinite words by comparing the quantitative languages they can define. For this purpose, we show the existence and non-existence of translations between classes of finite and weighted automata. All reducibility relationships are summarized in Table 4.2 and Figure 4.3.

4.4.1 Positive Reducibility Results

We start with the positive results about the existence of reductions between various classes of weighted automata, most of which can be obtained by generalizing corresponding results for finite automata. Our results also hold if we allow transition weights to be irrational numbers.

First, it is clear that Büchi and coBüchi automata can be reduced to LimSup- and LimInf-automata, respectively. In addition, we have the following results.

Theorem 4.8 *(i)* Sup*-automata can be determinized in $O(2^n)$ time; (ii)* LimInf*-automata can be determinized in $O(m^n)$ time; (iii) Deterministic* Sup*-automata can be reduced to deterministic* LimInf*-, to deterministic* LimSup*-, and to deterministic* LimAvg*-automata, all in $O(n \cdot m)$ time; (iv)* LimInf*-automata can be reduced to* LimSup*- and to* LimAvg*-automata, both in $O(n \cdot m)$ time.*

Reducibility		Boolean			quantitative							
		N/DCW	DBW	NBW	N/DSup	N/DLinf	DLsup	NLsup	DLavg	NLavg	DDisc	NDisc
Boolean	N/DCW	·	×	✓	×	✓	×	✓	×	✓	×	×
	DBW	×	·	✓	×	×	×	✓	×	×	×	×
	NBW	×	×	·	×	×	×	✓	×	×	×	×
quantitative	N/DSup				·	✓	✓	✓	✓	✓	×	×
	N/DLinf				×	·	×	✓	×	✓	×	×
	DLsup				×	×	·	✓	×	×	×	×
	NLsup				×	×	×	·	×	×	×	×
	DLavg		×		×	×	×	×	·	✓	×	×
	NLavg				×	×	×	×	×	·	×	×
	DDisc				×	×	×	×	×	×	·	✓
	NDisc				×	×	×	×	×	×	×	·

Table 4.2: Reducibility relation. $\mathcal{C}$ is reducible to $\mathcal{C}'$ if the entry $R(\mathcal{C}, \mathcal{C}')$ is ✓.

4.4.2 Negative Reducibility Results

We show that all other reducibility relationships do not hold. The most important results in this section show that (i) deterministic coBüchi automata cannot be reduced to deterministic LimAvg-automata, deterministic Büchi automata cannot be reduced to LimAvg-automata, and (ii) neither LimAvg- nor Disc-automata can be determinized. Over the alphabet $\hat{\Sigma} = \{a, b\}$, we use in the sequel the Boolean languages L_F, which contains all infinite words with finitely many a's, and L_I, which contains all infinite words with infinitely many a's. We also use the following definition. A class $\mathcal{C}$ of finite automata *can be weakly reduced* to a class $\mathcal{C}'$ of weighted automata if for every $A \in \mathcal{C}$ there exists an $A' \in \mathcal{C}'$ such that $\inf_{w \in L_A} L_{A'}(w) > \sup_{w \notin L_A} L_{A'}(w)$. Intuitively, weak reductions may not preserve the values of the words, but preserve the order on values: for two words $w \in L_A$ (i.e., $L_A(w) = 1$) and $w' \notin L_A$ (i.e., $L_A(w') = 0$), we have both $L_A(w) > L_A(w')$ and $L_{A'}(w) > L_{A'}(w')$.

The classical proof that deterministic coBüchi automata cannot be reduced to deterministic Büchi automata can be adapted to show the following theorem.

Theorem 4.9 *Deterministic coBüchi automata cannot be reduced to deterministic* LimSup-*automata.*

Since deterministic LimAvg- and deterministic Disc-automata can define quantita-

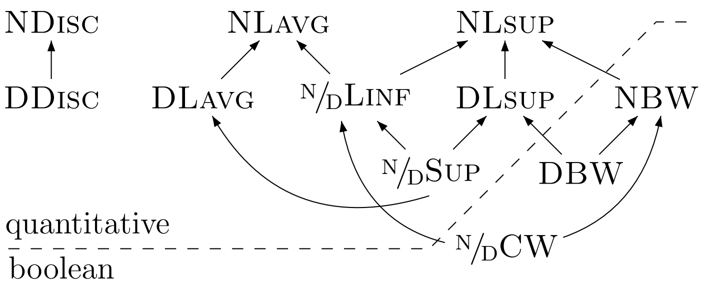


Figure 4.3: Reducibility relations: a class $\mathcal{C}$ of automata can be reduced to $\mathcal{C}'$ iff there is a path from $\mathcal{C}$ to $\mathcal{C}'$.

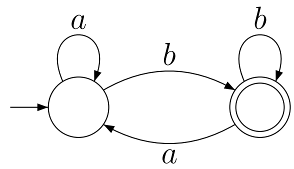


Figure 4.4: A deterministic coBüchi automaton.

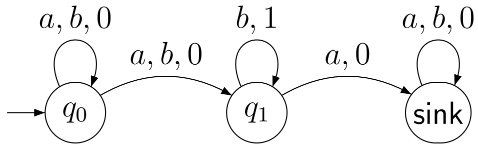


Figure 4.5: A nondeterministic limit-average automaton.

tive languages whose range is infinite, while LimSup-automata cannot, we obtain the following result.

Theorem 4.10 *Deterministic* LimAvg*-automata and deterministic* Disc*-automata cannot be reduced to* LimSup*-automata.*

The next theorem shows that nondeterministic LimAvg-automata are strictly more expressive than their deterministic counterpart. Theorem 4.12 will show that the expressive power of LimAvg- and LimSup-automata is incomparable.

Theorem 4.11 *Deterministic coBüchi automata cannot be weakly reduced to deterministic* LimAvg*-automata, and therefore they cannot be reduced to deterministic* LimAvg*-automata.* LimAvg*-automata cannot be determinized.*

Consider the language L_F of finitely many a's, which is the language defined by the deterministic coBüchi automaton shown in Figure 4.4. It is also easy to see that the nondeterministic LimAvg-automaton shown in Figure 4.5 defines L_F.

We need to show that L_F cannot be defined by any deterministic LimAvg-automaton to prove the desired claims. Intuitively, a deterministic LimAvg-automaton for L_F should assign positive weight to long sequences of b's (if the sequence is longer than the number of states of the automaton). Therefore, a word which alternates long sequences of b's with a single a would get a positive value, while such a word has value 0 according to L_F (it has infinitely many a's). Hence a deterministic LimAvg-automaton cannot exist for L_F.

An analogous argument, and the analysis of the structure of cycles in automata gives the next result.

Theorem 4.12 *Deterministic Büchi automata cannot be weakly reduced to* LimAvg-*automata, and therefore they cannot be reduced to* LimAvg-*automata.*

None of the weighted automata we consider can be reduced to Disc-automata (Theorem 4.13), and Disc-automata cannot be reduced to any of the other classes of weighted automata (Theorem 4.14, and also Theorem 4.10).

Theorem 4.13 *Deterministic coBüchi automata and deterministic Büchi automata cannot be weakly reduced to* Disc-*automata, and therefore they cannot be reduced to* Disc-*automata. Also deterministic* Sup-*automata cannot be reduced to* Disc-*automata.*

The proofs of Theorem 4.13 and 4.14 are based on the property that the value assigned by a Disc-automaton to an infinite word depends essentially on a finite prefix, in the sense that the values of two words become arbitrarily close when they have sufficiently long common prefixes. In other words, the quantitative language defined by a discounted-sum automaton is a continuous function in the Cantor topology. In contrast, for the other classes of weighted automata, the value of an infinite word depends essentially on its tail.

Theorem 4.14 *Deterministic* Disc-*automata cannot be reduced to* LimAvg-*automata.*

Finally, we show that discounted-sum automata cannot be determinized. An example of a nondeterministic discounted-sum automaton N that cannot be determinized is given in Figure 4.6. The automaton N computes the maximum of the discounted sum of a's and b's.

Our results show that N cannot be determinized for discount factors that are rational numbers greater than $\frac{1}{2}$. This result has been refined in [BH11] showing that discounted-sum automata with integer weights can be determinization is only for discount factors of the form $\frac{1}{k}$ with $k \in \mathbb{N}^{\geq 2}$.

Theorem 4.15 Disc-*automata cannot be determinized.*

We have also studied the expressiveness of quantitative languages defined by alternating weighted automata [26], and probabilistic weighted automata [28] with so-called almost-sure and positive semantics. The main result shows that alternation is strictly more expressive than nondeterminism for LimAvg- and Disc-automata. We refer to the corresponding papers for the detailed results.

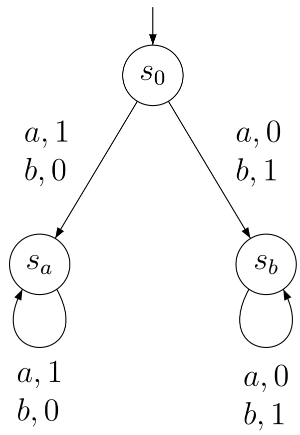


Figure 4.6: The automaton N.

4.5 The Closure Properties of Weighted Automata

We study the closure properties of weighted automata with respect to max, min, complement and sum. We say that a class $\mathcal{C}$ of weighted automata is *closed* under a binary operator $\mathrm{op}(\cdot, \cdot)$ (resp., a unary operator $\mathrm{op}'(\cdot)$) if for all $A_1, A_2 \in \mathcal{C}$, there exists $A_{12} \in \mathcal{C}$ such that $L_{A_{12}} = \mathrm{op}(L_{A_1}, L_{A_2})$ (resp., $L_{A_{12}} = \mathrm{op}'(L_{A_1})$). All closure properties that we present in this chapter are constructive: when $\mathcal{C}$ is closed under an operator, we can always construct the automaton $A_{12} \in \mathcal{C}$ given $A_1, A_2 \in \mathcal{C}$. We say that the *cost* of the closure property of $\mathcal{C}$ under a binary operator op is at most $O(f(n_1, m_1, n_2, m_2))$ if for all automata $A_1, A_2 \in \mathcal{C}$ with n_i states and m_i transitions (for $i = 1, 2$ respectively), the constructed automaton $A_{12} \in \mathcal{C}$ such that $L_{A_{12}} = \mathrm{op}(L_{A_1}, L_{A_2})$ has at most $O(f(n_1, m_1, n_2, m_2))$ many states. Analogously, the *cost* of the closure property of $\mathcal{C}$ under a unary operator op' is at most $O(f(n, m))$ if for all automata $A_1 \in \mathcal{C}$ with n states and m transitions, the constructed automaton $A_{12} \in \mathcal{C}$ such that $L_{A_{12}} = \mathrm{op}'(L_{A_1})$ has at most $O(f(n, m))$ many states. For all reductions presented, the size of the largest weight in A_{12} is linear in the size p of the largest weight in A_1, A_2 (however, the time needed to compute the weights is quadratic in p, as we need addition, multiplication, or comparison, which are quadratic in p).

