Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 225 (2009) 115–139

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Polynomial-Time Under-Approximation of Winning Regions in Parity Games

Adam Antonik,[1,2] Nathaniel Charlton,[3] and Michael Huth[4]

Department of Computing, South Kensington campus, Imperial College London, London, SW7 2AZ, United Kingdom

Abstract

We propose a pattern for designing algorithms that run in polynomial time by construction and under-approximate the winning regions of both players in parity games. This approximation is achieved by the interaction of finitely many aspects governed by a common ranking function, where the choice of aspects and ranking function instantiates the design pattern. Each aspect attempts to improve the under-approximation of winning regions or decrease the rank function by simplifying the structure of the parity game. Our design pattern is incremental as aspects may operate on the residual game of yet undecided nodes. We present several aspects and one higher-order transformation of our algorithms — based on efficient, static analyses — and illustrate the benefit of their interaction as well as their relative precision within pattern instantiations. Instantiations of our design pattern can be applied for local model checking and as pre-processors for algorithms whose worst-case running time is exponential. This design pattern and its aspects have already been implemented in [16].

Keywords: parity games, abstraction, computational complexity, algorithms

1 Introduction

A parity game G (e.g. [7]) specifies sets of finite or infinite plays between two players 0 and 1 on directed graphs (V, E) with non-empty, finite[5] set of nodes V and edge relation $E \subseteq V \times V$. Each node $v \in V$ is labeled with a priority $\chi(v)$, a value in the set $\{0, 1, \ldots, d-1\}$ where $d \geq 0$ is the index of parity game G. Furthermore, the set of nodes V is partitioned into V_0, the set of nodes owned by player 0, and

[1] We acknowledge the kind support of this work by the UK Engineering and Physical Sciences Research Council under grants EP/D50595X/1 *Efficient Specification Pattern Library for Model Validation* and EP/E028985/1 *Complete and Efficient Checks for Branching-Time Abstractions*

[2] Email: aa1001@doc.imperial.ac.uk

[3] Email: nac103@doc.imperial.ac.uk

[4] Email: M.Huth@doc.imperial.ac.uk

[5] We restrict our attention to finite parity games in this paper, although our ideas should transfer to computable settings for infinite parity games.

doi:10.1016/j.entcs.2008.12.070

V_1, those nodes owned by player 1.[6] Plays $\pi = v_0 v_1 \dots$ in game G are sequences of configurations (which are simply nodes) and may start at any node $v_0 \in V$:

- The player who owns v_0 needs to choose some v_1 with $(v_0, v_1) \in E$ as the next configuration. If no such v_1 exists, we then call v_0 a *dead end*, the player who owns v_0 loses that play.
- From configuration v_1 the play proceeds as in the previous item with v_1 now taking the role of v_0, resulting either in an infinite play, or a finite play if one player gets stuck in a dead end and loses the play.

The winning conditions for an infinite play $\pi = v_0 v_1 \dots$ are derived from the priorities it accumulates. Let $\mathsf{Inf}(\pi)$ be the set of those priorities that occur infinitely often in the sequence $\chi(v_0)\chi(v_1)\dots$. Then play π is won by player 0 if $\max \mathsf{Inf}(\pi)$ is even. Play π is won by player 1 if $\max \mathsf{Inf}(\pi)$ is odd.

Example 1.1 Consider the parity game in Fig. 1. A possible infinite play π is $v_7 v_0 v_7 v_0 \dots = (v_7 v_0)^\omega$ with $\mathsf{Inf}(\pi) = \{0, 1\}$. Thus π is won by player 1 as the largest priority that occurs infinitely often in that play is 1 and therefore odd.

As customary, we write σ to denote any of the players or values 0 and 1 and set $\bar{\sigma} = 1 - \sigma$. Memoryless strategies for player σ are partial functions $f \colon W \cap V_\sigma \to V$ for some subset W of nodes where $f(v)$ is defined iff $v \in W \cap V_\sigma$ is not a dead end; in which case $(v, f(v)) \in E$. Such a strategy is winning for W if all plays starting in W are won by player σ if played according to that strategy: at any configuration $v \in W \cap V_\sigma$ that is not a dead end, player σ chooses $f(v)$ as the next configuration.

Parity games are determined [6]: each player σ has a winning region $\mathsf{Win}_\sigma(G)$ of nodes in G for which she has a memoryless winning strategy, and these two regions $\mathsf{Win}_0(G)$ and $\mathsf{Win}_1(G)$ form a partition of the set of nodes in G.

Example 1.2 Reconsider the parity game in Fig. 1. The winning region for player 0 is $\{v_4, v_5, v_6\}$, and the winning region for player 1 is $\{v_0, v_1, v_2, v_3, v_7\}$; both sets partition V. A memoryless winning strategy for player 0 consists of "If at node v_6 or at node v_4, move to node v_5." Note that player 0's behavior at nodes v_0 and v_2 is irrelevant as these nodes are within the winning region of player 1.

Designing algorithms for the computation of winning regions in parity games is an important theoretical problem as the corresponding decision problem "is a node won by player 0?" is in UP and coUP [12] but not known to be in P. Designing such algorithms is also important for applications since determining winning regions in parity games is equivalent, in polynomial time, to other important problems — we mention model checking formulas of the modal mu-calculus [9,7].

Traditional approaches design algorithms that compute the exact winning regions of parity games, see e.g. the survey in [8]. These algorithms are then either revealed not to be in P — by constructing defeating, worst-case input games — or it is presently not known whether they are in P. In this paper we take a comple-

[6] In this paper a partition of a non-empty set Y is a set of pairwise disjoint sets $\{P_i \mid i \in I\}$ such that Y is the union of all P_i but where some P_i may be empty.

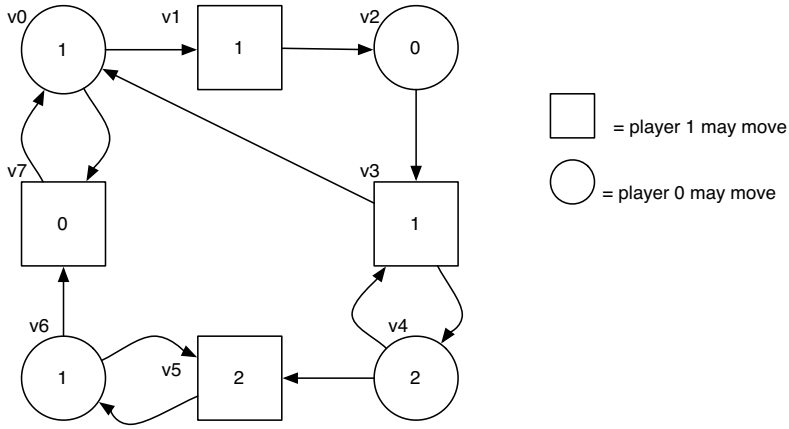

Fig. 1. A parity game, taken from [10]. Circled (resp. squared) nodes are owned by player 0 (resp. 1). Directed edges indicate possible game moves. Priorities of nodes are provided within nodes, e.g. $\chi(v_6) = 1$.

mentary approach. Instead of designing an algorithm that computes exact winning regions but has complexity that is unknown or known not to be in P, we present a design pattern for algorithms that are in P by construction but compute only subsets of winning regions by partitioning the set of nodes into three regions:

- known wins of player 0: a set W_0 contained in $\mathsf{Win}_0(G)$
- known wins of player 1: a set W_1 contained in $\mathsf{Win}_1(G)$
- and nodes whose winning status is unknown: the set $V \setminus (W_0 \cup W_1)$.

That is to say, we mean to under-approximate both winning regions $\mathsf{Win}_0(G)$ and $\mathsf{Win}_1(G)$, and propose to do so through efficient and sound static analyses.

Our approach has practical relevance for at least two reasons.

(i) In local model checking [15] one is only interested in whether a particular state satisfies a formula. This corresponds to determining which player wins a particular node, something our algorithms may well achieve.

(ii) Our algorithms can be used as efficient pre-processors for existing algorithms: due to the invariants satisfied by W_σ (detailed in Sec. 3), existing algorithms can run directly on the residual parity game induced by $V \setminus (W_0 \cup W_1)$.

