Strategy Construction for Parity Games with Imperfect Information*

Dietmar Berwanger[1], Krishnendu Chatterjee[2], Laurent Doyen[3], Thomas A. Henzinger[3,4], and Sangram Raje[5]

[1] RWTH Aachen, Germany
[2] CE, University of California, Santa Cruz, U.S.A.
[3] I&C, École Polytechnique Fédérale de Lausanne (EPFL), Switzerland
[4] EECS, University of California, Berkeley, U.S.A.
[5] IIT Bombay, India

Abstract. We consider *imperfect-information* parity games in which strategies rely on observations that provide imperfect information about the history of a play. To solve such games, i.e., to determine the winning regions of players and corresponding winning strategies, one can use the subset construction to build an equivalent *perfect-information* game. Recently, an algorithm that avoids the inefficient subset construction has been proposed. The algorithm performs a fixed-point computation in a lattice of antichains, thus maintaining a succinct representation of state sets. However, this representation does not allow to recover winning strategies.

In this paper, we build on the antichain approach to develop an algorithm for constructing the winning strategies in parity games of imperfect information. We have implemented this algorithm as a prototype. To our knowledge, this is the first implementation of a procedure for solving imperfect-information parity games on graphs.

1 Introduction

Parity games capture the algorithmic essence of fundamental problems in state-based system analysis [11]. They arise as natural evaluation games for the μ-calculus, an expressive logic that subsumes most specification formalisms for reactive systems, and they are closely related to alternating ω-automata [7].

In the basic variant, a parity game is played on a finite graph with nodes labeled by natural numbers denoting *priorities*. There are two players, Player 1 and Player 2, who take turns in moving a token along the edges of the graph starting from a designated initial node. In a play, the players thus form an infinite path, and Player 1 wins if the least priority that is visited infinitely often is even; otherwise Player 2 wins. These are games of *perfect information*: during the play each of the players is informed about the current position of the token. One key

* This research was supported in part by the NSF grants CCR-0132780, CNS-0720884, and CCR-0225610, by the Swiss National Science Foundation, by the European COMBEST project, and by the Deutsche Forschungsgemeinschaft (DFG).

property of parity games is memoryless determinacy: from every initial node, either Player 1 or Player 2 has a winning strategy that does not depend on the history of the play [5]. As a consequence, a winning strategy can be represented as a subset of the edges of the graph, and the problem of constructing a winning strategy is in $\mathrm{NP} \cap \mathrm{coNP}$.

The perfect-information setting is often not sufficient in practice. The need to model uncertainty about the current state of a system arises in many situations. For instance in controller-synthesis applications, certain parameters of the plant under control may not be observable by the controller. Likewise in multi-component design, individual components of a complex system may have private variables invisible to other components. As a way to handle state-explosion problems, one may accept a loss of information in a concrete model in order to obtain a manageable abstract model of imperfect information.

One fundamental question is how to model *imperfect information*. In the classical theory of extensive games, this is done by partitioning the game tree into information sets signifying that a player cannot distinguish between different decision nodes of the same information set [6]. Technically, this corresponds to restricting the set of strategies available to a player by requiring a uniform choice across all nodes of an information set. However, for the algorithmic analysis of games of infinite duration on graphs, the information sets need to be finitely represented. Such a model is obtained by restricting to strategies that rely on observations corresponding to a partitioning of the game graph.

The model of imperfect information games that we consider here was originally introduced in [10]. Like in the perfect-information case, the game is played by two opposing players on a finite graph. The nodes of the graph, called *locations*, are partitioned into information sets indexed by *observations*. Intuitively, the only visible information available to Player 1 during a play is the observation corresponding to the current location, whereas Player 2 has perfect information about the current location of the game. The starting location is known to both players. Following [2], the parity winning condition is defined in terms of priorities assigned to observations.

The basic algorithmic problems about parity games are (1) to determine the *winning region* of a player, that is, the set of initial locations from which he has a winning strategy, and (2) to construct such a *winning strategy*. One straightforward way to solve parity games of imperfect information is based on the following idea [10, 2]: after an initial prefix of a play, Player 1 may not know in which precise location the play currently is but, by keeping track of the history, he can identify a minimal set of locations that is guaranteed to contain the current location. Such a set, to which we refer as a *cell*, reflects the knowledge derived by a player from past play. Via a subset construction that associates moves in the game to transitions between cells, the original imperfect-information game over locations is transformed into an equivalent game of perfect information over cells. This approach, however, incurs an exponential increase in the number of states and is therefore inefficient.

For computing the winning region of a game, an algorithm that avoids the explicit subset construction has been proposed recently in [2]. The algorithm exploits a monotonicity property of imperfect-information games: if a cell is winning for Player 1, that is, if he wins from every location of the cell, then he also wins from every subset of the cell. Intuitively, the subcell represents more precise knowledge than the entire cell. It is therefore sufficient to manipulate sets of cells that are *downward-closed* in the sense that, if a cell belongs to the set, then all its subcells also belong to it. As a succinct representation for downward-closed sets of cells, the algorithm maintains antichains that consist of maximal elements in the powerset lattice of cells. The winning region can now be computed symbolically by evaluating its characterization as a μ-calculus formula over the lattice. One particular effect of this procedure is that the discovery of winning cells propagates backwards, rather than forwards from the initial location, and thus avoids the construction and exploration of cells that are not relevant for solving the game.

On many instances, the antichain algorithm performs significantly better than the subset construction for computing winning regions. However, in contrast to the latter, the antichain algorithm does not construct winning strategies. Indeed, we argue that there is no direct way to extract a winning strategy from the symbolic fixed-point computation. In terms of logic, the algorithm evaluates a μ-calculus formula describing the winning region, which corresponds to evaluating a monadic expression with second-order quantifiers that range over (sets of) nodes in the game graph. On the other hand, strategies are not monadic objects; already memoryless location- or observation-based strategies are composed of binary objects, namely, edges of the graph or pairs of cells. In particular, we show that already in parity games of perfect information knowing the winning region of a game does not make the problem of constructing a winning strategy easier. In imperfect-information games there are additional sources of complexity: the size of a winning strategy may be exponentially larger than the winning region, already for reachability objectives. Nevertheless, the construction of winning strategies is crucial for many applications such as controller synthesis or counterexample-guided abstraction-refinement [8].

In this paper, we present an algorithm for constructing winning strategies in parity games of imperfect information. One main concern is to avoid the subset construction. To accomplish this, our algorithm works with symbolic representations of set of cells and builds on the antichain technique. It is based on an elementary algorithm proposed by McNaughton [9] and presented for parity games by Zielonka [13]. This algorithm works recursively: from the viewpoint of Player 1, in each stage a smaller game is obtained by removing the attractor region from which Player 2 can ensure to reach the minimal odd priority. This operation of removal marks the main difficulty in adapting the algorithm to antichains, as the residual subgame is in general not downward-closed. Intuitively, switching between the sides of the two players breaks the succinct representation. We overcome this difficulty by letting, in a certain sense, Player 1 simulate Player 2. Technically, this amounts to replacing two alternating reachability computations by the computation of a strategy that simultaneously satisfies a reachability and a safety objective.