Notice that every class of weighted automata is closed under shift by c and under scale by $|c|$ for all $c \in \mathbb{Q}$. For Sum-automata and discounted-sum automata, we can define the shift by c by making a copy of the initial states and adding c to the weights of all its outgoing transitions. For the other automata, it suffices to add c to (resp., multiply by $|c|$) all weights of an automaton to obtain the automaton for the shift by c (resp., scale by $|c|$) of its language. Therefore, all closure properties also hold if the complement of a quantitative language L was defined as $k - L$ for any constant k.

Table 4.3 summarizes the closure properties for quantitative languages over infinite words.

	max.	min.	comp.	sum
N/DSUP	✓	✓	×	✓
N/DLINF	✓	✓	×	✓
DLSUP	✓	✓	×	✓
NLSUP	✓	✓	✓	✓
DLAVG	×	×	×	×
NLAVG	✓	×	×	×
DDISC	×	×	✓	✓
NDISC	✓	×	×	✓

Table 4.3: Closure properties. Acronyms are defined in p. 64.

4.5.1 Closure under max

The maximum of two quantitative languages defined by nondeterministic automata can be obtained by an initial nondeterministic choice between the two automata. This observation was also made in [DR07] for discounted-sum automata. For deterministic automata, a synchronized product can be used for Sup and LimSup, while for LimInf we use the fact that NLINF is determinizable with an exponential blow-up [5].

Theorem 4.16 *The nondeterministic* Sup-*,* LimSup-*,* LimInf-*,* LimAvg- *and* Disc-*automata are closed under* max*, with cost* $O(n_1 + n_2)$*, the deterministic* Sup- *and* LimSup-*automata with cost* $O(n_1 \cdot n_2)$*, the deterministic* LimInf-*automata with cost* $O((m_1 + m_2)^{n_1+n_2})$*.*

Theorem 4.17 *The deterministic* LimAvg- *and* Disc-*automata are not closed under* max*.*

The fact that DDISC is not closed under max follows from the proof of Theorem 4.15, where it is shown that the quantitative language $\max(L_1, L_2)$ cannot be defined by a DDISC, where L_1 (resp. L_2) is the language defined by the DDISC that assigns weight 1 (resp., 0) to a's and weight 0 (resp., 1) to b's.

To show that DLAVG is not closed under max, we consider the alphabet $\Sigma = \{a, b\}$ and the quantitative languages L_a and L_b that assign the value of long-run average number of a's and b's, respectively. There exists DLAVG for L_a and L_b, however a careful analysis of the structure of cycles in weighted automata show that $L_m = \max(L_a, L_b)$ cannot be expressed by a DLAVG.

4.5.2 Closure under min

The positive results about closure properties under min for quantitative languages generalize the closure properties of Boolean languages under intersection. The constructions are straightforward extensions of the standard constructions for finite, Büchi, and coBüchi automata (see e.g. [Var96]).

Theorem 4.18 *The (non)deterministic* Sup*-automata are closed under* min*, with cost* $O(n_1 \cdot m_1 \cdot n_2 \cdot m_2)$*, The deterministic* LimSup*-automata are closed under* min *with cost* $O(n_1 \cdot n_2 \cdot 2^{m_1+m_2})$*. The (non)deterministic* LimInf*-automata are closed under* min *with cost* $O(n_1 \cdot n_2)$*, and the nondeterministic* LimSup*-automata with cost* $O(n_1 \cdot n_2 \cdot (m_1 + m_2))$*.*

On the negative side, the (deterministic or not) limit-average and discounted-sum automata are not closed under min.

Theorem 4.19 *The (non)deterministic* LimAvg*-automata are not closed under* min*.*

Finally, using arguments similar to the proof of Theorem 4.15 we show that discounted-sum automata are not closed under min.

Theorem 4.20 *The (non)deterministic* Disc*-automata are not closed under* min*.*

4.5.3 Closure under complement

Most of the weighted automata are not closed under complement. The next results are a direct extension of the Boolean case. The closure under complementation for NLsup is obtained by a reduction to the complementation of nondeterministic Büchi automata.

Theorem 4.21 *The (non)deterministic* Sup- *and* LimInf*-automata, and the deterministic* LimSup*-automata are not closed under complement. The nondeterministic* LimSup*-automata are closed under complement, with cost* $O(m \cdot 2^{n \log n})$*.*

Theorem 4.22 *The deterministic* Disc*-automata are closed under complement, with cost* $O(n)$*. The deterministic* LimAvg*-automata are not closed under complement.*

For Disc-automata, it suffices to replace each weight v by $1 - \lambda - v$ (where λ is the discount factor) to obtain the Disc-automaton for the complement. For LimAvg-automata, an example of a deterministic automaton that cannot be complemented is given in Figure 4.7.

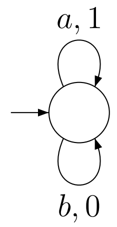


Figure 4.7: Deterministic Limit-average Automaton.

Theorem 4.23 *The nondeterministic* LimAvg- *and* Disc-*automata are not closed under complement.*

The fact that NLavg are not closed under complementation is as follows. Consider the quantitative language $L^* = 1 - \max\{L_a, L_b\}$ where L_a and L_b assign the long-run average number of a's and b's, respectively. Exactly the same argument as in the proof of Theorem 4.19 shows that L^* cannot be expressed as a NLavg, while the language $\max\{L_a, L_b\}$ can be expressed as NLavg by Theorem 4.16.

That NDisc are not closed under complement can be obtained as follows: given $0 < \lambda < 1$, consider the language L_a^λ and L_b^λ that assigns to words the λ-discounted sum of a's and b's, respectively. The language L_a^λ and L_b^λ can be expressed as DDisc, and the max of them can be defined by NDisc. Observe that $L_a^\lambda(w) + L_b^\lambda(w) = \frac{1}{1-\lambda}$ for all $w \in \Sigma^\omega$. Therefore, $\min\{L_a^\lambda, L_b^\lambda\} = \frac{1}{1-\lambda} - \max\{L_a^\lambda, L_b^\lambda\}$. Since NDisc is not closed under min (Theorem 4.20), we immediately obtain that NDisc are not closed under complementation.

4.5.4 Closure under sum

All weighted automata are closed under sum, except DLavg and NLavg. We sketch the construction for LimSup-automata. Given two NLsup A_1 and A_2, we construct a NLsup A for the sum of their languages as follows. Initially, we make a guess of a pair (v_1, v_2) of weights (v_i in A_i, for $i = 1, 2$) and we branch to a copy of the synchronized product of A_1 and A_2. We attach a bit b whose range is $\{1, 2\}$ to each state to remember that we expect A_b to visit the guessed weight v_b. Whenever this occurs, the bit b is set to $3 - b$, and the weight of the transition is $v_1 + v_2$. All other transitions (*i.e.* when b is unchanged) have weight $\min\{v_1 + v_2 \mid v_1 \in V_1 \wedge v_2 \in V_2\}$.

Theorem 4.24 *The (non)deterministic* Sup-*automata are closed under sum, with cost $O(n_1 \cdot m_1 \cdot n_2 \cdot m_2)$. The nondeterministic* LimSup-*automata are closed under sum, with cost $O(n_1 \cdot m_1 \cdot n_2 \cdot m_2)$. The deterministic* LimSup-*automata are closed under sum, with cost $O(n_1 \cdot n_2 \cdot 2^{m_1 \cdot m_2})$. The (non)deterministic* LimInf-*automata are closed under sum with cost $O(n_1 \cdot n_2 \cdot 2^{m_1 \cdot m_2})$.*

It is easy to see that the synchronized product of two NDISC (resp., DDISC) defines the sum of their languages, if the weight of a joint transition is defined as the sum of the weights of the corresponding transitions in the two NDISC (resp., DDISC).

Theorem 4.25 *The (non)deterministic* Disc*-automata are closed under sum, with cost* $O(n_1 \cdot n_2)$.

The analysis of the structure of cycles in LimAvg-automata yields the last result.

Theorem 4.26 *The (non)deterministic* LimAvg*-automata are not closed under sum.*

The closure properties alternating weighted automata [26], and probabilistic weighted automata [28] have been studied, and while alternating automata mostly enjoy positive closure properties, we note that alternating LimAvg-automata are not closed under sum. We refer to the corresponding papers for the detailed results.

4.6 Mean-Payoff Automaton Expressions: A Robust and Decidable Class of Quantitative Languages

In this section, we distinguish between two classes of LimAvg-automata defined using the two *mean-payoff values* of a sequence $\bar{v} = v_0 v_1 \ldots$ of real numbers:

$$\mathsf{LimInfAvg}(\bar{v}) = \liminf_{n\to\infty} \frac{1}{n} \cdot \sum_{i=0}^{n-1} v_i, \text{ or } \quad \mathsf{LimSupAvg}(\bar{v}) = \limsup_{n\to\infty} \frac{1}{n} \cdot \sum_{i=0}^{n-1} v_i.$$

Accordingly, the weighted automata and their quantitative language are called LimInfAvg or LimSupAvg. From now on, we generically use LimAvg-automata to call LimInfAvg- and LimSupAvg-automaton.