Related to the last point, our invariants guarantee that our design pattern can operate in a completely incremental way by ignoring $W_0 \cup W_1$ in the continuation of computation. We refer to [7,17] for more background material on parity games but strive to make this article self-contained.

Outline of paper.

In Section 2 we recall some basic concepts for parity games. Our design pattern, its invariants, and correctness are discussed in Section 3. Three aspects, with which our design pattern can be instantiated, are presented and proved to be correct in Section 4. In Section 5, the beneficial interaction of aspects are investigated and a

precision-enhancing transformation for our algorithms are proposed. Related work is featured in Section 6. Plans for future work are revealed in Section 7, and we conclude in Section 8.

2 Basic concepts

A set $Y \subseteq V$ of nodes determines its *σ-attractor* in G, denoted by $\mathsf{Attr}_\sigma(G, Y)$, the set of those nodes $v \in V$ for which player σ has a memoryless strategy ensuring that all plays started at v either reach a node in Y or a dead end for player $\bar{\sigma}$. In particular, $Y \subseteq \mathsf{Attr}_\sigma(G, Y)$. We call Y a σ-attractor in G iff $Y = \mathsf{Attr}_\sigma(G, Y)$. Such σ-attractors Y satisfy the following, characterizing, closure properties: if $v \in V_\sigma$ and $(v, v') \in E$ for some $v' \in Y$, then $v \in Y$. Dually, if $v \in V_{\bar{\sigma}}$ and for all $(v, v') \in E$ we have $v' \in V$, then $v \in Y$.

Example 2.1 The attractor $\mathsf{Attr}_\sigma(G, \{\})$ consists of those nodes v from which player σ can ensure that a dead end for player $\bar{\sigma}$ is reached. These attractors are empty for the parity game in Fig. 1 as no dead ends are present. In that parity game, we have $\mathsf{Attr}_0(G, \{v_4, v_5\}) = \{v_4, v_5, v_6\}$.

A set of nodes $T \subseteq V$ is called a *σ-trap in G* iff for all nodes $v \in T \cap V_\sigma$ and all $(v, v') \in E$ we have $v' \in T$; and for all nodes $w \in T \cap V_{\bar{\sigma}}$ we have some $(w, w') \in E$ with $w' \in T$. It is easy to establish that, for a σ-attractor Y, the complement $V \setminus Y$ is a σ-trap in G.

Example 2.2 For G in Fig. 1, the set $\{v_4, v_5, v_6\}$ is a 1-trap but $\{v_4, v_5, v_6, v_7\}$ is not: player 1 can escape that set through $(v_7, v_0) \in E$ as $v_7 \in V_1$.

A set of nodes $P \subseteq V$ is called a *σ-paradise in G* iff P is a $\bar{\sigma}$-trap in G and player σ has a memoryless strategy$f\colon P \cap V_\sigma \to V$ such that she wins all plays beginning in any node in P if played according to strategy f. It is not hard to see that $\mathsf{Attr}_\sigma(G, P)$ is a σ-paradise in G whenever this is so for P. The union of σ-paradises is a σ-paradise so $\mathsf{Win}_\sigma(G)$ is the largest σ-paradise in G.

Example 2.3 Sets $\{\}$, $\{v_5, v_6\}$, and $\{v_4, v_5, v_6\}$ are 0-paradises for G in Fig. 1.

For a set of nodes $W \subseteq V$, the parity game $G[W]$ has W as set of nodes, $(W \times W) \cap E$ as edges, and χ restricted to domain W as priority function. We say that $G[W]$ is a *sub-game of G* iff all $w \in W$ that are dead ends in $G[W]$ are also dead ends in G. It is easy to see that $G[W]$ is a sub-game in G whenever W is a σ-trap in G, for any player σ.

Example 2.4 For G from Fig. 1, $G[\{v_4, v_5, v_6\}]$ is a sub-game in G but $G[\{v_0, v_1\}]$ is not: v_1 is a dead end for player 1 in $G[\{v_0, v_1\}]$ but not in G.

A cycle in a directed graph (V, E) is a finite word $v_0 v_1 \dots v_n$ over V with $n \geq 1$ such that $v_0 = v_n$ and, for all $i = 0, \dots, n-1$, we have $(v_i, v_{i+1}) \in E$. In particular, self-loops at v — which are edges $(v, v) \in E$ — determine a cycle vv.

The work reported in this paper benefits from an explicit representation of redundancies in parity games. This leads to the notion of *lax* parity games.

Definition 2.5 A lax parity game with index d is a parity game G with index d, with two adjustments and one requirement:

- The function χ maps nodes v to elements of $\{0, 1, \ldots, d-1\} \cup \{X\}$, where X is a priority expressing "don't care".
- For an infinite play $\pi = v_0 v_1 \ldots$ in G, the set $\mathsf{Inf}(\pi)$ is defined as all those priorities *other than* X that occur in $\chi(v_0)\chi(v_1)\ldots$ infinitely often.
- We then require that $\mathsf{Inf}(\pi)$ is non-empty for all infinite plays π in G.

Example 2.6 Fig. 6 and 7 depict lax parity games. The parity game in Fig. 10 can be turned into an equivalent lax parity game by changing the priorities of nodes v_0 and v_2 to X. No further X labels are then allowed as otherwise there will be infinite plays π with empty set $\mathsf{Inf}(\pi)$.

All notions discussed for parity games, in particular strategies and winning regions, apply verbatim to lax parity games. Clearly, every parity game is a lax parity game where X is not in the image of χ. Conversely, for any lax parity game G_l we can construct, in linear time and logarithmic space, an equivalent parity game G that is simply G_l except that all nodes that were labeled with priority X are now labeled with 0. It is easy to see that G and G_l have the same winning regions. In particular, lax parity games are determined. By abuse of language we will refer to lax parity games as parity games subsequently, unless there is a specific need to highlight laxness. The evaluation of maxima and sums containing X, e.g. $\max\{0, 3, X\} = 3$ and $0 + 2 + 1 + X + 3 = 6$, treats X as being 0.

Using standard techniques we can transform, in logarithmic space and linear time, a parity game G with node set V into a parity game G' with node set V' containing V such that G has no dead ends, no self-loops, and $\mathsf{Win}_\sigma(G) = V \cap \mathsf{Win}_\sigma(G')$; see Sec. A of the appendix. Due to their simplicity, these transformations won't increase the computational burden on our algorithms. Thus we have:

Assumption of paper.
Without loss of generality parity games have no dead ends and no self-loops.

3 Design pattern

Our design pattern is depicted in Fig. 2. The design pattern first initializes W_0 and W_1 to $\{\}$. Next a cache for the rank function of the parity game is set to be 0. The while-statement that follows employs a standard termination pattern for the rank function. Its body iterates the execution of $k > 0$ many aspects `A#i` that run in polynomial time and potentially increase the sets W_σ, or decrease the rank of G in other ways. When that while-statement terminates (when no aspect decreases the rank), the final value of W_σ represents those nodes that are known to be won by player σ, whereas $V \setminus (W_0 \cup W_1)$ contains those nodes whose winning status

```
W_0 = W_1 = {};
cache = 0;
while (rank(G) != cache) {
  cache = rank(G);
  for (i=1; i++; i <= k) { A#i; }
}
return (W0,W1);
```

Fig. 2. Design pattern for algorithms that under-approximate winning regions of parity games in polynomial time. Instantiations of this pattern require a rank function that is polynomial in $|E|$, $|V|$, and d — where we write $|Y|$ for the cardinality of set Y — and $k \geq 0$ many aspects `A#i` that run in polynomial time and may soundly increment sets W_σ or decrease `rank(G)`.

the (instantiation of the) design pattern cannot classify. See Fig. 3 for illustration. This pattern is inspired by Zielonka's constructive proof of determinacy for parity games [17], except that our pattern is symmetric and non-recursive.

Definition 3.1 Subsequently we write U for the set $V \setminus (W_0 \cup W_1)$ of nodes whose winning status is yet undecided, and E_U for the edge relation E intersected with $U \times U$. Also, $\chi^U = \{\chi(v) \mid v \in U\} \setminus \{X\}$ is the set of priorities other than X that occur in set U. All these notions have state in our design pattern.