We have implemented the algorithm as a prototype. To our knowledge, this is the first automatic tool for solving imperfect-information parity games on graphs. A full version of this paper with detailed proofs is available in [1].

2 Definitions

Let Σ be a finite alphabet of actions and let Γ be a finite alphabet of observations. A *game structure of imperfect information* over Σ and Γ is a tuple $G = (L, l_0, \Delta, \gamma)$, where L is a finite set of locations (or states), $l_0 \in L$ is the initial location, $\Delta \subseteq L \times \Sigma \times L$ is a set of labelled transitions, and $\gamma : \Gamma \to 2^L \setminus \emptyset$ is an observability function that maps each observation to a set of locations. Abusing notation, we usually identify the set $\gamma(o)$ with the observation symbol o. We require the following two conditions on G: (i) for all $\ell \in L$ and all $\sigma \in \Sigma$, there exists $\ell' \in L$ such that $(\ell, \sigma, \ell') \in \Delta$, i.e., the transition relation is total, and (ii) the set $\{\gamma(o) \mid o \in \Gamma\}$ partitions L. For each $\ell \in L$, let $\mathsf{obs}(\ell) = o$ be the unique observation such that $\ell \in \gamma(o)$. In the special case where $\Gamma = L$ and $\mathsf{obs}(\ell) = \ell$, for all $\ell \in L$, we say that G is a game structure of *perfect information* over Σ. For infinite sequences of locations $\pi = \ell_1 \ell_2 \dots$, we define $\mathsf{obs}(\pi) = o_1 o_2 \dots$ where $\mathsf{obs}(\ell_i) = o_i$ for all $i \geq 1$, and similarly for finite sequences of locations. For $\sigma \in \Sigma$ and $s \subseteq L$, we define $\mathsf{post}_\sigma(s) = \{\ell' \in L \mid \exists \ell \in s : (\ell, \sigma, \ell') \in \Delta\}$ as the set of σ-successors of locations in s.

The game on G is played in rounds. In each round, Player 1 chooses an action $\sigma \in \Sigma$, and Player 2 chooses a successor ℓ' of the current location ℓ such that $(\ell, \sigma, \ell') \in \Delta$. A *play* in G is an infinite sequence $\pi = \ell_1 \ell_2 \dots$ of locations such that (i) $\ell_1 = l_0$, and (ii) for all $i \geq 0$, there exists $\sigma_i \in \Sigma$ such that $(\ell_i, \sigma_i, \ell_{i+1}) \in \Delta$.

A *strategy* for Player 1 in G is a function $\alpha : \Gamma^+ \to \Sigma$. The set of possible *outcomes* of α in G is the set $\mathsf{Outcome}(G, \alpha)$ of plays $\pi = \ell_1 \ell_2 \dots$ such that $(\ell_i, \alpha(\mathsf{obs}(\ell_1 \dots \ell_i)), \ell_{i+1}) \in \Delta$ for all $i \geq 1$. We say that a strategy α is *memoryless* if $\alpha(\rho \cdot o) = \alpha(\rho' \cdot o)$ for all $\rho, \rho' \in \Gamma^*$. We say that a strategy uses *finite memory* if it can be represented by a finite-state deterministic *transducer* $(M, m_0, \lambda, \delta)$ with finite set of states M (the memory of the strategy), initial state $m_0 \in M$, where $\lambda : M \to \Sigma$ labels states with actions, and $\delta : M \times \Gamma \to M$ is a transition function labeled by observations. In state m, the strategy recommends the action $\lambda(m)$, and when Player 2 chooses a location with observation o, it updates the internal state to $\delta(m, o)$. Formally, $(M, m_0, \lambda, \delta)$ defines the strategy α such that $\alpha(\rho) = \lambda(\hat{\delta}(m_0, \rho))$ for all $\rho \in \Gamma^+$, where $\hat{\delta}$ extends δ to sequences of observations in the usual way. The *size* of a finite-state strategy is the number $|M|$ of states of its transducer.

An *objective* for a game structure $G = (L, l_0, \Delta, \gamma)$ is a set $\phi \subseteq \Gamma^\omega$ of infinite sequences of observations. A strategy α for Player 1 is *winning* for an objective ϕ if $\mathsf{obs}(\pi) \in \phi$ for all $\pi \in \mathsf{Outcome}(G, \alpha)$. We say that set of locations $s \subseteq L$ is *winning* for ϕ if there exists a strategy α for Player 1 such that α is winning for ϕ in $G_\ell := (L, \ell, \Delta, \gamma)$ for all $\ell \in s$. A *game* is a pair (G, ϕ) consisting of a game structure and a matching objective. We say that Player 1 wins the game, if he has a winning strategy for the objective ϕ.

We consider the following classical objectives. Given a set $\mathcal{T} \subseteq \Gamma$ of target observations, the *safety* objective $\mathsf{Safe}(\mathcal{T})$ requires that the play remains within the set $\mathcal{T}$, that is, $\mathsf{Safe}(\mathcal{T}) = \{o_1 o_2 \dots \mid \forall k \geq 1 : o_k \in \mathcal{T}\}$. Dually, the *reachability* objective $\mathsf{Reach}(\mathcal{T})$ requires that the play visits the set $\mathcal{T}$ at least once, that is, $\mathsf{Reach}(\mathcal{T}) = \{o_1 o_2 \dots \mid \exists k \geq 1 : o_k \in \mathcal{T}\}$. The *Büchi* objective $\mathsf{Buchi}(\mathcal{T})$ requires that an observation in $\mathcal{T}$ occurs infinitely often, that is, $\mathsf{Buchi}(\mathcal{T}) = \{o_1 o_2 \dots \mid \forall N \cdot \exists k \geq N : o_k \in \mathcal{T}\}$. Dually, the *coBüchi* objective $\mathsf{coBuchi}(\mathcal{T})$ requires that only observations in $\mathcal{T}$ occur infinitely often. Formally, $\mathsf{coBuchi}(\mathcal{T}) = \{o_1 o_2 \dots \mid \exists N \cdot \forall k \geq N : o_k \in \mathcal{T}\}$. Finally, given a *priority function* $p : \Gamma \to \mathbb{N}$ that maps each observation to a non-negative integer priority, the *parity* objective $\mathsf{Parity}(p)$ requires that the minimum priority that appears infinitely often is even. Formally, $\mathsf{Parity}(p) = \{o_1 o_2 \dots \mid \min\{p(o) \mid \forall N \cdot \exists k \geq N : o = o_k\} \text{ is even}\}$. We denote by $\mathsf{coParity}(p)$ the complement objective of $\mathsf{Parity}(p)$, i.e., $\mathsf{coParity}(p) = \{o_1 o_2 \dots \mid \min\{p(o) \mid \forall N \cdot \exists k \geq N : o = o_k\} \text{ is odd}\}$. Parity objectives are a canonical form to express all ω-regular objectives [12]. In particular, they subsume safety, reachability, Büchi and coBüchi objectives.