A mean-payoff automaton expression expression is either a deterministic LimInfAvg- or LimSupAvg-automaton, or it is the max, min, or sum of two mean-payoff automaton expressions. Since deterministic $\{\mathsf{LimInfAvg}, \mathsf{LimSupAvg}\}$ automata are closed under complement, mean-payoff automaton expressions form a *robust* class that is closed under max, min, sum and complement. We show that (a) all decision problems (quantitative emptiness, universality, inclusion, and equivalence) are *decidable* for mean-payoff automaton expressions; (b) mean-payoff automaton expressions are incomparable in expressive power with both the nondeterministic and alternating LimAvg-automata; and (c) We present the first algorithm to compute the distance between two quantitative languages: we show that the distance can be computed for mean-payoff automaton

	Closure properties				Decision problems			
	max	min	Sum	comp.	empt.	univ.	incl.	equiv.
Deterministic	×	×	×	✓[1]	✓	✓	✓	✓
Nondeterministic	✓	×	×	×	✓	×	×	×
Alternating	✓	✓	×	✓[1]	×	×	×	×
Expressions	✓	✓	✓	✓	✓	✓	✓	✓

Table 4.4: Closure properties and decidability of the various classes of mean-payoff automata. Mean-payoff automaton expressions enjoy fully positive closure and decidability properties.

expressions. When quantitative language inclusion does not hold between an implementation L_A and a specification L_B, the distance is a relevant information to evaluate how far they are from each other, as we would prefer the least expensive one.

Our approach to show decidability of mean-payoff automaton expressions relies on the characterization and algorithmic computation of the *value set* $\{L_E(w) \mid w \in \Sigma^\omega\}$ of an expression E, i.e. the set of all values of words according to E. The value set can be viewed as an abstract representation of the quantitative language L_E, and we show that all decision problems, and distance computation can be solved efficiently once we have this set.

A *mean-payoff automaton expression* E is obtained by the following grammar rule:

$$E ::= A \mid \max(E, E) \mid \min(E, E) \mid \mathsf{Sum}(E, E)$$

where A is a *deterministic* LimInfAvg- or LimSupAvg-automaton. The quantitative language L_E of a mean-payoff automaton expression E is $L_E = L_A$ if $E = A$ is a deterministic automaton, and $L_E = \mathrm{op}(L_{E_1}, L_{E_2})$ if $E = \mathrm{op}(E_1, E_2)$ for op $\in \{\max, \min, \mathsf{Sum}\}$.

4.6.1 Mean-Payoff Automaton Expressions are Robust

By definition, the class of mean-payoff automaton expression is closed under max, min and Sum. Closure under complement follows from the fact that the complement of $\max(E_1, E_2)$ is $\min(-E_1, -E_2)$, the complement of $\min(E_1, E_2)$ is $\max(-E_1, -E_2)$, the complement of $\mathsf{Sum}(E_1, E_2)$ is $\mathsf{Sum}(-E_1, -E_2)$, and the complement of a deterministic LimInfAvg-automaton can be defined by the same automaton with opposite weights and interpreted as a LimSupAvg-automaton, and vice versa, since $-\limsup(v_0, v_1, \dots) = \liminf(-v_0, -v_1, \dots)$. Note that arbitrary linear combinations of deterministic mean-payoff automaton expressions (expressions such as $c_1 E_1 + c_2 E_2$ where $c_1, c_2 \in \mathbb{Q}$ are

[1] Closure under complementation holds because LimInfAvg-automata and LimSupAvg-automata are dual. It would not hold if only LimInfAvg-automata (or only LimSupAvg-automata) were allowed.

rational constants) can be obtained for free since scaling the weights of a LimAvg-automaton by a positive factor $|c|$ results in a quantitative language scaled by the same factor. Therefore, we say that mean-payoff automaton expressions are *robust*.

Expressive power comparison. We compare the expressive power of mean-payoff automaton expressions with nondeterministic and alternating LimAvg-automata. The results of [27] show that there exist deterministic LimAvg-automata A_1 and A_2 such that $\min(A_1, A_2)$ cannot be expressed by nondeterministic LimAvg-automata. The results of [26] show that there exist deterministic LimAvg-automata A_1 and A_2 such that $\mathsf{Sum}(A_1, A_2)$ cannot be expressed by alternating LimAvg-automata. It follows that there exist languages expressible by mean-payoff automaton expression that cannot be expressed by nondeterministic and alternating LimAvg-automata. In Theorem 4.27 we show the converse, that is, we show that there exist languages expressible by nondeterministic LimAvg-automata that cannot be expressed by mean-payoff automaton expression. It may be noted that the subclass of mean-payoff automaton expressions that only uses min and max operators (and no sum operator) is a strict subclass of alternating LimAvg-automata, and when only the max operator is used we get a strict subclass of the nondeterministic LimAvg-automata.

Theorem 4.27 *Mean-payoff automaton expressions are incomparable in expressive power with nondeterministic and alternating* LimAvg*-automata: (a) there exists a quantitative language that is expressible by mean-payoff automaton expressions, but cannot be expressed by an alternating* LimAvg*-automaton; and (b) there exists a quantitative language that is expressible by a nondeterministic* LimAvg*-automaton, but cannot be expressed by a mean-payoff automaton expression.*

4.6.2 Mean-Payoff Automaton Expressions are Decidable

Given a mean-payoff automaton expression E, let $A_1, \dots, A_n$ be the deterministic weighted automata occurring in E. The *vector set* of E is the set

$$V_E = \{\langle L_{A_1}(w), \dots, L_{A_n}(w)\rangle \in \mathbb{R}^n \mid w \in \Sigma^\omega\}$$

of tuples of values of words according to each automaton A_i. In this section, we characterize the vector set of mean-payoff automaton expressions, and in Section 4.6.2 we give an algorithmic procedure to compute this set. This will be useful to establish the decidability of all decision problems, and to compute the distance between mean-payoff automaton expressions. Given a vector $v \in \mathbb{R}^n$, we denote by $\|v\| = \max_i |v_i|$ the ∞-*norm* of v.

The *synchronized product* of $A_1, \dots, A_n$ such that $A_i = \langle Q_i, q_I^i, \Sigma, \delta_i, \gamma_i\rangle$ is the $\mathbb{Q}^n$-weighted automaton $A_E = A_1 \times \dots \times A_n = \langle Q_1 \times \dots \times Q_n, (q_I^1, \dots, q_I^n), \Sigma, \delta, \gamma\rangle$ such that $t = ((q_1, \dots, q_n), \sigma, (q'_1, \dots, q'_n)) \in \delta$ if $t_i := (q_i, \sigma, q'_i) \in \delta_i$ for all $1 \le i \le n$, and

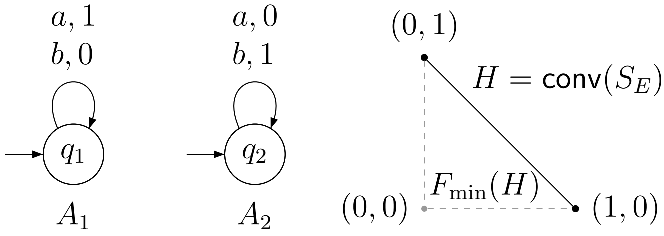


Figure 4.8: The vector set of $E = \max(A_1, A_2)$ is $F_{\min}(\mathsf{conv}(S_E)) \supsetneq \mathsf{conv}(S_E)$.

$\gamma(t) = (\gamma_1(t_1), \ldots, \gamma_n(t_n))$. In the sequel, we assume that all A_i's are deterministic LimInfAvg-automata (hence, A_E is deterministic) and that the underlying graph of the automaton A_E has only one strongly connected component (scc). The general results for automata with both deterministic LimInfAvg- and LimSupAvg automata, and with multi-scc underlying graph are given in [19].

For each (simple) cycle ρ in A_E, let the *vector value* of ρ be the mean of the tuples labelling the edges of ρ, denoted $\mathsf{Avg}(\rho)$. To each simple cycle ρ in A_E corresponds a (not necessarily simple) cycle in each A_i, and the vector value $(v_1, \ldots, v_n)$ of ρ contains the mean value v_i of ρ in each A_i. We denote by S_E the (finite) set of vector values of simple cycles in A_E. Let $\mathsf{conv}(S_E)$ be the convex hull of S_E.

Lemma 4.28 *Let E be a mean-payoff automaton expression. The set $\mathsf{conv}(S_E)$ is the closure of the set $\{L_E(w) \mid w \text{ is a lasso-word}\}$.*

The vector set of E contains more values than the convex hull $\mathsf{conv}(S_E)$, as shown by the following example.

Example 4.3 *Consider the expression $E = \max(A_1, A_2)$ where A_1 and A_2 are deterministic* LimInfAvg*-automata (see Figure 4.8). The product $A_E = A_1 \times A_2$ has two simple cycles with respective vector values $(1, 0)$ (on letter 'a') and $(0, 1)$ (on letter 'b'). The set $H = \mathsf{conv}(S_E)$ is the solid segment on Figure 4.8 and contains the vector values of all lasso-words. However, other vector values can be obtained: consider the word $w = a^{n_1} b^{n_2} a^{n_3} b^{n_4} \ldots$ where $n_1 = 1$ and $n_{i+1} = (n_1 + \cdots + n_i)^2$ for all $i \geq 1$. It is easy to see that the value of w according to A_1 is 0 because the average number of a's in the prefixes $a^{n_1} b^{n_2} \ldots a^{n_i} b^{n_{i+1}}$ for i odd is smaller than $\frac{n_1 + \cdots + n_i}{n_1 + \cdots + n_i + n_{i+1}} = \frac{1}{1 + n_1 + \cdots + n_i}$ which tends to 0 when $i \to \infty$. Since A_1 is a* LimInfAvg*-automaton, the value of w is 0 in A_1, and by a symmetric argument the value of w is also 0 in A_2. Therefore the vector $(0, 0)$ is in the vector set of E. Note that $z = (z_1, z_2) = (0, 0)$ is the pointwise minimum of $x = (x_1, x_2) = (1, 0)$ and $y = (y_1, y_2) = (0, 1)$, i.e. $z = f_{\min}(x, y)$ where $z_1 = \min(x_1, y_1)$ and $z_2 = \min(y_1, y_2)$. In fact, the vector set is the whole triangular region in Figure 4.8, i.e. $V_E = \{f_{\min}(x, y) \mid x, y \in \mathsf{conv}(S_E)\}$.* ■

We generalize $f_{\min}$ to finite sets of points $P \subseteq \mathbb{R}^n$ in n dimensions as follows: $f_{\min}(P) \in \mathbb{R}^n$ is the point $p = (p_1, p_2, \ldots, p_n)$ such that p_i is the minimum i^{th} coordinate of the points in P, for $1 \leq i \leq n$. For arbitrary $S \subseteq \mathbb{R}^n$, define $F_{\min}(S) = \{f_{\min}(P) \mid P \text{ is a finite subset of } S\}$. As illustrated in Example 4.3, the next lemma shows that the vector set V_E is equal to $F_{\min}(\mathsf{conv}(S_E))$.