We list the basic invariants for this pattern, which all instantiations need to honor before and after any iteration of the for-statement. If these iterations change state we use primes to denote state after an iteration. The four invariants for our design pattern are:

(P) W_σ is a σ-paradise in G, and so a $\bar{\sigma}$-trap in G

(A) W_σ is a σ-attractor in G and W_0 and W_1 are disjoint [7]

(L) $G[U]$ is lax (i.e. there are no cycles labeled with X only) and contains neither self-loops nor dead ends for either player

(F) $\mathsf{Win}_\sigma(G) = \mathsf{Win}_\sigma(G')$

Invariant (P), for "Paradise", states that W_σ is a σ-paradise in G and therefore must be contained in $\mathsf{Win}_\sigma(G)$, the largest σ-paradise in G; and a σ-paradise is a $\bar{\sigma}$-trap by definition. Invariant (A), for "Attractor", states that W_σ is a σ-attractor in G. Invariant (L), for "Loops", "Liveness", and "Laxness", states that the sub-graph of yet undecided nodes has no node that is a dead end or contains a self-loop, and that the sub-game for these undecided nodes is lax. Invariant (F), for "Frame Condition", states that the winning regions of parity game G don't change. This frame condition is a non-trivial constraint as the aspects `A#i` may change structure of G; e.g. add/remove edges or nodes, modify the priority function χ, etc. The initialization $W_\sigma = \{\}$ in our design pattern clearly secures all invariants.

Invariants (A) and (P) together guarantee that our design pattern can operate in a completely incremental manner: aspects can effectively ignore set $W_0 \cup W_1$ and its incoming and outgoing edges and execute on the residual game $G[U]$. Invariants (P), (A), and (F) refer to game G not to the incremental game $G[U]$ so our correctness

[7] This disjointness holds if invariant (P) is true, but some of our arguments only rely on invariant (A).

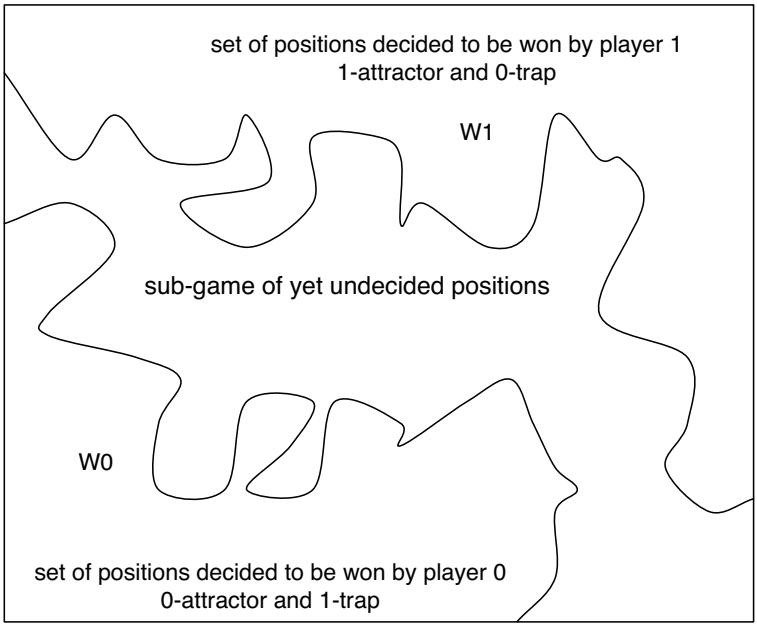

Fig. 3. Computing safe winning regions of parity games: the set V of nodes is partitioned into regions W_0, W_1, and the complement of their union. That partition satisfies invariants (P), (A), (L), and (F).

proofs will not always be incremental themselves. The next lemma establishes the facts needed for realizing such incrementality in our algorithms.

Lemma 3.2 *Let G satisfy invariant (A). Then $G[U]$ is a sub-game in G and*

$$\mathsf{Attr}_\sigma(G, W_\sigma \cup Z) = W_\sigma \cup \mathsf{Attr}_\sigma(G[U], Z) \qquad \forall Z \subseteq U \tag{1}$$

If, additionally, G satisfies invariant (P), then

$$\mathsf{Win}_\sigma(G[U]) = U \cap \mathsf{Win}_\sigma(G) \tag{2}$$

Proof. First, we show that $G[U]$ is a sub-game in G. Let $v \in U \cap V_\sigma$ such that there is no $(v, v') \in E$ with $v' \in U$. We need to show that there is no $(v, v') \in E$ at all.

- Proof by contradiction: Assume that there is some $v' \in V$ with $(v, v') \in E$. Since v is a dead end in $G[U]$ we have $v' \in W_\sigma \cup W_{\bar\sigma}$. We show that v' is in $W_{\bar\sigma}$.
  · Proof by contradiction: Assume $v' \in W_\sigma$. Then $v \in V_\sigma$ and $(v, v') \in E$ imply $v \in \mathsf{Attr}_\sigma(G, W_\sigma) = W_\sigma$ by invariant (A). This contradicts $v \in U$.
  Therefore all v' with $(v, v') \in E$ have to be in $W_{\bar\sigma}$. But then $v \in \mathsf{Attr}_\sigma(G, W_{\bar\sigma})$ follows, and the latter set is $W_{\bar\sigma}$ by invariant (A). This contradicts $v \in U$.

Second, we show (1). Let $Z \subseteq U$.

- Let $v \in \mathsf{Attr}_\sigma(G, W_\sigma \cup Z)$. We need to show that $v \in W_\sigma \cup \mathsf{Attr}_\sigma(G[U], Z)$. If $v \in W_\sigma$, this is the case. But v cannot be in $W_{\bar\sigma}$ as this is a σ-trap disjoint from $W_\sigma \cup Z$ by invariant (A). So the case $v \in U$ remains to be considered. By

assumption, player σ has a memoryless winning strategy f, in G, such that all plays from v in G that are played according to f either reach a dead end for player $\bar{\sigma}$ or a node in $W_\sigma \cup Z$.

Now consider any play in $G[U]$ from node v where player σ plays according to f. That such plays are played according to f is indeed possible since any node w that is reachable from v in $G[U]$ in such a play (including possibly v itself) and is in V_σ must satisfy that $f(v)$ is defined (as f is winning) and not in W_σ (as that set is a σ-attractor and $w \in U$). But then any such play in $G[U]$ looks to player σ like a play in G played according to f. Therefore any such play either reaches a dead end for player $\bar{\sigma}$ or a node in $W_\sigma \cup Z$. But since these plays are in $G[U]$, they never reach node set W_σ. This shows $v \in \mathsf{Attr}_\sigma(G[U], Z)$ as desired.

- Let $v \in W_\sigma \cup \mathsf{Attr}_\sigma(G[U], Z)$. We need to show $v \in \mathsf{Attr}_\sigma(G, W_\sigma \cup Z)$. If $v \in W_\sigma$, we are done as $Y \subseteq \mathsf{Attr}_\sigma(G, Y)$ for all $Y \subseteq V$. It remains to consider the case when $v \in \mathsf{Attr}_\sigma(G[U], Z)$. Then player σ can force, in $G[U]$, all plays from v to either reach a dead end for player $\bar{\sigma}$ or a node in Z. But then player σ can force, for all plays from v in the larger game G, to either reach a dead end for player $\bar{\sigma}$ or a node in $W_\sigma \cup Z$. This is so since
  - $W_{\bar{\sigma}}$ is a $\bar{\sigma}$-attractor in G by invariant (A) so no node in $U \cap V_{\bar{\sigma}}$ has an edge into $W_{\bar{\sigma}}$
  - and so player $\bar{\sigma}$ can avoid both, a dead end for him and node set Z, only by either player choosing an edge into W_σ.

  This shows $v \in \mathsf{Attr}_\sigma(G, W_\sigma \cup Z)$.