Notice that objectives are defined as sets of sequences of observations, and they are therefore visible to Player 1. A game with a safety (resp. reachability) objective defined as a set of plays can be transformed into an equivalent game with a visible safety (resp. reachability) objective in polynomial time.

3 Antichain Algorithm

Let Σ be an alphabet of actions and let Γ be an alphabet of observations. We consider the problem of deciding, given a game structure $G = (L, l_0, \Delta, \gamma)$ and a parity objective ϕ, whether Player 1 has a winning strategy for ϕ in G. If the answer is YES, we ask to construct such a winning strategy. This problem is known to be EXPTIME-complete already for reachability objectives [10, 2]. The basic algorithm proposed in [10] constructs a game (G^{K}, ϕ') such that (i) $G^{\mathsf{K}} = (S, s_0, \Delta', \gamma')$ is a game structure of *perfect information* over the action alphabet Σ, and (ii) Player 1 has a winning strategy for ϕ in G if and only if Player 1 has a winning strategy for ϕ' in G^{K}. The game structure G^{K} is obtained by a subset construction where $S = 2^L \setminus \{\emptyset\}$ and $(s_1, \sigma, s_2) \in \Delta'$ if and only if there exists an observation $o \in \Gamma$ such that $s_2 = \mathsf{post}_\sigma(s_1) \cap \gamma(o)$ and $s_2 \neq \emptyset$. In the sequel, we call a set $s \subseteq L$ a *cell*. A cell summarizes the current *knowledge* of Player 1, i.e., the set of possible locations in which the game G can be after the sequence of observations seen by Player 1. Notice that every cell reachable in G^{K} is a subset of some observation, and so the parity objective ϕ' is defined by extending to cells in the natural way the priority function p that defines ϕ. Notice that an objective for G^{K} is a set of infinite sequences of cells, since locations and observations coincide in games of perfect information. In (G^{K}, ϕ'), memoryless winning strategies always exist. Hence, they can be converted into winning strategies in (G, ϕ) that depend only on the current cell in G^{K}. Due to the explicit construction of G^{K}, this approach involves an exponential blow-up of the original game structure.

In [2], an alternative algorithm is proposed to solve games of imperfect information. Winning cells are computed symbolically, avoiding the exponential subset construction. The algorithm is based on the *controllable predecessor operator* $\mathsf{CPre} : 2^S \rightarrow 2^S$ which, given a set of cells q, computes the set of cells q' from which Player 1 can force the game into a cell of q in one round. Formally,

$$\mathsf{CPre}(q) = \{s \in S \mid \exists \sigma \in \Sigma \cdot \forall s' : \text{ if } (s, \sigma, s') \in \Delta' \text{ then } s' \in q\}.$$

The key of the algorithm is that $\mathsf{CPre}(\cdot)$ preserves downward-closedness, which intuitively means that if Player 1 has a strategy from s to force the game to be in q in the next round, then he also has such a strategy from all $s' \subseteq s$ because then Player 1 has a more precise knowledge in s' than in s. Formally, a set q of cells is *downward-closed* if $s \in q$ implies $s' \in q$ for all $s' \subseteq s$. If q is downward-closed, then so is $\mathsf{CPre}(q)$. Since parity games can be solved by evaluating a μ-calculus formula over the powerset lattice $(S, \subseteq, \cup, \cap)$, and since $\mathsf{CPre}(\cdot)$, $\cap$ and $\cup$ preserve downward-closedness, it follows that a symbolic algorithm maintains only downward-closed sets q of cells, and can therefore use a compact representation, namely their maximal elements $\lceil q \rceil = \{s \in q \mid s \neq \emptyset \text{ and } \forall s' \in q : s \not\subset s'\}$, forming *antichains* of cells, i.e., sets of $\subseteq$-incomparable cells. The set $\mathcal{A}$ of antichains is partially ordered as follows: for $q, q' \in \mathcal{A}$, let $q \sqsubseteq q'$ iff $\forall s \in q \cdot \exists s' \in q' : s \subseteq s'$. The least upper bound of $q, q' \in \mathcal{A}$ is $q \sqcup q' = \lceil \{s \mid s \in q \text{ or } s \in q'\} \rceil$, and their greatest lower bound is $q \sqcap q' = \lceil \{s \cap s' \mid s \in q \text{ and } s' \in q'\} \rceil$. The partially ordered set $(\mathcal{A}, \sqsubseteq, \sqcup, \sqcap)$ forms a complete lattice. We view antichains of location sets as a symbolic representation of $\subseteq$-downward-closed sets of cells.

The advantage of the symbolic antichain approach over the explicit subset construction has been established in practice for different applications in model-checking (*e.g.* [3, 4]). The next lemma shows that the antichain algorithm may be exponentially faster than the subset construction.

Lemma 1 (See also [3]). *There exists a family $(G_k)_{k \geq 2}$ of reachability games of imperfect information with k locations such that, on input G_k the subset-construction algorithm runs in time exponential in k whereas the antichain algorithm runs in time polynomial in k.*

The antichain algorithm computes a compact representation of the set of winning cells. However, it does not produce a winning strategy. We point out that, already for parity games with perfect information, if there exists a polynomial-time algorithm that, given a game and the set of winning locations for Player 1, constructs a memoryless winning strategy, then parity games can be solved in polynomial time.

Proposition 2. *The following two problems on parity games with perfect information in which Player 1 wins are polynomial-time equivalent.*

(i) Given a game, construct a memoryless winning strategy.

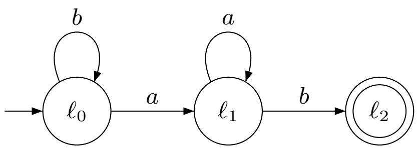

Fig. 1. A reachability game G.

(ii) Given a game and the set of winning locations for Player 1*, construct a memoryless winning strategy.*

Proof. For any instance of problem (i), that is, a game G where Player 1 wins from the initial location l_0, we construct an instance (G', W) of problem (ii) in such a way that every memoryless winning strategy in G' corresponds to a winning strategy for G. (The converse is trivial.)