Lemma 4.29 *Let E be a mean-payoff automaton expression built from deterministic* LimInfAvg*-automata, and such that A_E has only one strongly connected component. Then, the vector set of E is $V_E = F_{\min}(\mathsf{conv}(S_E))$.*

Computation of $F_{\min}(\mathsf{conv}(S))$ for a Finite Set S. For a finite set $S \subseteq \mathbb{R}^n$, the set $\mathsf{conv}(S)$ is in general infinite, thus it is not immediate that $F_{\min}(\mathsf{conv}(S))$ is computable. We first present some properties of the set $F_{\min}(\mathsf{conv}(S))$.

Lemma 4.30 *If X is a convex set, then $F_{\min}(X)$ is convex.*

By Lemma 4.30, the set $F_{\min}(\mathsf{conv}(S))$ is convex, and since $F_{\min}$ is a monotone operator and $S \subseteq \mathsf{conv}(S)$, we have $F_{\min}(S) \subseteq F_{\min}(\mathsf{conv}(S))$ and thus $\mathsf{conv}(F_{\min}(S)) \subseteq F_{\min}(\mathsf{conv}(S))$. It turns out that equality holds in two dimensions, i.e. $\mathsf{conv}(F_{\min}(S)) = F_{\min}(\mathsf{conv}(S))$ for all finite sets $S \subseteq \mathbb{R}^2$.

We show in the following example that in three dimensions this equality does not hold, i.e., we show that $F_{\min}(\mathsf{conv}(S_E)) \neq \mathsf{conv}(F_{\min}(S_E))$ in $\mathbb{R}^3$.

Example 4.4 *We show that in three dimension there is a finite set S such that $F_{\min}(\mathsf{conv}(S)) \not\subseteq \mathsf{conv}(F_{\min}(S))$. Let $S = \{q, r, s\}$ with $q = (0, 1, 0)$, $r = (-1, -1, 1)$, and $s = (1, 1, 1)$. Then $f_{\min}(r, s) = r$, $f_{\min}(q, r, s) = f_{\min}(q, r) = t = (-1, -1, 0)$, and $f_{\min}(q, s) = q$. Therefore $F_{\min}(S) = \{q, r, s, t\}$. Consider $p = (r + s)/2 = (0, 0, 1)$. We have $p \in \mathsf{conv}(S)$ and $f_{\min}(p, q) = (0, 0, 0)$. Hence $(0, 0, 0) \in F_{\min}(\mathsf{conv}(S))$. We now show that $(0, 0, 0)$ does not belong to $\mathsf{conv}(F_{\min}(S))$. Consider $u = \alpha_q \cdot q + \alpha_r \cdot r + \alpha_s \cdot s + \alpha_t \cdot t$ such that u in $\mathsf{conv}(F_{\min}(S))$. Since the third coordinate is non-negative for q, r, s, and t, it follows that if $\alpha_r > 0$ or $\alpha_s > 0$, then the third coordinate of u is positive. If $\alpha_s = 0$ and $\alpha_r = 0$, then we have two cases: (a) if $\alpha_t > 0$, then the first coordinate of u is negative; and (b) if $\alpha_t = 0$, then the second coordinate of u is 1. It follows $(0, 0, 0)$ is not in $\mathsf{conv}(F_{\min}(S))$.*

Example 4.4 shows that in general $F_{\min}(\mathsf{conv}(S)) \not\subseteq \mathsf{conv}(F_{\min}(S))$. In this section we present an explicit construction of $F_{\min}(\mathsf{conv}(S))$ as a finite set of linear constraints.

Consider the *positive orthant* anchored at the origin in $\mathbb{R}^n$, that is, the set of points with non-negative coordinates: $\mathbb{R}^n_+ = \{(z_1, z_2, \ldots, z_n) \mid z_i \geq 0, \forall i\}$. Similarly, the *negative orthant* is the set of points with non-positive coordinates, denoted as $\mathbb{R}^n_- = -\mathbb{R}^n_+$. Using vector addition, we write $y + \mathbb{R}^n_+$ for the positive orthant anchored at y. Similarly, we write $x + \mathbb{R}^n_- = x - \mathbb{R}^n_+$ for the negative orthant anchored at x. The positive and negative orthants satisfy the following simple *duality relation*: $x \in y + \mathbb{R}^n_+$ iff $y \in x - \mathbb{R}^n_+$.

Note that $\mathbb{R}^n_+$ is an n-dimensional convex polyhedron. For each $1 \leq j \leq n$, we consider the $(n-1)$-dimensional face $\mathbb{L}_j$ spanned by the coordinate axes except the j^{th} one, that is, $\mathbb{L}_j = \{(z_1, z_2, \ldots, z_n) \in \mathbb{R}^n_+ \mid z_j = 0\}$.

We say that $y + \mathbb{R}^n_+$ is *supported* by X if $(y + \mathbb{L}_j) \cap X \neq \varnothing$ for every $1 \leq j \leq n$. Assuming $y + \mathbb{R}^n_+$ is supported by X, we can construct a set $Y \subseteq X$ by collecting one point per $(n-1)$-dimensional face of the orthant and get $y = f(Y)$. It is also allowed that two faces contribute the same point to Y. Similarly, if $y = f(Y)$ for a subset $Y \subseteq X$, then the positive orthant anchored at y is supported by X. Hence, we get the following lemma.

Lemma 4.31 (Orthant Lemma) *$y \in F_{\min}(X)$ iff $y + \mathbb{R}^n_+$ is supported by X.*

Construction. We use the Orthant Lemma to construct $F_{\min}(X)$. We begin by describing the set of points y for which the j^{th} face of the positive orthant anchored at y has a non-empty intersection with X. Define $F_j = X - \mathbb{L}_j$, the set of points of the form $x - z$, where $x \in X$ and $z \in \mathbb{L}_j$.

Lemma 4.32 (Face Lemma) *$(y + \mathbb{L}_j) \cap X \neq \varnothing$ iff $y \in F_j$.*

It is now easy to describe the set defined in our problem statement.

Lemma 4.33 (Characterization) *$F_{\min}(X) = \bigcap_{j=1}^n F_j$.*

Algorithm for computation of $F_{\min}(\mathsf{conv}(S))$. Following the construction, we get an algorithm that computes $F_{\min}(\mathsf{conv}(S))$ for a finite set S of points in $\mathbb{R}^n$. Let $|S| = m$. We first represent $X = \mathsf{conv}(S)$ as intersection of half-spaces: we require at most m^n half-spaces (linear constraints). It follows that $F_j = X - \mathbb{L}_j$ can be expressed as m^n linear constraints, and hence $F_{\min}(X) = \bigcap_{j=1}^n F_j$ can be expressed as $n \cdot m^n$ linear constraints. This gives us the following result.

Theorem 4.34 *Given a finite set S of m points in $\mathbb{R}^n$, we can construct in $O(n \cdot m^n)$ time $n \cdot m^n$ linear constraints that represent $F_{\min}(\mathsf{conv}(S))$.*

Decision problems and distance. Several problems on quantitative languages can be solved for the class of mean-payoff automaton expressions using the vector set. The decision problems of quantitative emptiness and universality, and quantitative language inclusion and equivalence are all decidable, as well as computing the distance between mean-payoff languages.

Complexity. All problems studied in this section can be solved easily (in polynomial time) once the value set is constructed, which can be done in quadruple exponential time. The quadruple exponential blow-up is caused by (a) the synchronized product construction for E, (b) the computation of the vector values of all simple cycles in A_E, (c) the construction of the vector set $F_{\min}(\mathsf{conv}(S_E))$, and ($d$) the successive projections of the vector set to obtain the value set. Therefore, all the above problems can be solved in 4EXPTIME.

Theorem 4.35 *For the class of mean-payoff automaton expressions, the quantitative emptiness, universality, language inclusion, and equivalence problems, as well as distance computation can be solved in 4EXPTIME.*

Theorem 4.35 is in sharp contrast with the nondeterministic and alternating mean-payoff automata for which language inclusion is undecidable (see also Table 4.4).

4.7 Conclusion and Perspectives

We have proposed a quantitative generalization of Boolean languages where the value of a trace is computed as the supremum, limsup, liminf, limit-average, or discounted sum of the weights of a trace. We have defined and studied a quantitative version of the classical decision problems (emptiness, universality, language inclusion and equivalence) and exhaustively compared the expressive power and closure properties of the various classes of quantitative languages.

We have presented a new class of quantitative languages, the *mean-payoff automaton expressions* which are both robust and decidable (see Table 4.4), and for which the distance between quantitative languages can be computed. The decidability results come with a high worst-case complexity, and it is a natural question for future works to improve the algorithmic solution. The unpublished results of [Vel11] suggest that the problem is PSPACE-complete.

We identify the following important research directions:

- finding a robust and decidable class of quantitative languages that would subsume discounted-sum languages is completely open. It would be of great interest to understand whether such a class exists, or what are the reasons that make the discounted sum behave differently as compared to the mean-payoff measure. Most likely, an important first step in this direction would be to find an algorithmic solution to discounted sum multi-weighted games, i.e. to solve games with Boolean combination of objectives defined by discounted sum.

- while the results presented in this chapter solve the model-checking and distance problems, a more challenging question is the synthesis problem for mean-payoff automaton expressions. This question involves games and may be hard to solve. Finding subclasses of mean-payoff automaton expressions (e.g., without the Σ operator) may lead to interesting investigations.

- a more general and long-term objective is to describe canonical ways of defining quantitative languages, as well as decomposition of quantitative languages into simple building blocks, with the objective of providing a compositional (or hierarchical) theory of such languages, in analogy to the theory of ω-regular languages where for instance a classification such as the Borel hierarchy, or the decomposition of any language into safety and liveness languages are fundamental for a better understanding of the theory. In this context, it will be useful to explore logical characterizations of quantitative languages.

Chapter 5

Conclusion

Chanter, rêver, rire, passer, être seul, être libre,
Avoir l'œil qui regarde bien, la voix qui vibre.

Edmond Rostand, *Cyrano de Bergerac*.

5.1 Summary

The results presented in this thesis follow three main lines of research on *antichain algorithms* for efficient analysis of finite-state models, on the complexity and algorithmic solution of *quantitative games*, and on the development of new formalisms to specify *quantitative languages*. Each chapter contains a summary of the contributions, as well as some research directions for the future. Below, we select and emphasize for each chapter one outstanding result.