Third, we show (2). Since σ is 0 or 1 and since parity games are determined it suffices to show that $\mathsf{Win}_\sigma(G[U]) \subseteq \mathsf{Win}_\sigma(G)$. Let $v \in \mathsf{Win}_\sigma(G[U])$. Then $v \in U$. Consider any play $\pi = v_0 v_1 \ldots$ obtained when player σ plays a winning strategy for $v \in \mathsf{Win}_\sigma(G[U])$ in $G[U]$ (until and if π reaches $W_0 \cup W_1$ when player σ plays her winning strategy in W_σ and whichever way in $W_{\bar{\sigma}}$) and player $\bar{\sigma}$ plays any strategy in G. Then π cannot reach the set $W_0 \cup W_1$ unless π is won by player σ, as can be seen by a case analysis of the ownership of any node v_n reached in that play:

- If v_n is owned by player σ, then his winning strategy in $G[U]$ will pick some $v_{n+1} \in U$ with $(v_n, v_{n+1}) \in E$.
- If v_n is owned by player $\bar{\sigma}$, he also needs to pick some v_{n+1} with $(v_n, v_{n+1}) \in E$. If that chosen v_{n+1} is in W_σ, then π is won by player σ as she will trap π in its σ-paradise W_σ in G by invariant (P). On the other hand, v_{n+1} cannot be chosen from $W_{\bar{\sigma}}$ as this contradicts that $W_{\bar{\sigma}}$ is a $\bar{\sigma}$-attractor in G and $v_n \in U \cap V_{\bar{\sigma}}$.

In conclusion, π is won by player σ in U or in W_σ. □

The lemma above means that it suffices to analyze the winning status of yet undecided nodes $v \in U$ by performing these analyses in the sub-game $G[U]$, effectively ignoring the sets W_σ and their incoming and outgoing transitions. In particular, aspects that can increase W_σ maintain invariants (A) and (P) in a completely incremental manner by computing the right-hand side of (1) as new value of W_σ whenever $Z \subseteq U$ has been revealed as a σ-paradise in $G[U]$. We will follow this incremental

approach in aspect A_3 discussed below. This incremental approach will also allow an incremental computation and storage of memoryless winning strategies in our design pattern which, for sake of brevity, we will not discuss in this paper. Adding such functionality is a routine matter.

4 Aspects

Aspects only have to maintain the invariants (P), (A), (L), and (F) and ensure that their instantiations in our design pattern run in polynomial time. In this paper we explore a few salient aspects, based on efficient static analyses, which have one or more of the following features:

- An aspect may abstract away the role of players by exploiting the cyclic structure in the directed graph (U, E_U) labeled by χ in order to change and simplify χ.
- An aspect may abstract, soundly for wins of one player, sets of priority values into a single priority value in order to increase W_σ.
- An aspect may be deterministic or contain non-deterministic choices. In the latter case, a scheduler is needed to resolve such non-determinism.

Aspect A_1

Aspect A_1 ignores the role of players and exploits cycles in the directed graph (U, E_U) to modify the priority function χ. It checks, for each node $v \in U$ with $2 \leq \chi(v) \neq X$, whether no cycle in (U, E_U) through v contains some node w with $\chi(w) = \chi(v) - 1$. If there is indeed no such cycle, it decrements $\chi(v)$ to $\chi(v) - 2$. This aspect gets rid of "gaps" in χ^U. For example, $\chi^U = \{0, 1, 3\}$, which may occur on-the-fly, would change eventually to $\{0, 1\}$.

Example 4.1 • For G in Fig. 1 with $W_\sigma = \{\}$ there are only two nodes v_4 and v_5 with priorities larger than 1. But their priority is 2 and each of these nodes has a cycle on which priority 1 occurs. Thus aspect A_1 has no effect on G.

- Consider the parity game G in Fig. 4. Only node v_1 has priority strictly larger than 1, namely 3. Since there is no node with priority 2, aspect A_1 re-sets $\chi(v_1)$ to 1 on empty W_σ, reducing the index of G.

We prove the correctness of aspect A_1.

Theorem 4.2 *Let G satisfy invariants (P), (A), (L), and (F). Any decrement of χ according to aspect A_1 maintains these invariants.*

Proof. Let G satisfy the invariants (P), (A), (L), and (F). Let v be a node in U such that no cycle in (U, E_U) through v contains a node with priority $\chi(v) - 1$. Let G' be the game resulting from G by assigning priority $\chi(v) - 2$ to v and leaving G unchanged in all other regards. We have to prove that the invariants still hold.

For invariant (P), since the game graph of G' is that of G and since χ did not change within W_σ we infer that W_σ is (still) a σ-paradise in G and in G'. A similar reasoning applies to invariants (A) and (L). (For example, $G'[U]$ is a lax parity game

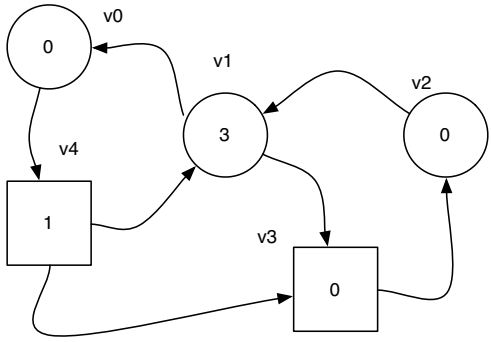

Fig. 4. A parity game G for which aspect A_1 re-sets $\chi(v_1)$ from 3 to 1, thereby decreasing the index of G.

since $G[U]$ is, the game graph of $G'[U]$ is that of $G[U]$, and the inverse image of X is the same for χ and χ'.)

It remains to show invariant (F), i.e. $\mathsf{Win}_\sigma(G') = \mathsf{Win}_\sigma(G)$. Since invariant (P) holds for G', it suffices to show that $\mathsf{Win}_\sigma(G'[U]) = \mathsf{Win}_\sigma(G[U])$ by (2). Let $p_v = \chi(v) \mod 2$. Since aspect A_1 only made $\chi(v)$ smaller by 2, any node z that is won by player $\overline{p_v}$ in $G[U]$ is certainly still won by that player in $G[U]'$. Since parity games are determined it therefore suffices to show that any node z that is won by player p_v in $G[U]$ is still won by player p_v in game $G'[U]$.

Proof by contradiction: Let z be won by player $\overline{p_v}$ in game $G'[U]$ and let π be the play starting in z and obtained when player p_v plays her strategy for winning z in $G[U]$, and player $\overline{p_v}$ plays his strategy for winning z in $G'[U]$. Since the only change from $G[U]$ to $G'[U]$ is a decrement of χ by 2 at node v, this therefore means that $\chi(v) - 1$ has to be in $\mathsf{Inf}(\pi)$ in $G[U]$; otherwise player $\overline{p_v}$ could not win π in $G'[U]$. But then $\chi(v) - 1$ is the maximal parity in $\mathsf{Inf}(\pi)$ in $G'[U]$ and so the maximal parity in $\mathsf{Inf}(\pi)$ in $G[U]$ has to be $\chi(v)$ and occurs only since v occurs infinitely often in π; otherwise player p_v would not win π in $G[U]$ and player $\overline{p_v}$ could not win π in $G'[U]$. To summarize, there is an infinite play π in U from z such that v occurs infinitely often in π and $\chi(v) - 1$ is in $\mathsf{Inf}(\pi)$. This implies that there is a cycle C in (U, E_U) through v containing a node w_C satisfying $\chi(w_C) = \chi(v) - 1$, a contradiction to the application criterion for aspect A_1 at node v in G. □

We remark that aspect A_1 can be implemented to run in polynomial time. For each $v \in V$ with $2 \leq \chi(v) \neq X$, consider the set

$$T = \{w \in U \mid \chi(w) = \chi(v) - 1\}$$

We then have to check, in (U, E_U), whether there is some $t \in T$ that is reachable from v; and from which v is in turn reachable in (U, E_U). If there is no such $t \in T$, we can decrement the priority of v by 2.

Aspect A_2

Aspect A_2 asks whether, for any $v \in U$ with $\chi(v) \neq X$, all cycles C in (U, E_U) through v have some $w_C \neq v$ in C with $\chi(v) \leq \chi(w_C) \neq X$. If so, aspect A_2 re-sets

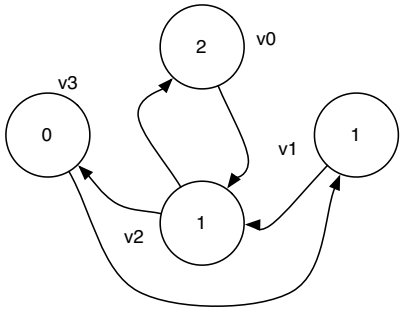

Fig. 5. A parity game for which aspect A_2 has non-deterministic choices that result in different lax parity games upon the scheduling of execution for this aspect.