Without loss, we assume that no priorities in G are less than 2. The game G' is obtained by adding to G a "reset" location z of priority 1, with transitions that allow Player 1 to reach z from any location of G where he moves, and with one transition from z back to l_0. In the new game, Player 1 wins from any location by first moving via z to l_0 and then following the winning strategy he has in G. Thus, G' together with the set of all locations is an instance of problem (ii). Obviously this can be constructed in polynomial time. Let now α be a memoryless winning strategy in G'. No play starting from l_0 that follows α can reach z, otherwise Player 1 loses. Thus, α is readily a winning strategy in the original game G. ■

We also argue that, in games with imperfect information, even for simple reachability objectives the antichain representation of the set of winning cells may not be sufficient to construct a winning strategy. Consider the game G depicted in Fig. 1, with reachability objective $\mathsf{Reach}(\{\ell_2\})$. The observations are $\{\ell_0, \ell_1\}$ and $\{\ell_2\}$. Since $\mathsf{CPre}(\{\{\ell_2\}\}) = \{\{\ell_1\}\}$ (by playing action b) and $\mathsf{CPre}(\{\{\ell_1\}, \{\ell_2\}\}) = \{\{\ell_0, \ell_1\}\}$ (by playing action a), the fixed-point computed by the antichain algorithm is $\{\{\ell_2\}, \{\ell_0, \ell_1\}\}$. However, from $\{\ell_0, \ell_1\}$, after playing a, Player 1 reaches the cell $\{\ell_1\}$ which is not in the fixed-point (however, it is subsumed by the cell $\{\ell_0, \ell_1\}$). Intuitively, the antichain algorithm has forgotten which action is to be played next. Notice that playing a again, and thus forever, is not winning. The next lemma formalizes this intuition.

Lemma 3. *There exists a family of games G_k with $O(p(k))$ many locations for a polynomial p, and a reachability objective ϕ, such that the fixed point computed by the antichain algorithm for (G_k, ϕ) is of polynomial size in k, whereas any finite-memory winning strategy for (G_k, ϕ) is of exponential size in k.*

We first present the ingredients of the proof informally. Let $p_1, p_2, \dots$ be the list of prime numbers in increasing order. For $k \geq 1$, let $\Sigma_k = \{1, \dots, k\}$.

The action alphabet of the game G_k is $\Sigma_k \cup \{\#, \bot\}$. The game is composed of subgames H_i, each consisting of a loop over p_i many locations $\ell_1, \dots, \ell_{p_i}$. From a location ℓ_j all actions in Σ_k lead to ℓ_{j+1} and from the last location ℓ_{p_i} Player 1 can return to the initial location ℓ_1 with any action in Σ_k except i. Formally, for all $1 \le i \le k$, we define the subgame H_i with location space $L_i = \{\ell_1, \dots, \ell_{p_i}\}$, initial location ℓ_1, and transition relation $E_i = \{(\ell_j, \sigma, \ell_{j+1}) \mid 1 \le j \le p_i - 1 \wedge \sigma \in \Sigma_k\} \cup \{(\ell_{p_i}, \sigma, \ell_1) \mid \sigma \in \Sigma_k \setminus \{i\}\}$. In the sequel, we assume that the location spaces of all H_i are disjoint, *e.g.* by adding a superscript i to the locations of L_i ($L_i = \{\ell_1^i, \dots, \ell_{p_i}^i\}$).

Fig. 2 shows the game G_k for $k = 2$. In general, in G_k, there is a unique trivial observation, so it is a blind game. We also assume that playing a particular action in a location where it is not allowed leads to a sink location from which Goal is not reachable. The plays start in location ℓ_0 where every move in Σ_k is allowed. The next location can be any of the initial locations of the subgames H_i. Thus, Player 1 can henceforth play any action $\sigma \in \Sigma_k$, except in the last location ℓ_{p_i} where playing $\sigma = i$ would lead to the sink. As he does not know in which of the H_i the play currently is, he should avoid playing $\sigma = i$ whenever his knowledge set contains $q_{p_i}^i$. However, after a certain number of steps (namely $p_k^* = \prod_{i=1}^k p_i$), the current location of the game will be one of the $\ell_{p_i}^i$. Then, taking a transition labeled by $\#$ necessarily leads to Goal. The $\#$ is not allowed in any other location, so that Player 1 needs to count the first p_k^* steps before playing that move. Notice that after the first round, Player 1 could play $\bot$, but this would not reduce the amount of memory needed to win. However, it shows that he is winning uniformly from all locations of the subgames H_i. Since the size p_k^* of the strategy is exponential in the size $\sum_{i=1}^k p_i$ of the game, the theorem follows.

Proof of Lemma 3. The location space of G_k is the disjoint union of $L_1, \dots, L_k$ and $\{q_0, \mathsf{Goal}, \mathsf{Bad}\}$. The initial location is q_0, the target observation consists of Goal, and the sink location is Bad. The transition relation contains each set E_i, the transitions $(\ell_j^i, \bot, \ell_0)$, and the transitions $(\ell_0, \sigma, \ell_1^i)$ and $(\ell_{p_i}^i, \#, \mathsf{Bad})$ for all $1 \le i \le k$, $1 \le j \le p_i$ and $\sigma \in \Sigma_k$. The transition relation is made total by adding the transitions $(q, \sigma, \mathsf{Bad})$ for each location ℓ of G_n and $\sigma \in \Sigma_k \cup \{\#\}$ such that there is no transition of the form (q, σ, q') for $q' \neq \mathsf{Bad}$. There is only one trivial observation, i.e., the observation alphabet Γ is a singleton.

First we show that Player 1 wins G_k. As there is exactly one observation, a strategy for Player 1 is a function $\lambda : \mathbb{N}^{\ge 0} \to \Sigma_k \cup \{\#, \bot\}$. We define the sets S_j such that any strategy λ such that $\lambda(j) \in S_j$ for all $j \ge 1$ is winning for Player 0. We take $S_1 = \Sigma_k$, $S_j = \{i \in \Sigma_k \mid j - 1 \mod p_i \neq 0\}$ for $2 \le j \le p_k^*$. Notice that $S_j \neq \emptyset$ because the least common multiple of $p_1, \dots, p_k$ is p_k^*. Finally, for $j > p_k^*$ we take $S_j = \{\#\}$. It is easy to show that any strategy defined by these sets is winning for Player 1.