- Dramatic performance improvements have been obtained by the antichain algorithms of Chapter 2 for several decision problems of automata theory, for example deciding universality of Büchi automata for automata of size ten times larger as before [6]. Other teams are applying and extending the technique [FJR09, FV10, ACH$^+$10, ACC$^+$10, LGG11].
- An important result of Chapter 3 is that deciding the winner of energy parity games can be solved in NP $\cap$ coNP while exponential memory may be required for winning strategies. Mean-payoff parity games can also be solved in NP $\cap$ coNP.
- The mean-payoff automaton expressions presented in Chapter 4 are the first robust and decidable class of quantitative languages (they are closed under the four pointwise operations, and the four decision problems are decidable).

As a conclusion and summary, it turns out that the technique of antichain algorithms has several applications that go beyond the original problem for which it was designed (see Chapter 2), and that the study of the emerging energy games provides new insights and new results for the tightly connected and well-established mean-payoff games.

5.2 Perspectives

Promising research directions have been outlined at the end of each chapter. From a broader perspective, we identify the following themes that we plan to investigate.

Partial-observation stochastic games. In several applications of infinite games in computer science, it is more realistic to drop the assumption of *perfect observation* (i.e., both players have perfect observation about the state of the game). For example in the context of hybrid systems, the controller acquires information about the state of a plant using digital sensors with finite precision, which gives partial information about the state of the plant [39, HK99]. Similarly, in a concurrent system where the players represent individual processes, each process has only access to the public variables of the other processes, not to their private variables [Rei84, AHK02]. Such problems are better modeled in the more general framework of *partial-observation* games [Rei84, 9, BGG09] and have been studied in the context of verification and synthesis [RP80, KV00].

As we mention in the introduction of this thesis, the study of partial-observation games is an active area of research. From a theoretical point of view, partial-observation games have tight connections with automata. In particular, solving a blind game (where one player cannot distinguish any state) is equivalent to solving universality questions on automata. Similarly, partial-observation *stochastic* games have connections with probabilistic automata. While the complexity and algorithms for automata have been extensively studied, the theory of partial-observation (stochastic) games has been little studied and several questions have not received all the attention they deserve. The short-term aim of our research in this area is to close complexity gaps for decision problems, and to identify classes of partial-observation games with lower complexity. From a practical point of view, the applications in program synthesis (such as lock-placement in concurrent programs) require efficient algorithms. Techniques related to abstraction and symbolic computation need to be developed to make possible the analysis of cases of practical interest.

Counter systems. The quantitative games and quantitative languages can be viewed as applications of counter systems. While the energy objective is a conceptually simple condition, a wealth of results have been obtained for energy games. This may sound surprising since not so many interesting classes of games on counter systems are decidable. In the same way, the resulting advances in the complexity and decidability of mean-payoff games are unexpected since mean-payoff games have been studied for so long.

From this experience, we conclude that more investigations of counter systems are necessary, first concerning games on counter systems which have received little attention due to many undecidability results [ABd08, RSB05], and also concerning the quantitative theory of verification. The results on quantitative games and quantitative languages suggest to study new classes of counter systems (e.g., with discounting, or with new operations such as averaging, or division by a constant) to establish connections with known or open problems. As a most promising direction, we will investigate new game models on counter systems and the recent developments in the theory of cost functions and distance automata [CL08, Col09, CKL10].

The other direction of research is the systematic study of games played on the various classes of counter systems. For example, with Alexander Rabinovich we have identified a simple class of games, the *robot games*, for which the decidability status is not known. Given two finite sets $U, V \subseteq \mathbb{Z}^2$ of two-dimensional integer vectors, a *robot game* is played in rounds from an initial configuration $x_0 \in \mathbb{Z}^2$ as follows. In each round, player 1 chooses a vector $u \in U$, then player 2 chooses a vector $v \in V$, and the configuration in the next round is $x + u + v$ where x is the configuration in the current round. The objective of player 1 is to reach the configuration $(0, 0)$. Of apparent extreme simplicity, the problem of deciding the existence of a winning strategy in robot games is not solved. Extensions to higher dimensions, to more complex objectives, or players with fixed internal state are interesting as well.

Classification of quantitative languages. The perspectives presented at the end of Chapter 4 are fascinating, such as finding a robust and decidable class of quantitative languages that subsumes discounted-sum languages, or solving games with objective defined by a quantitative language. In parallel, the development of verification tools based on the quantitative approach to model-checking is still missing, mainly due to the lack of convincing and appropriate data-structure to deal with weights.

Our main long-term objective in the area of quantitative verification is to identify canonical ways of defining quantitative languages and to construct an elegant theory of quantitative languages that would induce a convincing hierarchy of languages (such as the Borel hierarchy for Boolean languages), an hierarchy of weighted machines of increasing complexity (such as the finite automata, pushdown automata, etc. for non-weighted machines), robust properties (closure properties, equivalence with other formalisms such as logics), and of course include simple class of quantitative languages such as mean-payoff and discounted-sum languages.

Bibliography

[ABd08] P. A. Abdulla, A. Bouajjani, and J. d'Orso. Monotonic and downward closed games. *J. Log. Comput.*, 18(1):153–169, 2008.

[ACC+10] P. A. Abdulla, Y.-F. Chen, L. Clemente, L. Holík, C.-D. Hong, R. Mayr, and T. Vojnar. Simulation subsumption in Ramsey-based Büchi automata universality and inclusion testing. In *Proc. of CAV: Computer Aided Verification*, LNCS 6174, pages 132–147. Springer-Verlag, 2010.

[ACC+11] P. A. Abdulla, Y.-F. Chen, L. Clemente, L. Holík, C.-D. Hong, R. Mayr, and T. Vojnar. Advanced Ramsey-based Büchi automata inclusion testing. In *Proc. of CONCUR: Concurrency Theory*, LNCS 6901, pages 187–202. Springer-Verlag, 2011.

[ACH+10] P. A. Abdulla, Y.-F. Chen, L. Holík, R. Mayr, and T. Vojnar. When simulation meets antichains. In *Proc. of TACAS: Tools and Algorithms for the Construction and Analysis of Systems*, LNCS 6015, pages 158–174. Springer-Verlag, 2010.

[ACJT96] P. A. Abdulla, K. Cerans, B. Jonsson, and Y.-K. Tsay. General decidability theorems for infinite-state systems. In *Proc. of LICS: Logic in Computer Science*, pages 313–321, 1996.

[ADMW09] R. Alur, A. Degorre, O. Maler, and G. Weiss. On omega-languages defined by mean-payoff conditions. In *Proc. of FOSSACS*, LNCS 5504, pages 333–347. Springer-Verlag, 2009.

[AHK02] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49:672–713, 2002.

[AHKV98] R. Alur, T. A. Henzinger, O. Kupferman, and M.Y. Vardi. Alternating refinement relations. In *Proc. of CONCUR: Concurrency Theory*, LNCS 1466, pages 163–178. Springer-Verlag, 1998.

[And06] D. Andersson. An improved algorithm for discounted payoff games. In *ESSLLI Student Session*, pages 91–98, 2006.

[BCHJ09] R. Bloem, K. Chatterjee, T. A. Henzinger, and B. Jobstmann. Better quality in synthesis through quantitative objectives. In *Proc. of CAV: Computer-Aided Verification*, LNCS 5643, pages 140–156. Springer-Verlag, 2009.

[BFL+08] P. Bouyer, U. Fahrenberg, K. G. Larsen, N. Markey, and J. Srba. Infinite runs in weighted timed automata with energy constraints. In *Proc. of FORMATS: Formal Modeling and Analysis of Timed Systems*, LNCS 5215, pages 33–47. Springer-Verlag, 2008.

[BGG09] N. Bertrand, B. Genest, and H. Gimbert. Qualitative determinacy and decidability of stochastic games with signals. In *Proc. of LICS: Logic in Computer Science*, pages 319–328. IEEE Computer Society, 2009.

[BH11] U. Boker and T. A. Henzinger. Determinizing discounted-sum automata. In *Proc. of CSL: Computer Science Logic*, volume 12 of *LIPIcs*, pages 82–96. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011.

[BHH+08] A. Bouajjani, P. Habermehl, L. Holík, T. Touili, and T. Vojnar. Antichain-based universality and inclusion testing over nondeterministic finite tree automata. In *Proc. of CIAA: Implementation and Applications of Automata*, LNCS 5148, pages 57–67. Springer-Verlag, 2008.

[BJK10] T. Brázdil, P. Jancar, and A. Kucera. Reachability games on extended vector addition systems with states. In *Proc. of ICALP: Automata, Languages and Programming*, LNCS 6199, pages 478–489. Springer-Verlag, 2010.

[BL69] J. R. Büchi and L. H. Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the Amer. Math. Soc.*, 138:295311, 1969.

[BSV04] H. Björklund, S. Sandberg, and S. G. Vorobyov. Memoryless determinacy of parity and mean payoff games: a simple proof. *Theor. Comput. Sci.*, 310(1-3):365–378, 2004.

[BV07] H. Björklund and S. G. Vorobyov. A combinatorial strongly subexponential strategy improvement algorithm for mean payoff games. *Discrete Applied Mathematics*, 155:210–229, 2007.

[CCGR00] A. Cimatti, E. Clarke, F. Giunchiglia, and M. Roveri. Nusmv: A new symbolic model checker. *STTT: Software Tools for Technology Transfer*, 2(4):410–425, 2000.

[CCH+05] A. Chakrabarti, K. Chatterjee, T. A. Henzinger, O. Kupferman, and R. Majumdar. Verifying quantitative properties using bound functions. In *Proc. of CHARME: Correct Hardware Design and Verification Methods*, LNCS 3725, pages 50–64. Springer-Verlag, 2005.

[CdF+06] K. Chatterjee, L. de Alfaro, M. Faella, T. A. Henzinger, R. Majumdar, and M. Stoelinga. Compositional quantitative reasoning. In *QEST: Quantitative Evaluation of Systems*, pages 179–188. IEEE Computer Society, 2006.

[CdHS03] A. Chakrabarti, L. de Alfaro, T. A. Henzinger, and M. Stoelinga. Resource interfaces. In *Proc. of EMSOFT: Embedded Software*, LNCS 2855, pages 117–133. Springer-Verlag, 2003.