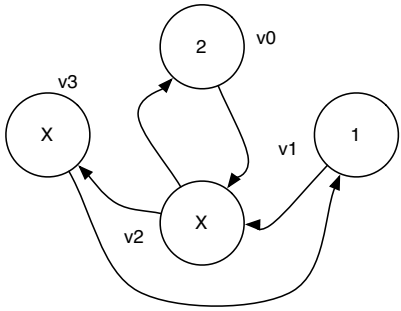

Fig. 6. A lax parity game computed by aspect A_2 when it prefers to change the priority at v_2 over changing that of v_1. Changing one preempts any subsequent change of the other.

$\chi(v)$ to X.[8]

Example 4.3 • Reconsider the parity game G in Fig. 4. Since all cycles in the game graph of G have to pass through v_1, aspect A_2 computes the same lax parity game G' regardless of the order in which priorities are re-set to X. In G' all states except v_1 have priority X. (Since $\chi(v_1) \mod 2 = 1$, this means $V = \mathsf{Win}_1(G)$.)

- Non-determinism of aspect A_2 is not always confluent. Applying aspect A_2 to the parity game in Fig. 5 results in the parity games depicted in Fig. 6 and 7.

We prove the correctness of aspect A_2.

Theorem 4.4 *Let G satisfy invariants (P), (A), (L), and (F). Any change of χ in aspect A_2 maintains these invariants.*

Proof. Let G satisfy the invariants (P), (A), (L), and (F). Let v be a node in U with $\chi(v) \neq X$ such that all cycles C in (U, E_U) through v contain some node $w_C \neq v$ with $\chi(v) \leq \chi(w_C) \neq X$. Let G' be the game resulting from G by assigning priority X to v and leaving G unchanged in all other regards. We have to prove

[8] This aspect has a strict version $A_2^<$ based on $\chi(v) < \chi(w_C) \neq X$ instead of $\chi(v) \leq \chi(w_C) \neq X$ but defined like A_2 in all other regards. We will not discuss $A_2^<$ further in this paper.

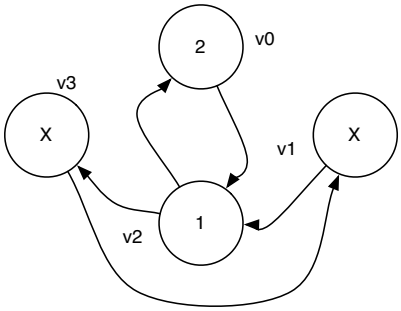

Fig. 7. A lax parity game, different from the one in Fig. 6, computed by aspect A_2 when it prefers to change the priority at v_1 over changing that of v_2.

that the invariants still hold. For invariant (P) and (A), and for the dead-end and self-loop part of invariant (L) we reason in the same manner as for aspect A_1.

For the remaining part of invariant (L), the laxness of $G[U]$, consider any infinite play π in $G[U]$. If v does not occur infinitely often in π, then $\mathsf{Inf}(\pi)$ is the same set for $G[U]$ and for $G'[U]$ and so non-empty as $G[U]$ is lax. If v does occur infinitely often in π, then by assumption on the applicability of aspect A_2 at v there is a cycle C through v contained in π that has some $w_C \neq v$ in C with $\chi(v) \leq \chi(w_C) \neq X$ in G. So $\chi(w_C)$ is in $\mathsf{Inf}(\pi)$ in $G[U]$ and in $G'[U]$.

For invariant (F), we reason as for aspect A_1 and so it suffices, by (2), to show that $\mathsf{Win}_\sigma(G'[U]) = \mathsf{Win}_\sigma(G[U])$. Let $z \in \mathsf{Win}_\sigma(G[U])$. Since $\sigma \in \{0,1\}$ and since parity games are determined it remains to show $z \in \mathsf{Win}_\sigma(G'[U])$. Consider π, the play starting in z and obtained when player σ plays her strategy for winning z in $G[U]$, and player $\bar{\sigma}$ plays any strategy. It remains to show that π is won by player σ in $G'[U]$. By invariant (L) for $G'[U]$, the play π is infinite and in U. Since the change from G to G' concerns only the priority of node v, we are done if v does not occur infinitely often in π. Otherwise v occurs infinitely often in π. But then, since π is in U, there is a cycle C through v in (U, E_U) such that all of its nodes are on π. By the application criterion for aspect A_2 at v there is some $w_C \neq v$ in C with $\chi(v) \leq \chi(w_C) \neq X$. But then $\chi(v)$ and $\chi(w_C)$ are in $\mathsf{Inf}(\pi)$ in $G[U]$ and so setting the priority of v to X does not affect the winning condition for π in $G'[U]$ as $v \neq w_C$ and the priority value of v in $G[U]$ has $\chi(w_C)$ as an upper bound in $G[U]$.□

Aspect A_2 can be implemented to run in polynomial time by checking, for $v \in U$, the negation of this aspect's applicability criterion: that there is a cycle C in (U, E_U) through v such that all nodes $w \neq v$ in C satisfy "$\chi(w) = X$ or $\chi(w) < \chi(v)$". By invariant (L) the edge (v, v) is not in E so this can be reduced to a reachability analysis that checks whether there is a cycle through v in the full subgraph of (U, E_U) for node set

$$\{v\} \cup \{w \in U \mid \chi(w) = X \text{ or } \chi(w) < \chi(v)\}$$

```
n := max { chi(v) | v in U };
s := n mod 2;
for (k = 0; k++; 2*k <= n) {
  Z_s = checkInfFin({n,n-2,...,n-2*k}, {n-1,n-3,...,n-2*k+1}, s);
  if (Z_s != {}) { W_s := W_s + Attr_s(G[U], Z_s); }
  Z_{1-s} = checkInfFin({n-1,n-3,...,n-2*k-1}, {n,n-2,...,n-2*k}, 1-s);
  if (Z_{1-s} != {}) { W_{1-s} := W_{1-s} + Attr_{1-s}(G[U], Z_{1-s}); }
```

Fig. 8. Pseudo-code for aspect A_3. It computes maximal priority n occurring in U and its parity s. Then it has $\lfloor n/2 \rfloor$ iterations in which intervals of odd (resp. even) priorities are abstracted into a single odd (resp. even) priority and fed to a parity game `checkInfFin` over U with index 3, of which only the non-empty winning region for its argument player $\sigma(=\mathtt{s})$ is added to W_σ and closed under σ-attraction in G.

Aspect A_3

Aspects A_1 and A_2 may simplify χ but won't change W_σ themselves. In an attempt to increase W_σ, aspect A_3 retains the role of players in the game graph (U, E_U) but abstracts intervals of odd (resp. even) priorities into an odd (resp. even) priority and solves, in polynomial time, a game for each such abstraction. Its pseudo-code is given in Fig. 8 where `+` denotes set union, `{x,..,y}` is interpreted as {} whenever $x < y$, and the method `checkInfFin(I,F,p)` returns those nodes in U for which player $p \in \{0, 1\}$ can win all plays in the game graph (U, E_U) of sub-game $G[U]$ with the new winning condition

"priorities set I is met infinitely often, and priorities set F only finitely often"

for all infinite plays. This can be expressed as a parity game of index 3 over that game graph (U, E_U).[9] Note the incremental computation of W_σ in aspect A_3 as $W_\sigma \cup \mathsf{Attr}_\sigma(G[U], Z_\sigma)$, which is sound due to (1) .