For the second part of the theorem assume, towards a contradiction, that there exists a finite-state winning strategy $\hat{\lambda}$ with less than p_k^* states. Clearly, when playing any winning strategy, the $(p_k^* + 1)$-th location of the play in G_k must be $\ell_{p_i}^i$ for some $i \in \{1, \dots, k\}$. Moreover, each of the states $\ell_{p_i}^i$ could be

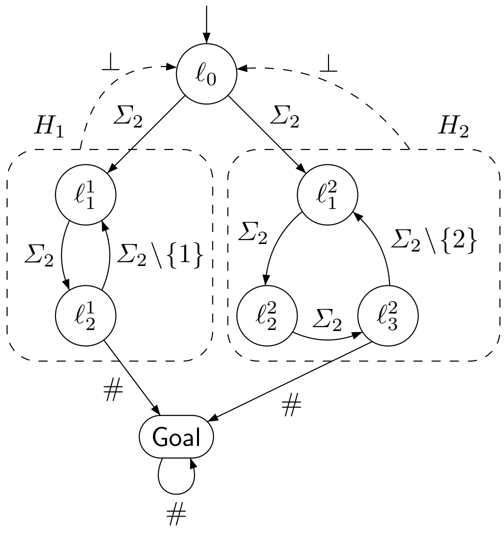

Fig. 2. The game G_2.

the current one, depending on the initial choice of Player 2 (after the first move of Player 1). Therefore, after p_k^* steps, any winning strategy must play #. In the case of $\hat{\lambda}$, the state of the automaton for $\hat{\lambda}$ after p_k^* steps has necessarily been visited in one of the previous steps. This means that # has been played before and thus $\hat{\lambda}$ is not a winning strategy as for all $j < p_k^*$, one of the subgames H_i is not in location $\ell^i_{p_i}$ after j steps of play, and thus playing # leads to a loss for Player 1. ■

Finally, we show that it is not trivial to efficiently compute $\mathsf{CPre}(\cdot)$. In the antichain representation, the controllable predecessor operator is defined as

$$\mathsf{CPre}(q) = \lceil \{s \subseteq L \mid \exists \sigma \in \Sigma \cdot \forall o \in \Gamma \cdot \exists s' \in q : \mathsf{post}_\sigma(s) \cap \gamma(o) \subseteq s'\} \rceil,$$

or equivalently as

$$\mathsf{CPre}(q) = \bigsqcup_{\sigma \in \Sigma} \bigsqcap_{o \in \Gamma} \bigsqcup_{s' \in q} \{\widetilde{\mathsf{pre}}_\sigma(s' \cup \overline{\gamma(o)})\}, \tag{1}$$

where $\widetilde{\mathsf{pre}}_\sigma(s) = \{s' \in S \mid \mathsf{post}_\sigma(\{s'\}) \subseteq s\}$ and $\overline{\gamma(o)} = L \setminus \gamma(o)$.

Notice that the least upper bound of a set $\{\ell_1, \dots, \ell_k\}$ of antichains can be computed in polynomial time, whereas a naive algorithm for the greatest lower bound is exponential. The next lemma shows that, as long as we use a reasonable representation of antichains which allows to decide in polynomial time whether an antichain contains a set larger than n, it is unlikely that $\mathsf{CPre}(\cdot)$ is computable in polynomial time.

Lemma 4. *The following problem is* NP-HARD*: given a game of imperfect information G, an antichain q and an integer n, decide whether there exists a set $B \in \mathsf{CPre}(q)$ with $|B| \geq n$.*

4 Strategy Construction with Antichains

We present a procedure to construct a winning strategy for a parity game of imperfect information $G = (L, l_0, \Delta, \gamma)$ over the alphabets Σ and Γ. It is sometimes convenient to reason in terms of the equivalent perfect-information game G^{K} obtained via the subset construction in Section 3. Let $\mathcal{C}$ denote the set of all cells s such that $s \subseteq \gamma(o)$ for some $o \in \Gamma$. Thus, $\mathcal{C}$ contains all locations of G^{K}. For $\mathcal{R} \subseteq \mathcal{C}$, a *cell strategy* on $\mathcal{R}$ is a memoryless strategy $\alpha : \mathcal{R} \to \Sigma$ for Player 1 in G^{K}. Given an objective $\phi \subseteq \mathcal{C}^\omega$ in G^{K}, we define

$$\begin{aligned}\mathsf{Win}^{\mathcal{R}}(\phi) := \{\, s \in \mathcal{R} \mid\ & \text{there exists a cell strategy } \alpha \text{ such that} \\ & \mathsf{Outcome}(G^{\mathsf{K}}_s, \alpha) \subseteq \phi \cap \mathsf{Safe}(\mathcal{R}) \,\}.\end{aligned}$$

In words, $\mathsf{Win}^{\mathcal{R}}(\phi)$ consists of cells s such that given the initial cell is s there exists a winning cell strategy for Player 1 to ensure ϕ while maintaining the game G^{K} in $\mathcal{R}$.

In Algorithm 1, we present a procedure to construct a winning cell strategy in G^{K} for objectives of the form

$$\mathsf{Reach}(\mathcal{T}) \cup (\mathsf{Parity}(p) \cap \mathsf{Safe}(\mathcal{F})),$$

where $\mathcal{T}, \mathcal{F} \subseteq \mathcal{C}$ are downward-closed sets of cells and $p : \Gamma \to \mathbb{N}$ is a priority function over observations. As p can be naturally extended to cells, the set $\mathsf{Parity}(p)$ contains the sequence of cells such that the minimal priority cell appearing infinitely often is even. The parity objective $\mathsf{Parity}(p)$ corresponds to the special case where $\mathcal{F} = \mathcal{C}$ and $\mathcal{T} = \emptyset$. Note that a winning strategy need not be defined on $\mathcal{T}$ since $\mathsf{Reach}(\mathcal{T})$ is satisfied for all cells in $\mathcal{T}$. Memoryless strategies are sufficient for this kind of objective in games of perfect information. Thus, we can restrict our attention without loss to memoryless cell strategies.

Informal description. The algorithm is based on two procedures $\mathsf{ReachOrSafe}(\mathcal{T}, \mathcal{F})$ and $\mathsf{ReachAndSafe}(\mathcal{T}, \mathcal{F})$ that use antichains to compute the set of winning cells and a winning strategy for the objectives $\mathsf{Reach}(\mathcal{T}) \cup \mathsf{Safe}(\mathcal{F})$ and $\mathsf{Reach}(\mathcal{T}) \cap \mathsf{Safe}(\mathcal{F})$, respectively, given downward-closed sets of cells $\mathcal{T} \subseteq \mathcal{C}$ and $\mathcal{F} \subseteq \mathcal{C}$. For perfect-information games, it is known that memoryless winning strategies exist for such combinations of safety and reachability objectives.

The procedure is called recursively, reducing the number of priorities. Given a parity function p we denote by $p - 2$ the parity function such that for all $o \in \Gamma$ we have $(p-2)(o) = p(o)$ if $p(o) \leq 1$, and $(p-2)(o) = p(o) - 2$ otherwise. For $i \geq 0$, we denote by $\mathcal{C}_p(i) = \{\, s \in \mathcal{C} \mid s \subseteq \gamma(o), o \in \Gamma, p(o) = i \,\}$ the set of cells with priority i. Let W_1 and W_2 be disjoint sets of cells, and let α_1 be a cell strategy on W_1 and α_2 be a cell strategy on W_2. We denote by $\alpha_1 \cup \alpha_2$ the cell strategy on $W_1 \cup W_2$ such that for all $s \in W_1 \cup W_2$, we have $(\alpha_1 \cup \alpha_2)(s) = \alpha_1(s)$ if $s \in W_1$, and $(\alpha_1 \cup \alpha_2)(s) = \alpha_2(s)$ otherwise.