[CGH+08] K. Chatterjee, A. Ghosal, T. A. Henzinger, D. Iercan, C. Kirsch, C. Pinello, and A. Sangiovanni-Vincentelli. Logical reliability of interacting real-time tasks. In *Proc. of DATE: Design, Automation and Test*, pages 909–914. ACM, 2008.

[Cha07] K. Chatterjee. Concurrent games with tail objectives. *Theoretical Computer Science*, 388:181–198, 2007.

[Cha10] J. Chaloupka. Z-reachability problem for games on 2-dimensional vector addition systems with states is in P. In *Proc. of RP: Reachability Problems*, LNCS 6227, pages 104–119. Springer-Verlag, 2010.

[CHJ05] K. Chatterjee, T. A. Henzinger, and M. Jurdziński. Mean-payoff parity games. In *Proc. of LICS: Logic in Computer Science*, pages 178–187. IEEE Computer Society, 2005.

[Chu62] A. Church. Logic, arithmetic, and automata. In *Proc. of the International Congress of Mathematicians*, page 2335. Institut Mittag-Leffler, 1962.

[CK94] K. Culik and J. Karhumäki. Finite automata computing real functions. *SIAM J. Computing*, 23:789–814, 1994.

[CKL10] T. Colcombet, D. Kuperberg, and S. Lombardy. Regular temporal cost functions. In *Proc. of ICALP: Automata, Languages and Programming*, LNCS 6199, pages 563–574. Springer-Verlag, 2010.

[CL08] T. Colcombet and C. Löding. The non-deterministic mostowski hierarchy and distance-parity automata. In *Proc. of ICALP: Automata, Languages and Programming*, LNCS 5126, pages 398–409. Springer-Verlag, 2008.

[Col09] T. Colcombet. The theory of stabilisation monoids and regular cost functions. In *Proc. of ICALP: Automata, Languages and Programming*, LNCS 5556, pages 139–150. Springer-Verlag, 2009.

[Con92] A. Condon. The complexity of stochastic games. *Information and Computation*, 96:203–224, 1992.

[Con93] A. Condon. On algorithms for simple stochastic games. In *Advances in Computational Complexity Theory*, volume 13 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 51–73. American Mathematical Society, 1993.

[dAFS04] L. de Alfaro, M. Faella, and M. Stoelinga. Linear and branching metrics for quantitative transition systems. In *Proc. of ICALP: Automata, Languages and Programming*, LNCS 3142, pages 97–109. Springer-Verlag, 2004.

[dAH01a] L. de Alfaro and T. A. Henzinger. Interface automata. In *Proc. of FSE: Foundations of Software Engineering*, pages 109–120. ACM Press, 2001.

[dAH01b] L. de Alfaro and T. A. Henzinger. Interface theories for component-based design. In *Proc. of EMSOFT: Embedded Software*, volume 2211 of *LNCS*, pages 148–165. Springer-Verlag, 2001.

[dAHM03] L. de Alfaro, T. A. Henzinger, and R. Majumdar. Discounting the future in systems theory. In *Proc. of ICALP: International Colloquium on Automata, Languages and Programming*, LNCS 2719, pages 1022–1037. Springer-Verlag, 2003.

[DG06] V. Dhingra and S. Gaubert. How to solve large scale deterministic games with mean payoff by policy iteration. In *Proc. Performance evaluation methodolgies and tools, art. 12*. ACM, 2006.

[DG07] M. Droste and P. Gastin. Weighted automata and weighted logics. *Theoretical Computer Science*, 380:69–86, 2007.

[DGJP99] J. Desharnais, V. Gupta, R. Jagadeesan, and P. Panangaden. Metrics for labeled Markov systems. In *Proc. of CONCUR: Concurrency Theory*, LNCS 1664, pages 258–273. Springer-Verlag, 1999.

[DGJP03] J. Desharnais, V. Gupta, R. Jagadeesan, and P. Panangaden. Approximating labelled Markov processes. *Inf. Comput.*, 184(1):160–200, 2003.

[Dic13] L. E. Dickson. Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors. *Am. J. of Mathematics*, 35(4):413–422, 1913.

[DK03] M. Droste and D. Kuske. Skew and infinitary formal power series. In *Proc. of ICALP: International Colloquium on Automata, Languages and Programming*, LNCS 2719, pages 426–438. Springer-Verlag, 2003.

[DKR08] M. Droste, W. Kuich, and G. Rahonis. Multi-valued MSO logics over words and trees. *Fundamenta Informaticae*, 84:305–327, 2008.

[DR07] M. Droste and G. Rahonis. Weighted automata and weighted logics with discounting. In *Proc. of CIAA: Implementation and Application of Automata*, LNCS 4783, pages 73–84. Springer-Verlag, 2007.

[DZL03] S. Dal-Zilio and D. Lugiez. XML schema, tree logic and sheaves automata. In *Proc. of RTA: Rewriting Techniques and Applications*, pages 246–263, 2003.

[Ehl10] R. Ehlers. Symbolic bounded synthesis. In *Proc. of CAV: Computer Aided Verification*, LNCS 6174, pages 365–379. Springer-Verlag, 2010.

[EJ91] E. A. Emerson and C. Jutla. Tree automata, mu-calculus and determinacy. In *Proc. of FOCS: Foundations of Computer Science*, pages 368–377. IEEE, 1991.

[EJS93] E. A. Emerson, C. Jutla, and A. P. Sistla. On model checking for fragments of the μ-calculus. In *Proc. of CAV: Computer Aided Verification*, LNCS 697, pages 385–396. Springer-Verlag, 1993.

[ÉK04] Z. Ésik and W. Kuich. An algebraic generalization of omega-regular languages. In *Proc. of MFCS: Mathematical Foundations of Computer Science*, LNCS 3153, pages 648–659. Springer-Verlag, 2004.

[EM79] A. Ehrenfeucht and J. Mycielski. Positional strategies for mean payoff games. *International Journal of Game Theory*, 8:109–113, 1979.

[FJR09] E. Filiot, N. Jin, and J.-F. Raskin. An antichain algorithm for LTL realizability. In *Proc. of CAV: Computer Aided Verification*, LNCS 5643, pages 263–277. Springer-Verlag, 2009.

[FJR10] E. Filiot, N. Jin, and J.-F. Raskin. Compositional algorithms for LTL synthesis. In *Proc. of ATVA: Automated Technology for Verification and Analysis*, LNCS 6252, pages 112–127. Springer-Verlag, 2010.

[FS01] A. Finkel and P. Schnoebelen. Well-structured transition systems everywhere! *Theor. Comput. Sci.*, 256(1-2):63–92, 2001.

[FV97] J. Filar and K. Vrieze. *Competitive Markov Decision Processes*. Springer-Verlag, 1997.

[FV09] S. Fogarty and M. Y. Vardi. Büchi complementation and size-change termination. In *Proc. of TACAS: Tools and Algorithms for the Construction and Analysis of Systems*, volume 5505 of *LNCS*, pages 16–30. Springer-Verlag, 2009.

[FV10] S. Fogarty and M. Y. Vardi. Efficient Büchi universality checking. In *Proc. of TACAS: Tools and Algorithms for the Construction and Analysis of Systems*, LNCS 6015, pages 205–220. Springer-Verlag, 2010.

[GC03] A. Gurfinkel and M. Chechik. Multi-valued model checking via classical model checking. In *Proc. of CONCUR: Concurrency Theory*, LNCS 2761, pages 263–277. Springer-Verlag, 2003.

[GH82] Y. Gurevich and L. Harrington. Trees, automata, and games. In *Proc. of STOC: Symposium on Theory of Computing*, pages 60–65. ACM, 1982.

[Gim06] H. Gimbert. *Jeux positionnels*. PhD thesis, Universitég Paris 7, 2006.

[GKK88] V. A. Gurvich, A. V. Karzanov, and L. G. Kachiyan. USSR computational mathematics and mathematical physics. *Cyclic Games and an Algorithm to Find Minmax Cycle Means in Directed Graphs*, 5(28):85–91, 1988.

[GKSV03] S. Gurumurthy, O. Kupferman, F. Somenzi, and M. Y. Vardi. On complementing nondeterministic Büchi automata. In *Proc. of CHARME: Correct Hardware Design and Verification Methods*, LNCS 2860, pages 96–110. Springer-Verlag, 2003.

[GNT04] E. Giunchiglia, M. Narizzano, and A. Tacchella. QuBE++: An Efficient QBF Solver. In *Proc. of FMCAD: Formal Methods in Computer-Aided Design*, LNCS 3312, pages 201–213. Springer-Verlag, 2004.

[GTW02] E. Grädel, W. Thomas, and T. Wilke, editors. *Automata, Logics, and Infinite Games*, LNCS 2500. Springer-Verlag, 2002.

[Hen10] T. A. Henzinger. From boolean to quantitative notions of correctness. In *Proc. of POPL: Principles of Programming Languages*, pages 157–158. ACM, 2010.

[HHK95] M. R. Henzinger, T. A. Henzinger, and P. W. Kopke. Computing simulations on finite and infinite graphs. In *Proc. of FOCS: Foundations of Computer Science*, pages 453–462. IEEE Computer Society Press, 1995.

[HK97] M. Huth and M. Z. Kwiatkowska. Quantitative analysis and model checking. In *Proc. of LICS: Logic in Computer Science*, pages 111–122. IEEE Computer Society Press, 1997.

[HK99] T. A. Henzinger and P. W. Kopke. Discrete-time control for rectangular hybrid automata. *Theoretical Computer Science*, 221:369–392, 1999.

[HKR97] T. A. Henzinger, O. Kupferman, and S. K. Rajamani. Fair simulation. In *Proc. of CONCUR: Concurrency Theory*, LNCS 1243, pages 273–287. Springer-Verlag, 1997.

[Kar78] R. M. Karp. A characterization of the minimum cycle mean in a digraph. *Discrete Mathematics*, 23(3):309–311, 1978.

[Kar05] W. Karianto. Adding monotonic counters to automata and transition graphs. In *Proc. of DLT: Developments in Language Theory*, LNCS 3572, pages 308–319. Springer-Verlag, 2005.

[KL07] O. Kupferman and Y. Lustig. Lattice automata. In *Proc. of VMCAI: Verification, Model Checking, and Abstract Interpretation*, LNCS 4349, pages 199–213. Springer-Verlag, 2007.

[Kön36] D. König. *Theorie der endlichen und unendlichen Graphen.* Akademische Verlagsgesellschaft, Leipzig, 1936.