Example 4.5 For G in Fig. 1 let $U = V$ and $W_\sigma = \{\}$ initially. We execute the pseudo-code for aspect A_3. The maximal priority in U is $n = 2$ and its parity $s = 0$. The for-statement has two iterations. For $k = 0$, two games are being played:

- In game `checkInfFin({2},{},0)` the set Z_0 gets assigned those nodes in U from which player 0 can force priority 2 to occur infinitely often in the game graph of $G[U] = G$. Then $Z_0 = \{v_4, v_5, v_6\}$ follows and W_0 is the union of {} and $\mathsf{Attr}_0(G[U], \{v_4, v_5, v_6\})$, which is $\{v_4, v_5, v_6\}$ and equals $\mathsf{Win}_0(G)$.
- In game `checkInfFin({1},{2},1)` the set Z_1 is comprised of those nodes where player 1 can force 1 to occur infinitely often while, at the same time, ensuring that 2 only occurs finitely often in the sub-game $G[U]$ which now equals $G[V \setminus \{v_4, v_5, v_6\}] = G[\{v_0, v_1, v_2, v_3, v_7\}]$. Since priority 2 does not occur in $G[U]$, and since $1 \in \mathsf{Inf}(\pi)$ for all plays π is $G[U]$, we get $Z_1 = \{v_0, v_1, v_2, v_3, v_7\}$ and so W_1 computes to $\{v_0, v_1, v_2, v_3, v_7\} = \mathsf{Win}_1(G)$.

Therefore further iterations won't add anything.[10]

[9] The choice of index 3 is somewhat ad-hoc as similar versions of aspect A_3 for any fixed and finite index $i > 3$ can be designed. Alternatively, one can "tune up" A_3 with `probe` of Sec. 5.

[10] This paper will not discuss any optimizations of algorithms as this is best left for future work on implementation and experimentation with this design pattern. For example, aspect A_3 may check itself whether `cache` equals `rank(G)` and, if so, terminate global execution.

Note that it is sound not to re-compute `n` and `s` in, or in between, iterations with aspect A_3 when U has become smaller, as seen in Example 4.5. We illustrate aspect A_3 with a more complex example.

Example 4.6 The algorithm in [13] is among the most efficient ones known for solving parity games. Jurdziński demonstrated in loc. cit. that this algorithm has exponential running time in the worst case. Fig. 9 shows an instance of his parameterized family of worst-case input games (in our paper presented with a maximum parity acceptance condition), the parity game $H_{3,4}$ of loc. cit. The maximal priority occurring in the initial $U = V$ is $n = 7$. Its parity is $s = 1$.

- First iteration, $k = 0$:
  - In game `checkInfFin({7},{},1)` player 1 cannot win any nodes in U as player 0 can simply move to the right from any node v_4, v_6, and v_8.
  - In game `checkInfFin({6},{7},0)` player 0 wins nodes $v_0, \ldots, v_9$ as player 0 can simply move to the right from any node v_4, v_6, and v_8. Thus W_0 becomes $\{\} \cup \mathsf{Attr}_0(G[V], \{v_0, \ldots, v_9\}) = \{v_0, \ldots, v_9\}$.
- Second iteration, $k = 1$:
  - In game `checkInfFin({7,5},{6},1)` player 1 cannot win any nodes in $U = \{v_{10}, \ldots, v_{36}\}$ as player 0 can simply move to the right from any node v_{21}, v_{23}, and v_{25}.
  - In game `checkInfFin({6,4},{7,5},0)` player 0 wins nodes $v_{20}, \ldots, v_{26}$ in U as player 0 can simply move to the right from any node v_{21}, v_{23}, and v_{25}. Thus W_0 becomes $\{v_0, \ldots, v_9\} \cup \{v_{20}, \ldots, v_{26}\}$.
- Third iteration, $k = 2$:
  - In game `checkInfFin({7,5,3},{6,4},1)` player 1 cannot win any nodes in $U = \{v_{10}, \ldots, v_{19}\} \cup \{v_{27}, \ldots, v_{36}\}$ as player 0 can simply move to the right from any node v_{31}, v_{33}, and v_{35}.
  - In game `checkInfFin({6,4,2},{7,5,3},0)` player 0 wins nodes $v_{30}, \ldots, v_{36}$ in U as player 0 can simply move to the right from any node v_{31}, v_{33}, and v_{35}. Thus W_0 becomes $V \setminus \{v_{10}, \ldots, v_{16}\}$ and U becomes $\{v_{10}, \ldots, v_{16}\}$.
- Fourth iteration, $k = 3$:
  - In game `checkInfFin({7,5,3,1},{6,4,2},1)` player 1 wins all nodes in $U = \{v_{10}, \ldots, v_{16}\}$ as player 1 can simply move to the right from any node v_{11}, v_{13}, and v_{15}. So W_1 becomes $\{v_{10}, \ldots, v_{16}\}$ and U becomes $\{\}$, meaning that aspect A_3 cannot compute anything further and solves that game completely.

We show the correctness of aspect A_3.

Theorem 4.7 *Let G satisfy the invariants (P), (A), (L), and (F). Then the execution of aspect A_3 maintains all of these invariants.*

Proof. Let G satisfy the invariants (P), (A), (L), and (F) and let G' be the result of executing aspect A_3 on G.

For invariant (P), by Lemma 3.2 it suffices to show that all sets Z_σ computed in A_3 are σ-paradises in $G[U]$. This is so since W_σ satisfies (P) and since σ-paradises

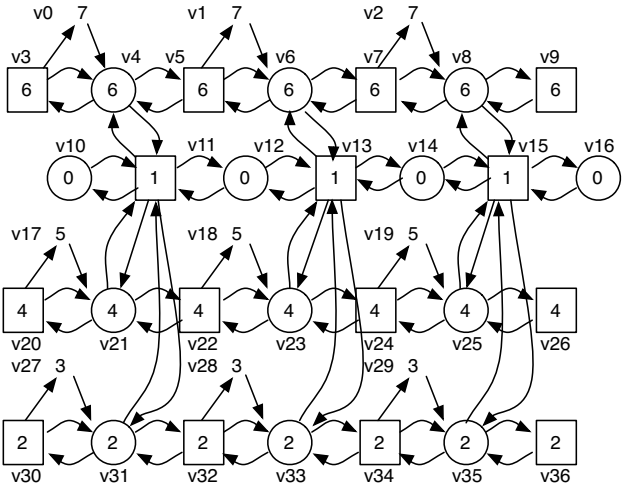

Fig. 9. A parity game $H_{3,4}$ for which the algorithm in [13] has exponential running time, as shown in loc. cit.. Nodes with priority values 3, 5, and 7 omit ownership of players as $\mathsf{Win}_\sigma(H_{3,4})$ isn't impacted by this.

in G are closed under unions and σ-attractors in G. Let $0 \leq 2k \leq n$.

- Let $v \in Z_s$ for the Z_s computed for that value of k. Then player s has a memoryless strategy for which all plays π in $G[U]$ beginning at v and following that strategy satisfy $\mathsf{Inf}(\pi) \cap \{n, n-2, \ldots, n-2k\} \neq \{\}$, $\mathsf{Inf}(\pi) \cap \{n-1, n-3, n-2k+1\} = \{\}$, and trap player $\bar{s}$ in set Z_s. Therefore, all these plays are won by player s in $G[U]$.
- A similar reasoning applies to $v \in Z_{\bar{s}}$ to show that $Z_{\bar{s}}$ is contained in $\mathsf{Win}_{\bar{s}}(G[U])$.

Invariant (A) is enforced by appealing to (1). Invariants (L) and (F) hold as the game graph and priority function of G won't change.

□

Note that aspect A_3 has a straightforward implementation in polynomial time as it involves at most n games `checkInfFin` — parity games of bounded index 3 — and n computations of σ-attractors in $G[U]$.

5 Interaction

By abuse of notation, we write $A_{i_1} A_{i_2} \ldots A_{i_k}$ for the algorithm that instantiates our design pattern with each `A#i` being A_{i_k}. In particular, `A#i` may equal `A#j` if $i \neq j$. We write $\{A_{i_1} A_{i_2} \ldots A_{i_k}\}$ if we mean any of the $k!$ algorithms obtained by permutations of these aspects A_{i_j}.

Fundamental questions are whether the interaction of these aspects commutes, whether swapping two aspects is in some sense confluent, and whether certain aspects can't aid the progress of certain others. These questions are similar to the "phase ordering problem" in the design of optimizing compilers, and won't be addressed further in this paper due to space limitations.