Without loss of generality we assume that the cells in the target set $\mathcal{T}$ are *absorbing* (i.e., have self-loops only). In line 1 of Algorithm 1, we compute $W = \mathsf{Win}^{\mathcal{C}}(\phi)$ using the antichain algorithm of [2]. Since we assume that cells in $\mathcal{T}$ are absorbing, a winning cell strategy for the objective ϕ ensures that the set W is never left. In the rest of the algorithm and in the arguments below, we consider the sub-game induced by W. In line 2, the set W^* of winning cells and a winning cell strategy α^* on $W^* \setminus \mathcal{T}$ for the objective $\mathsf{Reach}(\mathcal{T})$ is computed by invoking the procedure $\mathsf{ReachOrSafe}$ with target $\mathcal{T}$ and safe set W. Then the set W_0 of cells is obtained along with a cell strategy α_0 that ensures that either W^* is reached or the set of priority 0 cells in W is reached. After this, the algorithm iterates a loop as follows: at iteration $i+1$, let W_i be the set of cells already obtained by the previous iteration and let $A_i = W \setminus W_i$. The algorithm is invoked recursively with W_i as target set, $A_i \setminus \mathcal{C}_p(1)$ as the safe set, and $p-2$ as the priority function to obtain a set W_{i+1} as a result. In the base case, where W consists of priorities 0, 1 and 2 only, since A_i has no priority 0 cells, the objective $\mathsf{Reach}(W_i) \cup (\mathsf{Parity}(p-2) \cap \mathsf{Safe}(A_i \setminus \mathcal{C}_p(1))$ can be equivalently written as $\mathsf{Reach}(W_i) \cup \mathsf{Safe}(A_i \cap \mathcal{C}_p(2))$. Therefore, in the base case, the recursive call is replaced by $\mathsf{ReachOrSafe}(W_i, A_i \cap \mathcal{C}_p(2))$. Notice that $W_i \subseteq W_{i+1}$. The algorithm proceeds until a fixpoint of $W_i = W_{i+1}$ is reached.

Correctness of the iteration. First, we have $W \setminus W^* \subseteq \mathcal{F}$ which essentially follows from the fact that from $W \setminus W^*$ Player 1 cannot reach $\mathcal{T}$. More precisely, if a cell $s \in W \setminus W^*$ does not belong to $\mathcal{F}$, then against every cell strategy for Player 1, there is a Player 2 strategy to ensure that the set $\mathcal{T}$ is not reached from s. Hence from s, against every cell strategy for Player 1, there is a Player 2 strategy to ensure that $\mathsf{Reach}(\mathcal{T}) \cup \mathsf{Safe}(\mathcal{F})$ is violated, and thus $\phi = \mathsf{Reach}(\mathcal{T}) \cup (\mathsf{Parity}(p) \cap \mathsf{Safe}(\mathcal{F}))$ is violated. This contradicts $s \in W = \mathsf{Win}^{\mathcal{C}}(\phi)$. The significance of the claim is that if W^* is reached, then Player 1 can ensure that $\mathcal{T}$ is reached, and since $W \setminus W^* \subseteq \mathcal{F}$ it follows that if W^* is not reached then the game stays safe in $\mathcal{F}$.

To establish the correctness of the iterative step, we claim that from the set W_{i+1} the cell strategy α_{i+1} on $W_{i+1} \setminus W_i$ which ensures

$$\mathsf{Reach}(W_i) \cup \big(\mathsf{Parity}(p-2) \cap \mathsf{Safe}(A_i \setminus \mathcal{C}_p(1))\big),$$

also ensures that

$$\mathsf{Reach}(W_i) \cup \big(\mathsf{Parity}(p) \cap \mathsf{Safe}(\mathcal{F} \setminus \mathcal{C}_p(1))\big).$$

Notice that in $A_i \setminus \mathcal{C}_p(1)$, there is no cell with priority 0 or priority 1 for the priority function p since $\mathcal{C}_p(0) \cap W \subseteq W_0 \subseteq W_i$. Hence, we have

$$\mathsf{Parity}(p-2) \cap \mathsf{Safe}(A_i \setminus \mathcal{C}_p(1)) = \mathsf{Parity}(p) \cap \mathsf{Safe}(A_i \setminus \mathcal{C}_p(1)).$$

Since $A_i \subseteq W \setminus W_0 \subseteq W \setminus W^* \subseteq \mathcal{F}$, it follows that the cell strategy α_{i+1} on $W_{i+1} \setminus W_i$ to ensure

$$\mathsf{Reach}(W_i) \cup \big(\mathsf{Parity}(p-2) \cap \mathsf{Safe}(A_i \setminus \mathcal{C}_p(1))\big),$$

Algorithm 1: Imperfect-Information Game Solver - **Solve**(G, $\mathcal{T}$, $\mathcal{F}$, p)

Input : A game structure G with target $\mathcal{T} \subseteq \mathcal{C}$, safe set $\mathcal{F} \subseteq \mathcal{C}$ and parity function p on Γ.

Output : $W = \mathsf{Win}^{\mathcal{C}}(\phi)$ where $\phi := \mathsf{Reach}(\mathcal{T}) \cup (\mathsf{Parity}(p) \cap \mathsf{Safe}(\mathcal{F}))$, and a winning cell strategy α on $W \setminus \mathcal{T}$ for ϕ.

begin
- $W \leftarrow \mathsf{Win}^{\mathcal{C}}(\phi)$
- $(W^*, \alpha^*) \leftarrow \mathsf{ReachAndSafe}(\mathcal{T}, W)$
- $(W_0, \alpha_0) \leftarrow \mathsf{ReachAndSafe}(W^* \cup (\mathcal{C}_p(0) \cap W), W))$
- Let α'_0 be a cell strategy on $(\mathcal{C}_p(0) \cap W) \setminus W^*$ such that $\mathsf{post}_{\alpha'_0(s)}(s) \cap \gamma(o) \in W$ for all $o \in \Gamma$ and $s \in (\mathcal{C}_p(0) \cap W) \setminus W^*$
- $\overline{\alpha}_0 \leftarrow \alpha_0 \cup \alpha'_0 \cup \alpha^*$
- $i \leftarrow 0$
- **repeat**
  - $A_i \leftarrow W \setminus W_i$
  - **if** $W \subseteq \mathcal{C}_p(0) \cup \mathcal{C}_p(1) \cup \mathcal{C}_p(2)$ **then**
    - $(W_{i+1}, \alpha_{i+1}) \leftarrow \mathsf{ReachOrSafe}(W_i, A_i \cap \mathcal{C}_p(2))$
  - **else**
    - $(W_{i+1}, \alpha_{i+1}) \leftarrow$ **Solve**(G, W_i, $A_i \setminus \mathcal{C}_p(1)$, $p-2$)
  - $\overline{\alpha}_{i+1} \leftarrow \overline{\alpha}_i \cup \alpha_{i+1}$
  - $i \leftarrow i+1$
- **until** $W_i = W_{i-1}$
- **return** $(W_i, \overline{\alpha}_i)$