[Koz83] D. Kozen. Results on the propositional mu-calculus. *Theor. Comput. Sci.*, 27:333–354, 1983.

[KR03] F. Klaedtke and H. Rueß. Monadic second-order logics with cardinalities. In *Proc. of ICALP: International Colloquium on Automata, Languages and Programming*, pages 681–696, 2003.

[KS86] W. Kuich and A. Salomaa. *Semirings, Automata, Languages*, volume 5 of *Monographs in Theoretical Computer Science. An EATCS Series.* Springer-Verlag, 1986.

[KS88] S. R. Kosaraju and G. F. Sullivan. Detecting cycles in dynamic graphs in polynomial time (preliminary version). In *Proc. of STOC: Symposium on Theory of Computing*, pages 398–406. ACM, 1988.

[KV00] O. Kupferman and M. Y. Vardi. Synthesis with incomplete informatio. In *Advances in Temporal Logic*, pages 109–127. Kluwer Academic Publishers, January 2000.

[KV01] O. Kupferman and M. Y. Vardi. Weak alternating automata are not that weak. *ACM Trans. Comput. Log.*, 2(3):408–429, 2001.

[LB10] F. Lonsing and A. Biere. DepQBF: A dependency-aware QBF solver (System Description). *Journal on Satisfiability, Boolean Modeling and Computation*, 7:71–76, 2010.

[LGG11] M. Lindström, G. Geeraerts, and G. Goossens. A faster exact multiprocessor schedulability test for sporadic tasks. In *Proc. of RNTS: Real-Time and Network Systems*, 2011.

[LLM05] A. Lluch-Lafuente and U. Montanari. Quantitative μ-calculus and CTL based on constraint semirings. *Electr. Notes Theor. Comput. Sci.*, 112:37–59, 2005.

[LP07] Y. Lifshits and D. Pavlov. Potential theory for mean payoff games. *Journal of Mathematical Sciences*, 145(3):4967–4974, 2007.

[LX01] E. A. Lee and Y. Xiong. System-level types for component-based design. In *Proc. of EMSOFT: Embedded Software*, pages 237–253. Springer, 2001.

[Mar98] D. A. Martin. The determinacy of blackwell games. *J. Symb. Log.*, 63(4):1565–1581, 1998.

[McN93] R. McNaughton. Infinite games played on finite graphs. *Annals of Pure and Applied Logic*, 65(2):149–184, 1993.

[MH84] S. Miyano and T. Hayashi. Alternating finite automata on omega-words. In *Proc. of CAAP: Int. Colloquium on Trees in Algebra and Programming*, pages 195–210, 1984.

[Mil71] R. Milner. An algebraic definition of simulation between programs. In *Proc. of IJCAI: Int. Joint Conference on Artificial Intelligence*, pages 481–489. British Computer Society, 1971.

[MM02] A. McIver and C. Morgan. Games, probability and the quantitative μ-calculus q$M\mu$. In *Proc. of LPAR: Logic for Programming, Artificial Intelligence, and Reasoning*, LNCS 2514, pages 292–310. Springer, 2002.

[Moh97] M. Mohri. Finite-state transducers in language and speech processing. *Comp. Linguistics*, 23(2):269–311, 1997.

[MS72] A. R. Meyer and L. J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *Proc. of FOCS: Foundations of Computer Science*, pages 125–129. IEEE, 1972.

[NPPT09] M. Narizzano, C. Peschiera, L. Pulina, and A. Tacchella. Evaluating and certifying QBFs: A comparison of state-of-the-art tools. *AI Communications*, 22:191–210, 2009.

[Pap94] C. H. Papadimitriou. *Computational complexity*. Addison-Wesley Publishing Company, Reading, MA, 1994.

[Paz71] A. Paz. *Introduction to probabilistic automata*. Computer Science and Applied Mathematics. Academic Press, New York, 1971.

[Pis99] N. Pisaruk. Mean cost cyclical games. *Mathematics of Operations Research*, 24(4):817–828, 1999.

[Rab69] M. O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the Amer. Math. Soc.*, 141:1–35, 1969.

[Rab72] M. O. Rabin. *Automata on Infinite Objects and Church's Problem.* American Mathematical Society, Boston, MA, USA, 1972.

[Rei84] J. H. Reif. The complexity of two-player games of incomplete information. *Journal of Computer and System Sciences*, 29:274–301, 1984.

[RH04] T. C. Ruys and G. J. Holzmann. Advanced spin tutorial. In *SPIN*, LNCS 2989, pages 304–305. Springer-Verlag, 2004.

[RHN00] R. Raimi, R. Hojati, and K. S. Namjoshi. Environment modeling and language universality. *ACM Trans. Design Autom. Electr. Syst.*, 5(3):705–725, 2000.

[RP80] J. H. Reif and G. L. Peterson. A dynamic logic of multiprocessing with incomplete information. In *Proc. of POPL: Principles of Programming Languages*, pages 193–202. ACM, 1980.

[RSB05] J.-F. Raskin, M. Samuelides, and L. Van Begin. Games for counting abstractions. *Electr. Notes Theor. Comput. Sci.*, 128(6):69–85, 2005.

[RV07] K. Rozier and M. Y. Vardi. LTL satisfiability checking. In *14th Int. SPIN Workshop*, LNCS 4595, pages 149–167. Springer-Verlag, 2007.

[Saf88] S. Safra. On the complexity of ω-automata. In *Proc. of FOCS: Foundations of Computer Science*, pages 319–327. IEEE, 1988.

[Sch61] M. P. Schützenberger. On the definition of a family of automata. *Information and Control*, 4:245–270, 1961.

[Sch09] S. Schewe. Tighter bounds for the determinisation of Büchi automata. In *Proc. of FOSSACS: Foundations of Software Science and Computational Structures*, LNCS 5504, pages 167–181. Springer-Verlag, 2009.

[Sha53] L. S. Shapley. Stochastic games. In *Proc. of the National Acadamy of Science USA*, volume 39, pages 1095–1100, 1953.

[SSM03] H. Seidl, T. Schwentick, and A. Muscholl. Numerical document queries. In *Proc. of PODS: Principles of Database Systems*, pages 155–166. ACM, 2003.

[SSMH04] H. Seidl, T. Schwentick, A. Muscholl, and P. Habermehl. Counting in trees for free. In *Proc. of ICALP: International Colloquium on Automata, Languages and Programming*, pages 1136–1149, 2004.

[STV05] R. Sebastiani, S. Tonetta, and M. Vardi. Symbolic systems, explicit properties: On hybrid approaches for LTL symbolic model checking. In *Proc. of CAV: Computer-Aided Verification*, LNCS 3576, pages 350–363. Springer-Verlag, 2005.

[SVW87] A. P. Sistla, M. Y. Vardi, and P. Wolper. The complementation problem for Büchi automata with applications to temporal logic. *Theoretical Computer Science*, 49:217–237, 1987.

[TH03] A. Tozawa and M. Hagiya. XML schema containment checking based on semi-implicit techniques. In *Proc. of CIAA: Implementation and Application of Automata*, LNCS 2759, pages 213–225. Springer-Verlag, 2003.

[Tho97] W. Thomas. Languages, automata, and logic. In *Handbook of Formal Languages*, volume 3, Beyond Words, chapter 7, pages 389–455. Springer-Verlag, 1997.

[TV05] D. Tabakov and M. Y. Vardi. Experimental evaluation of classical automata constructions. In *Proc. of LPAR: Logic for Programming, Artificial Intelligence, and Reasoning*, LNCS 3835, pages 396–411. Springer-Verlag, 2005.

[TV07] D. Tabakov and M. Y. Vardi. Model-checking Büchi specifications. In *Pre-proceedings of LATA: Language and Automata Theory and Applications*, 2007.

[Var96] M. Y. Vardi. An automata-theoretic approach to linear temporal logic. In *Proc. of Banff Higher Order Workshop*, LNCS 1043, pages 238–266. Springer-Verlag, 1996.

[Vel11] Y. Velner. The complexity of mean-payoff automaton expression. *CoRR*, abs/1106.3054, 2011.

[VR11] Y. Velner and A. Rabinovich. Church synthesis problem for noisy input. In *Proc. of FOSSACS: Foundations of Software Science and Computational Structures*, LNCS 6604, pages 275–289. Springer-Verlag, 2011.

[VW86] M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification (preliminary report). In *Proc. of LICS: Logic in Computer Science*, pages 332–344. IEEE Computer Society, 1986.

[VW94] M. Y. Vardi and P. Wolper. Reasoning about infinite computations. *Inf. Comput.*, 115(1):1–37, 1994.

[Zie98] W. Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theor. Comput. Sci.*, 200:135–183, 1998.

[ZP96] U. Zwick and M. Paterson. The complexity of mean payoff games on graphs. *Theor. Comput. Sci.*, 158(1&2):343–359, 1996.

Publication list

An up-to-date publication list is available at:

`http://www.lsv.ens-cachan.fr/~doyen/publications.html`

Book chapter

[1] L. Doyen and J.-F. Raskin. Games with imperfect information: Theory and algorithms. In *Lectures in Game Theory for Computer Scientists*, pages 185–212. Cambridge University Press, 2010.

Journal papers

[2] L. Brim, J. Chaloupka, L. Doyen, R. Gentilini, and J.-F. Raskin. Faster algorithms for mean-payoff games. *Formal Methods in System Design*, 38(2):97–118, 2011, Springer-Verlag.

[3] D. Berwanger, K. Chatterjee, M. De Wulf, L. Doyen, and T. A. Henzinger. Strategy construction for parity games with imperfect information. *Information and Computation*, 208(10):1206–1220, 2010, Elsevier.

[4] K. Chatterjee, L. Doyen, and T. A. Henzinger. Expressiveness and closure properties for quantitative languages. *Logical Methods in Computer Science*, 6(3:10), 2010, LMCS-Online.

[5] K. Chatterjee, L. Doyen, and T. A. Henzinger. Quantitative languages. *ACM Transactions on Computational Logic*, 11(4), 2010, ACM Press.

[6] L. Doyen and J.-F. Raskin. Antichains for the automata-based approach to model-checking. *Logical Methods in Computer Science*, 5(1:5), 2009, LMCS-Online.

[7] M. De Wulf, L. Doyen, N. Markey, and J.-F. Raskin. Robust safety of timed automata. *Formal Methods in System Design*, 33(1-3):45–84, 2008, Springer-Verlag.