For each instantiation, a suitable rank function has to be determined. A rank

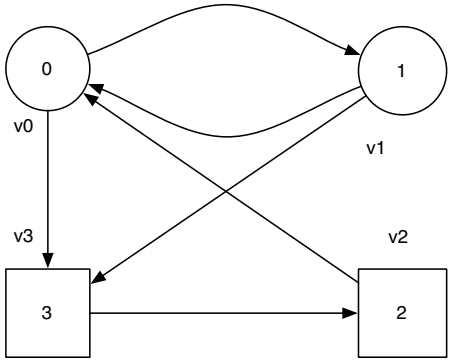

Fig. 10. A parity game for which the instantiation A_3 of our design pattern returns $W_\sigma = \{\}$ but for which the instantiations $\{A_2A_3\}$ solve the game completely.

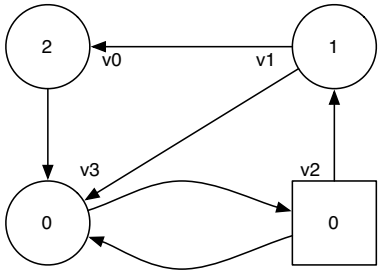

Fig. 11. A parity game of index 3 for which $\{A_1A_2A_3\}$ returns $W_\sigma = \{\}$.

function for A_3 is

$$\mathsf{rank}(G) = |U|$$

A rank function for $\{A_1A_3\}$, $\{A_2A_3\}$, and $\{A_1A_2A_3\}$ is [11]

$$\mathsf{rank}(G) = |U| + \sum_{v \in U} \chi(v)$$

The interaction of aspects can improve the precision of algorithms derived from our design pattern. We illustrate this point by means of a simple example.

Example 5.1 For the parity game G in Fig. 10, the instantiation A_3 of our design pattern computes empty sets W_σ, whereas $\{A_2A_3\}$ completely solves this parity game: aspect A_2 re-sets $\chi(v_2)$ from 2 to X and then `checkInfFin({3,1},{2},1)` easily determines within aspect A_3 that player 1 wins all nodes.

We now show the incompleteness of $\{A_1A_2A_3\}$, even for parity games of index 3. This illustrates a potential weakness of our design pattern when all its aspects limit the role of players to a bounded scope of alternation.

[11] Although this rank function is sound, an implementation may want to note that turning some $\chi(v)$ from 0 to X decreases the rank, which the function as stated does not do.

Example 5.2 Parity game G in Fig. 11 has index 3. Aspect A_1 has no effect as any cycle through v_0 passes through v_1. Aspect A_3 won't increase any W_σ in the game `checkInfFin({2},{},0)` (since player 1 may move from v_2 to v_3), game `checkInfFin({1},{2},1)` (since player 0 may move from v_1 to v_0), game `checkInfFin({2,0},{1},0)` (since player 1 may move from v_2 to v_1), and game `checkInfFin({1},{2,0},1)` (since player 0 may move from v_1 to v_0). The only applications for aspect A_2 are at nodes v_2 and v_3. But changing the priority to X at any of these nodes prevents a change at the other, and the resulting game has no applications of A_2, and none for A_1 and A_3 for the same reasons as given before.

In Fig. 12 we suggest a transformation $A \mapsto \texttt{probe}(A)$ of any instantiation $A = A_{i_1}A_{i_2}\ldots A_{i_k}$ of our algorithm that can be used to increase its precision. The method `probe` has as input a method A that, if called on a parity game G, returns a pair $(\mathit{first}(A(G)), \mathit{second}(A(G)))$ such that $\mathit{first}(A(G)) \subseteq \mathsf{Win}_G(0)$ and $\mathit{second}(A(G)) \subseteq \mathsf{Win}_G(1)$. Moreover, it is not hard to see (but won't be needed in this paper) that the parity game and regions $W_0 = \mathit{first}(\texttt{probe}(A)(G))$ and $W_1 = \mathit{second}(\texttt{probe}(A)(G))$ computed by `probe(`A`)` satisfy our four invariants if those computed by A do.

Method `probe` uses method A to probe whether the winning status of additional nodes can be recognized. A call to `probe(`A`)` attempts to increase W_σ in different stages. For sake of brevity we only present two of the four stages we discovered:

- **Stage (1):** It assumes that an undecided node v is not won by the player who owns v, and uses method `A` to then derive a contradiction (as the winning regions of G won't change if all edges but a single one out of v are being removed). If a contradiction is found, v is then known to be won by the player who owns v. If no contradiction is found, nothing is known about the winning status of v. So stage (1) fixes a node v and computes winning regions for derived games in an attempt of making inferences about node v only.
- **Stage (2):** It makes no assumptions about the winning status of any undecided node but, for such an undecided node v, it computes the winning regions under-approximated by `A` for those games on which only one outgoing edge of v remains. If a node z is thus determined to be won by a particular player σ, regardless of which edge out of v remained, we know that, irrespective of who owns v or z, node z is indeed won by player σ in the original game. So stage (2) fixes a node v and computes sets of under-approximated winning regions for derived games and then decides the winning status of nodes in the intersection of these computed regions. In particular, it decides the winning status of node v only if v occurs in said intersection.

We now give a more detailed account of these stages and their soundness.

In stage (1), the hypothesis is that nodes are won by those players that own them. So let $v \in V_\sigma$ and let

$$S = \{w \in V \mid (v, w) \in E\}$$

be the set of nodes that can be reached from v via a single edge. By invariant (L) we may assume that S is non-empty. If S is a singleton, then v is a deterministic node and won't be tested by `probe` under this hypothesis. So let $|S| \geq 2$. The analysis attempts to use proof by contradiction: Assume that $v \in \mathsf{Win}_{\bar{\sigma}}(G)$.

- Then it won't matter which edge (v, w) player σ chooses as player $\bar{\sigma}$ can trap any play through v in $\mathsf{Win}_{\bar{\sigma}}(G)$. In particular, if for $w \in S$ we write $G_{(v,w)}$ for the parity game that is G except that (v, w) is now the only outgoing edge from v, then G and $G_{(v,w)}$ have the same winning regions. Now let $w_1 \neq w_2$ in S. If there is some $z \in \mathit{first}(A(G_{(v,w_1)})) \cap \mathit{second}(A(G_{(v,w_2)}))$ we know that z is in $\mathit{first}(A(G_{(v,w_1)})) \subseteq \mathsf{Win}_0(G_{(v,w_1)}) = \mathsf{Win}_0(G)$ as method A under-approximates winning regions of G and since v is assumed to be in $\mathsf{Win}_{\bar{\sigma}}(G)$. Similarly, we know $z \in \mathit{second}(A(G_{(v,w_2)})) \subseteq \mathsf{Win}_1(G_{(v,w_2)}) = \mathsf{Win}_1(G)$. But then $z \in \mathsf{Win}_0(G) \cap \mathsf{Win}_1(G)$ contradicts the determinacy of parity game G. Therefore $v \notin \mathsf{Win}_{\bar{\sigma}}(G)$, i.e. $v \in \mathsf{Win}_{\sigma}(G)$, follows.
- A similar argument for proving $v \in \mathsf{Win}_{\sigma}(G)$ can be made if the intersection $\mathit{second}(A(G_{(v,w_1)})) \cap \mathit{first}(A(G_{(v,w_2)}))$ is non-empty.

This reasoning principle is implemented in the for-statement `forall w1 != w2 in S` of the pseudo-code in Fig. 12. Note that these arguments only work if at least one of these intersections is non-empty. For example, if v has exactly two outgoing edges (v, w) and (v, w'), no new wins will be determined iff the sets $\mathit{first}(A(G_{(v,w)}))$, $\mathit{first}(A(G_{(v,w')}))$, $\mathit{second}(A(G_{(v,w)}))$, and $\mathit{second}(A(G_{(v,w')}))$ are pairwise disjoint.