end

also ensures that

$$\mathsf{Reach}(W_i) \cup \big(\mathsf{Parity}(p) \cap \mathsf{Safe}(\mathcal{F} \setminus \mathcal{C}_p(1))\big).$$

holds from all cells in W_{i+1}. By induction on i, composing the cell strategies (i.e., by taking the union of strategies obtained in the iteration) we obtain that from W_{i+1}, the cell strategy $\overline{\alpha}_{i+1}$ on $W_{i+1} \setminus \mathcal{T}$ for Player 1 ensures $\mathsf{Reach}(W_0) \cup \big(\mathsf{Parity}(p) \cap \mathsf{Safe}(\mathcal{F}) \cap \mathsf{coBuchi}(\mathcal{F} \setminus \mathcal{C}_p(1))\big)$. Note that to apply the induction step for i times, one may visit cells in $\mathcal{C}_p(1)$, but only finitely many times.

Termination. We claim that upon termination, we have $W_i = W$. Assume towards a contradiction that the algorithm terminates with $W_i = W_{i+1}$ and $W_{i+1} \neq W$. Then the following assertions hold. The set $A_i = W \setminus W_i$ is nonempty and

$$W_{i+1} = W_i = \mathsf{Win}^{W}\Big(\mathsf{Reach}(W_i) \cup \big(\mathsf{Parity}(p-2) \cap \mathsf{Safe}(A_i \setminus \mathcal{C}_p(1))\big)\Big),$$

that is, in the whole set A_i against all Player 1 cell strategies, Player 2 can ensure the complementary objective, i.e.,

$$\mathsf{Safe}(A_i) \cap \big(\mathsf{coParity}(p-2) \cup \mathsf{Reach}(A_i \cap \mathcal{C}_p(1))\big).$$

Now, we show that satisfying the above objective also implies satisfying $\mathsf{Safe}(A_i) \cap \mathsf{coParity}(p)$. Consider a cell strategy for Player 1, and consider the counter-strategy for Player 2 that ensures that the game stays in A_i, and also ensures that $\mathsf{coParity}(p-2) \cup \mathsf{Reach}(A_i \cap \mathcal{C}_p(1))$ is satisfied. If a play visits $A_i \cap \mathcal{C}_p(1)$ only finitely many times, then from some point onwards it only visits cells in A_i that do not have priority 1 or priority 0 for the priority function p, and then $\mathsf{coParity}(p-2) = \mathsf{coParity}(p)$. Otherwise, the set $A_i \cap \mathcal{C}_p(1)$ is visited infinitely often and A_i is never left. Since A_i has no 0 priority cells for the priority function p, it means that Player 2 satisfies the $\mathsf{coParity}(p)$ objective. It follows that in A_i against all Player 1 cell strategies, Player 2 can ensure $\mathsf{Safe}(A_i) \cap \mathsf{coParity}(p)$. This is a contradiction to the fact that $A_i \subseteq W = \mathsf{Win}^W(\phi)$ and $\mathsf{Safe}(A_i) \cap \mathsf{coParity}(p) \subseteq \Gamma^\omega \setminus \phi$. This leads to the following theorem.

Theorem 5. *Given an imperfect-information game G with target $\mathcal{T} \subseteq \mathcal{C}$, safe set $\mathcal{F} \subseteq \mathcal{C}$ and a parity function p on Γ, Algorithm 1 computes $W = \mathsf{Win}^{\mathcal{C}}(\phi)$, where $\phi = \mathsf{Reach}(\mathcal{T}) \cup (\mathsf{Parity}(p) \cap \mathsf{Safe}(\mathcal{F}))$, and a winning cell strategy α on $W \setminus \mathcal{T}$ for ϕ.*

Proof. This follows from the correctness of the iteration, and the fact $W = W_i$ for some i, it follows that from all locations in W, the obtained cell strategy ensures

$$\mathsf{Reach}(W_0) \cup (\mathsf{Parity}(p) \cap \mathsf{Safe}(\mathcal{F}) \cap \mathsf{coBuchi}(\mathcal{F} \setminus \mathcal{C}_p(1)).$$

We now complete the argument by showing that the cell strategy is winning for ϕ. The cell strategy on W_0 ensures that $\mathcal{T}$ is reached from cells in W^*, from cells in $\mathcal{C}_p(0) \cap W$ it ensures to stay in W, and in all remaining cells in W_0 it ensures to reach $W^* \cup (\mathcal{C}_p(0) \cap W)$. The following case analysis completes the proof.

1. If the set W_0 is visited infinitely often, then (a) if W^* is reached, then $\mathcal{T}$ is reached; (b) otherwise $\mathcal{C}_p(0) \cap W$ is visited infinitely often and the game always stays safe in $W \setminus W^* \subseteq \mathcal{F}$. This ensures that $\mathsf{Parity}(p)$ is also satisfied.

2. If W_0 is visited only finitely often, then the play never reaches W^*, otherwise it would reach $\mathcal{T}$ and stay in $\mathcal{T}$ forever, and hence $\mathsf{Safe}(\mathcal{F})$ is satisfied, such that the objective $\mathsf{Parity}(p) \cap \mathsf{Safe}(\mathcal{F}) \cap \mathsf{coBuchi}(\mathcal{F} \setminus \mathcal{C}_p(1))$ is attained. Overall, it follows the objective ϕ is satisfied. ■

Antichain algorithm. To turn Algorithm 1 into an antichain algorithm, all set operations must preserve the downward-closed property. The union and intersection operations on sets preserve the downward-closed property of sets, but the complementation operation does not. Observe that Algorithm 1 performs complementation in line 9 ($A_i \leftarrow W \setminus W_i$) and uses the set A_i in lines 11 and 12. This was done for the ease of correctness proof of the algorithm. To see that the complementation step is not necessary, observe that

$$\mathsf{Reach}(W_i) \cup \big(\mathsf{Parity}(p-2) \cap \mathsf{Safe}(A_i \setminus \mathcal{C}_p(1))\big) = \mathsf{Reach}(W_i) \cup \big(\mathsf{Parity}(p-2) \cap \mathsf{Safe}(W \setminus \mathcal{C}_p(1))\big).$$

Indeed, if a play never visits W_i, then the play is in $\mathsf{Safe}(A_i \setminus \mathcal{C}_p(1))$ if, and only if, it is in $\mathsf{Safe}(W \setminus \mathcal{C}_p(1))$. Also note that the expression $\mathsf{Parity}(p-2) \cap \mathsf{Safe}(W \setminus \mathcal{C}_p(1))$ can be equivalently written as $\mathsf{Parity}(p-2) \cap \mathsf{Safe}(W \cap \bigcup_{i \geq 2} \mathcal{C}_p(i))$. It follows that every set operation in Algorithm 1 preserves downward-closed property. This demonstrates the following statement.