[8] L. Doyen, T. A. Henzinger, and J.-F. Raskin. Equivalence of labeled markov chains. *Inernational Journal of Foundations of Computer Science*, 19(3):549–563, 2008, World Scientific.

[9] K. Chatterjee, L. Doyen, T. A. Henzinger, and J.-F. Raskin. Algorithms for omega-regular games of incomplete information. *Logical Methods in Computer Science*, 3(3:4), 2007, LMCS-Online.

[10] L. Doyen. Robust parametric reachability for timed automata. *Information Processing Letters*, 102(5):208–213, 2007, Elsevier.

[11] Martin De Wulf, Laurent Doyen, and Jean-François Raskin. Almost ASAP semantics: From timed models to timed implementations. *Formal Aspects of Computing*, 17(3):319–341, 2005, Springer-Verlag.

Conference papers

[12] T. Brihaye, V. Bruyère, L. Doyen, M. Ducobu, and J.-F. Raskin. Antichain-based QBF solving. In *Proc. of ATVA: Automated Technology for Verification and Analysis*, LNCS. Springer-Verlag, 2011.

[13] T. Brihaye, L. Doyen, G. Geeraerts, J. Ouaknine, J.-F. Raskin, and J. Worrell. On reachability for hybrid automata over bounded time. In *Proc. of ICALP: Automata, Languages and Programming*, LNCS 6756, pages 416–427. Springer-Verlag, 2011.

[14] K. Chatterjee and L. Doyen. Energy and mean-payoff parity Markov decision processes. In *Proc. of MFCS: Mathematical Foundations of Computer Science*, LNCS 6907, pages 206–218. Springer-Verlag, 2011.

[15] K. Chatterjee, L. Doyen, and R. Singh. On memoryless quantitative objectives. In *Proc. of FCT: Fundamentals of Computation Theory*, LNCS 6914, pages 148–159. Springer-Verlag, 2011.

[16] L. Doyen, T. Massart, and M. Shirmohammadi. Infinite synchronizing words for probabilistic automata. In *Proc. of MFCS: Mathematical Foundations of Computer Science*, LNCS 6907, pages 278–289. Springer-Verlag, 2011.

[17] K. Chatterjee and L. Doyen. The complexity of partial-observation parity games. In *Proc. of LPAR: Logic for Programming, Artificial Intelligence, and Reasoning*, LNCS 6397, pages 1–14. Springer-Verlag, 2010.

[18] K. Chatterjee and L. Doyen. Energy parity games. In *Proc. of ICALP: International Colloquium on Automata, Languages and Programming (Part II)*, LNCS 6199, pages 599–610. Springer-Verlag, 2010.

[19] K. Chatterjee, L. Doyen, H. Edelsbrunner, T. A. Henzinger, and P. Rannou. Mean-payoff automaton expressions. In *Proc. of CONCUR: Concurrency Theory*, LNCS 6269, pages 269–283. Springer-Verlag, 2010.

[20] K. Chatterjee, L. Doyen, H. Gimbert, and T. A. Henzinger. Randomness for free. In *Proc. of MFCS: Mathematical Foundations of Computer Science*, LNCS 6281, pages 246–257. Springer-Verlag, 2010.

[21] K. Chatterjee, L. Doyen, and T. A. Henzinger. Qualitative analysis of partially-observable Markov decision processes. In *Proc. of MFCS: Mathematical Foundations of Computer Science*, LNCS 6281, pages 258–269. Springer-Verlag, 2010.

[22] K. Chatterjee, L. Doyen, T. A. Henzinger, and J.-F. Raskin. Generalized mean-payoff and energy games. In *Proc. of FSTTCS: Foundations of Software Technology and Theoretical Computer Science*, Dagstuhl Seminar Proceedings 08008. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), 2010.

[23] A. Degorre, L. Doyen, R. Gentilini, J.-F. Raskin, and S. Toruńczyk. Energy and mean-payoff games with imperfect information. In *Proc. of CSL: Computer Science Logic*, LNCS 6247, pages 260–274. Springer-Verlag, 2010.

[24] L. Doyen, T. A. Henzinger, A. Legay, and D. Nickovic. Robustness of sequential circuits. In *Proc. of ACSD: Application of Concurrency to System Design*, pages 77–84. IEEE Computer Society Press, 2010.

[25] L. Doyen and J.-F. Raskin. Antichains algorithms for finite automata. In *Proc. of TACAS: Tools and Algorithms for the Construction and Analysis of Systems*, LNCS 6015, pages 2–22. Springer-Verlag, 2010.

[26] K. Chatterjee, L. Doyen, and T. A. Henzinger. Alternating weighted automata. In *Proc. of FCT: Fundamentals of Computation Theory*, LNCS 5699, pages 3–13. Springer-Verlag, 2009.

[27] K. Chatterjee, L. Doyen, and T. A. Henzinger. Expressiveness and closure properties for quantitative languages. In *Proc. of LICS: Logic in Computer Sciences*, pages 199–208. IEEE, 2009.

[28] K. Chatterjee, L. Doyen, and T. A. Henzinger. Probabilistic weighted automata. In *Proc. of CONCUR: Concurrency Theory*, LNCS 5710, pages 244–258. Springer-Verlag, 2009.

[29] K. Chatterjee, L. Doyen, and T. A. Henzinger. A survey of stochastic games with limsup and liminf objectives. In *Proceedings of ICALP: International Colloquium on Automata, Languages and Programming (Part II)*, Lecture Notes in Computer Science 5556, pages 1–15. Springer-Verlag, 2009.

[30] L. Doyen, G. Geeraerts, J.-F. Raskin, and J. Reichert. Realizability of real-time logics. In *Proc. of FORMATS: Formal Modelling and Analysis of Timed Systems*, LNCS 5813, pages 133–148. Springer-Verlag, 2009.

[31] D. Berwanger, K. Chatterjee, L. Doyen, T. A. Henzinger, and S. Raje. Strategy construction for parity games with imperfect information. In *Proc. of CONCUR: Concurrency Theory*, LNCS 5201, pages 325–339. Springer-Verlag, 2008.

[32] D. Berwanger and L. Doyen. On the power of imperfect information. In *Proc. of FSTTCS: Foundations of Software Technology and Theoretical Computer Science*, Dagstuhl Seminar Proceedings 08004. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), 2008.

[33] K. Chatterjee, L. Doyen, and T. A. Henzinger. Quantitative languages. In *Proc. of CSL: Computer Science Logic*, LNCS 5213, pages 385–400. Springer-Verlag, 2008.

[34] M. De Wulf, L. Doyen, N. Maquet, and J.-F. Raskin. Antichains: Alternative algorithms for LTL satisfiability and model-checking. In *Proc. of TACAS: Tools and Algorithms for the Construction and Analysis of Systems*, LNCS 4963, pages 63–77. Springer-Verlag, 2008.

[35] L. Doyen, T. A. Henzinger, B. Jobstmann, and T. Petrov. Interface theories with component reuse. In *Proc. of EMSOFT: Embedded Software*, pages 79–88. ACM-Press, 2008.

[36] L. Doyen and J.-F. Raskin. Improved algorithms for the automata-based approach to model-checking. In *Proc. of TACAS: Tools and Algorithms for the Construction and Analysis of Systems*, LNCS 4424, pages 451–465. Springer-Verlag, 2007.

[37] K. Chatterjee, L. Doyen, T. A. Henzinger, and J.-F. Raskin. Algorithms for omega-regular games of incomplete information. In *Proceedings of CSL: Computer Science Logic*, Lecture Notes in Computer Science 4207, pages 287–302. Springer-Verlag, 2006.

[38] M. De Wulf, L. Doyen, T. A. Henzinger, and J.-F. Raskin. Antichains: A new algorithm for checking universality of finite automata. In *Proc. of CAV: Computer-Aided Verification*, LNCS 4144, pages 17–30. Springer-Verlag, 2006.

[39] M. De Wulf, L. Doyen, and J.-F. Raskin. A lattice theory for solving games of imperfect information. In *Proc. of HSCC: Hybrid Systems—Computation and Control*, LNCS 3927, pages 153–168. Springer-Verlag, 2006.

[40] Martin De Wulf, L. Doyen, and J.-F. Raskin. Systematic implementation of real-time models. In *Proc. of FM: Formal Methods*, LNCS 3582, pages 139–156. Springer-Verlag, 2005.

[41] L. Doyen, T. A. Henzinger, and J.-F. Raskin. Automatic rectangular refinement of affine hybrid systems. In *Proc. of FORMATS: Formal Modelling and Analysis of Timed Systems*, LNCS 3829, pages 144–161. Springer-Verlag, 2005.

[42] M. De Wulf, L. Doyen, N. Markey, and J.-F. Raskin. Robustness and implementability of timed automata. In *Proc. of FORMATS-FTRTFT*, LNCS 3253, pages 118–133. Springer-Verlag, 2004.

[43] M. De Wulf, L. Doyen, and J.-F. Raskin. Almost ASAP semantics: From timed models to timed implementations. In *Proc. of HSCC: Hybrid Systems—Computation and Control*, LNCS 2993, pages 296–310. Springer-Verlag, 2004.

Tool papers

[44] D. Berwanger, K. Chatterjee, M. De Wulf, L. Doyen, and T. A. Henzinger. Alpaga: A tool for solving parity games with imperfect information. In *Proc. of TACAS: Tools and Algorithms for the Construction and Analysis of Systems*, LNCS 5505, pages 58–61. Springer, 2009.

[45] M. De Wulf, L. Doyen, N. Maquet, and J.-F. Raskin. Alaska: Antichains for logic, automata and symbolic kripke structures analysis. In *Proc. of ATVA: Automated Technology for Verification and Analysis*, LNCS 5311, pages 240–245. Springer-Verlag, 2008.

Other publications

[46] K. Chatterjee and L. Doyen. Games and Markov decision processes with mean-payoff parity and energy parity objectives. In *Proc. of MEMICS 2011: Mathematical and Engineering Methods in Computer Science*, Lecture Notes in Computer Science. Springer-Verlag, 2011.

[47] L. Doyen, T. Massart, and M. Shirmohammadi. Synchronizing objectives for Markov decision processes. In *Proc. of iWIGP: Interactions, Games and Protocols*, Elec. Proc. in Theoretical Computer Science, pages 61–75, 2011.