Theorem 6. *Algorithm 1 is compatible with the antichain representation.*

We remark that the explicit construction of the strategies takes place only in few steps of the algorithm: at line 2 and 3 of each recursive call where cell strategies are computed for reachability objectives, and in the base case (parity games with priorities $0, 1$ and 2) in line 11 where cell strategies are computed for union of safety and reachability objectives. Also note that we never need to compute strategies for the target set $\mathcal{T}$, and therefore in line 10, we would obtain strategies for the set $W_{i+1} \setminus W_i$. Hence, once the strategy is computed for a set, then it is never modified in any subsequent iteration.

5 Implementation

We have implemented Algorithm 1 in a prototype written in C. The input is a text-file description of the game structure, transitions and observations. Internally, transitions and sets of locations are represented as arrays of integers.

The building blocks of the algorithm are the computation of $\mathsf{CPre}(\cdot)$, and the two procedures $\mathsf{ReachOrSafe}$ and $\mathsf{ReachAndSafe}$. The implementation for $\mathsf{CPre}(q)$ follows Equation (1) using three nested loops over the sets Σ, Γ and q. In the worst case it may therefore be exponential in $|\Gamma|$ which is not avoidable in view of Lemma 4. To compute $\mathsf{ReachOrSafe}(\mathcal{T}, \mathcal{F})$, we evaluate the following fixpoint formula in the lattice of antichains: $\varphi_1 \equiv \nu X.(\mathcal{F} \sqcap \mathsf{CPre}(X)) \sqcup \mathcal{T}^*$ where $\mathcal{T}^* = \mu X.\mathsf{CPre}(X) \sqcup \mathcal{T}$. To compute $\mathsf{ReachAndSafe}(\mathcal{T}, \mathcal{F})$, we use $\varphi_2 \equiv \mu X.\mathcal{F} \sqcap (\mathsf{CPre}(X) \sqcup \mathcal{T})$.

When computing $q' = \mathsf{CPre}(q)$, we associate with each cell in the antichain q' the action to be played in order to ensure reaching a set in q. For φ_1, this information is sufficient to extract a winning strategy from the fixpoint: the action associated with each winning cell ensures to reach an element of the fixpoint, thus either confining the game inside $\mathcal{F}$ forever, or eventually reaching $\mathcal{T}^*$. On the other hand, for $\mathcal{T}^*$ and φ_2 (which has the flavor of reachability), we have seen in Section 3 that the final fixpoint is not sufficient to recover the winning strategy. Therefore, we have to construct on the fly the winning strategy while computing the fixpoint. We output a reachability strategy as a tree structure whose nodes are the sets in the successive antichains computed in the least-fixpoint iterations together with their associated action $\sigma \in \Sigma$. If $q' = \mathsf{CPre}(q)$ and σ is the action to be played in cell $s \in q'$, then for each observation o (given by Player 2) we know that there exists a cell $s_o \in q$ such that $\mathsf{post}(s) \cap \gamma(o) \subseteq s_o$. Correspondingly, each node for the sets in q' has $|\Gamma|$ outgoing edges to some sets in q.To evaluate the scalability of our algorithm, we have generated game structures and objectives randomly. We fixed the alphabet $\Sigma = \{0, 1\}$ and we used the following parameters to generate game instances: the *size* $|L|$ of the game, the *transition density* $r = \frac{|\Delta|}{|L|\cdot|\Sigma|}$, i.e., the average branching degree of the game graph, and the *density* $f = \frac{|\Gamma|}{|L|}$ *of observations*. For each $\sigma \in \Sigma$, we generate $r \cdot |L|$ pairs $(\ell, \ell') \in L \times L$ uniformly at random; each location is randomly assigned one of the $f \cdot |L|$ observations. We have tested reachability and Büchi objectives for games with transition density varying from 0.5 to 4 and density of observation varying from 0.1 to 0.9. We have limited the execution time to 10s for each instance. The size of the generated instances ranges from 50 to 500. For all values of the parameters, our prototype solved half of the instances of size 100 for both reachability and Büchi objectives. When the transition density is below 1.5, the instances are much easier to solve and the maximal size is 350 for reachability and 200 for Büchi objectives. Finally, we did not observe significant influence of the number of observations on the performance of the prototype. It seems that the exponential cost of computing $\mathsf{CPre}(\cdot)$ is compensated by the fact that for large number of observations, the games are closer to perfect-information games.

References

1. D. Berwanger, K. Chatterjee, L. Doyen, T. A. Henzinger, and S. Raje. Strategy construction for parity games with imperfect information. Technical Report MTC-REPORT-2008-005, http://infoscience.epfl.ch/record/125011, EPFL, 2008.
2. 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.
3. 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 2006*, LNCS 4144, pages 17–30. Springer-Verlag, 2006.
4. M. De Wulf, L. Doyen, N. Maquet, and J.-F. Raskin. Antichains: Alternative algorithms for LTL satisfiability and model-checking. In *Proc. of TACAS 2008*, LNCS 4693, pages 63–77. Springer-Verlag, 2008.
5. E. A. Emerson and C. S. Jutla. Tree automata, mu-calculus and determinacy. In *Proc. of FoCS 1991*, pages 368–377. IEEE, 1991.
6. D. Fudenberg and J. Tirole. *Game Theory*. MIT Press, 1991.
7. E. Grädel, W. Thomas, and T. Wilke, editors. *Automata, Logics, and Infinite Games*. LNCS 2500. Springer-Verlag, 2002.
8. T. A. Henzinger, R. Jhala, and R. Majumdar. Counterexample-guided control. In *Proc. of ICALP 2003*, LNCS 2719, pages 886–902. Springer-Verlag, 2003.
9. R. McNaughton. Infinite games played on finite graphs. *Annals of Pure and Applied Logic*, 65(2):149–184, 1993.
10. J. Reif. The complexity of two-player games of incomplete information. *Journal of Computer and System Sciences*, 29:274–301, 1984.
11. W. Thomas. On the synthesis of strategies in infinite games. In *Proc. of STACS 1995*, pages 1–13. Springer-Verlag, 1995.
12. W. Thomas. Languages, automata, and logic. *Handbook of Formal Languages*, 3:389–455, 1997.
13. W. Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200:135–183, 1998.