In stage (2), method `probe` employs a second analysis, in which nothing is being assumed about the winning status of any undecided node. The for-statement `forall w in S`, for a fixed node v, collects in `Z_0` those nodes z that are known to be won by player 0 in all games $G_{(v,w)}$ with $(v, w) \in E$ an outgoing edge of v. Let $v \in V_{\sigma}$. We claim that $z \in \mathsf{Win}_0(G)$ and do a case analysis on who wins v:

- Let $v \in \mathsf{Win}_{\sigma}(G)$. Then player σ has some edge $(v, w) \in E$ as part of her memoryless winning strategy and so the winning regions of G and $G_{(v,w)}$ are the same. In particular, $z \in \mathsf{Win}_0(G_{(v,w)}) = \mathsf{Win}_0(G)$ since A under-approximates winning regions.
- Let $v \in \mathsf{Win}_{\bar{\sigma}}(G)$. Then player σ, who owns v, loses node v and so the winning regions of G and all $G_{(v,w)}$ are equal, from which $z \in \mathsf{Win}_0(G)$ follows as in the previous item.

Note that we need to form intersections in stage (2) since we don't know which player wins v, or which edge from v is part of a memoryless winning strategy.

For the computation of `Z_1` a similar argument for showing $z \in \mathsf{Win}_1(G)$ applies if z is in $\mathit{second}(A(G_{(v,w)}))$ for all $w \in S$.

The remaining parts of the pseudo-code just provide the infrastructure for collecting nodes whose winning status has been discovered by one of these two stages, noting that `probe(A)` ensures invariant (A).

Example 5.3 • We illustrate the working of the first stage of `probe(A)` where `A` is simply A_3 and G the parity game in Fig. 13. Assuming that v_1 is won by player

```
probe(A) {

  W_0 = W_1 = {};
  forall v in V {
    S = { w | (v,w) in E };

    % stage (1)
    forall w1 != w2 in S {
      if ((first(A(G_(v,w1))) & second(A(G_(v,w2)))) != {} ) ||
          (second(A(G_(v,w1))) & first(A(G_(v,w2)))) != {} ) {
        if (v in V_0) { W_0 = Attr_0(G, W_0 + {v} ) ; }
        else { W_1 = Attr_1(G, W_1 + {v} ) ; }
      }
    }

    % stage (2)
    if (S = {}) { Z_0 = Z_1 = {}; }
    else { Z_0 = Z_1 = V; }
    forall w in S {
      Z_0 = Z_0 & first(A(G_(v,w)));
      Z_1 = Z_1 & second(A(G_(v,w)));
    }

    % postprocessing: secure invariant (A)
    W_0 = Attr_0( G, W_0 + Z_0 );
    W_1 = Attr_1( G, W_1 + Z_1 );
  }
  return (W_0,W_1);
```

Fig. 12. Pseudo-code for method `probe`. Its input is a method `A` that computes under-approximations of winning regions. In `probe(A)` one uses method `A` on versions of G that commit to specific edges from nodes and then analyses the collection of results to detect winning nodes for G. The combinatorics of these analyses rely on the fact that winning strategies are memoryless and that nodes that are not won by the player who owns them satisfy a non-interference property: committing to an edge from that node won't change winning regions. Expressions `first(A(G))` and `second(A(G))` denote the under-approximations computed by method `A` for $\mathsf{Win}_0(G)$ and $\mathsf{Win}_1(G)$, respectively. We write `G_(v,w)` for the parity game that is `G` except that `(v,w)` is the only outgoing edge for `v`. We use `&` to denote the intersection of sets.

1, who does not own v_1, it must be the case that games $G_{(v_1,v_3)}$ and $G_{(v_1,v_2)}$ have the same winning regions. But aspect A_3 then recognizes that node v_0 is won by player 1 in game $G_{(v_1,v_3)}$, and by player 0 in game $G_{(v_1,v_2)}$. Therefore, `probe(`A_3`)` recognizes in its first stage that v_1 is won by player 1 in G.

- We illustrate the utility of the second stage in `probe` where G is the parity game from Fig. 11 and `A` is $A_1A_2A_3$. Recall that this `A` returns a pair of empty sets for this G. Let `v` be v_2. Then `S` equals $\{v_1, v_3\}$. Consider the for-statement `forall w in S` for this choice of `v`. Then `Z_0` = `Z_1` initially.
  · In $G_{(v_2,v_1)}$ edge (v_2, v_3) is gone and $\mathit{first}(A(G_{(v_2,v_1)})) = V$ since A_3 realizes $V = \mathsf{Win}_0(G_{(v_2,v_1)})$ by running `checkInfFin({2},{}, 0)` on $G_{(v_2,v_1)}$.

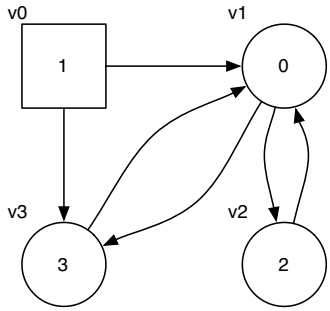

Fig. 13. A parity game G for which aspect `probe(`A_3`)` can decide, in its first stage, that node v_1 is won by the player who does not own v_1 — which is player 1.

· In $G_{(v_2,v_3)}$ edge (v_2, v_1) is gone and $\mathit{first}(A(G_{(v_2,v_3)})) = V$ since A_3 realizes $V = \mathsf{Win}_0(G_{(v_2,v_3)})$ by running `checkInfFin({2,0},{1}, 0)` on $G_{(v_2,v_3)}$.

This results in final states V and $\{\}$ for `Z_0` and `Z_1`, respectively. Therefore, `probe(A)` returns that pair $(V, \{\})$ and solves G completely.

Remark 5.4 The first stage of `probe(`A_3`)` won't discover anything that aspect A_3 would not already have discovered. Suppose we have nodes v and z and that there are $w_1 \neq w_2$ with $\{(v, w_1), (v, w_2)\} \in E$ such that

(i) aspect A_3 tells us that z is won by player 0 in the reduced game $G_{(v,w_1)}$

(ii) aspect A_3 tells us that z is won by player 1 in the reduced game $G_{(v,w_2)}$

Let $v \in V_\sigma$. We let $\sigma = 0$, the proof for $\sigma = 1$ is symmetric and omitted. Then item (i) ensures that, under some interval abstraction of priorities, player σ can force a win from node z in the reduced game $G_{(v,w_0)}$. Let f be the corresponding memoryless winning strategy. But only player σ can move in configuration v, and so in the original game G, under the same interval abstraction from item (i), aspect A_3 will show that player σ can force a win at z (by choosing to go to w_1 from v, and otherwise playing f as in item (i)). So aspect A_3 would already have added z to the winning region W_σ. This can be seen at work in the first item of Example 5.3.

Nothing stops us from thinking of `probe(A)` as an input to `probe` and so we get higher-order analyses, e.g. the second-order one `probe(probe(A))`.

Since some aspects change W_σ and some don't, we put the computation of σ-attractors for invariant (A) into the pseudo-code of those aspects that change W_σ and not at the end of each iteration in the for-statement of our design pattern. In this paper, this only concerns aspect A_3 and `probe(A)`.

The design pattern and all of its aspects have been implemented by Huaxin Wang in a framework [16]. That implementation lead to several empirical and theoretical insights, detailed in loc. cit. and not reported here.

6 Related work

This paper is based on preliminary work reported in the extended abstract [1]. In loc. cit., neither the aspect transformer nor proofs are given but the design pattern (without handling dead ends and self-loops) and weaker invariants are presented. That paper features four aspects, of which A_1 is as presented in this paper and A_3 is formulated in a non-incremental manner. Using the notion of lax parity games, we could and did merge aspects A_2 and A_4 of [1] into the single aspect A_2 of this paper.

We refer to [8] for references to extant algorithms for solving parity games whose worst-case time complexity is exponential.

Polynomial-time algorithms for solving parity games completely can be found in the literature but they only operate on certain parity games, typically by using a measure $\mu(G) \in \{0, 1, \dots\}$ on parity games and applying the algorithm only to those games whose measure is below a fixed, finite bound k. This is the case for the polynomial-time algorithm for solving parity games with bounded entanglement [2], and for the polynomial-time algorithm for solving parity games with bounded DAG-width [3]. It would be of interest to determine whether these algorithms have sound abstractions that can operate as aspects on arbitrary parity games.

In [11], the characterization of winning strategies through progress measures (as given in [13]) is exploited to reduce the problem of solving parity games to satisfiability checks in propositional logic, and first experimental results for this reduction are presented. This approach only decides the winning status of a chosen node and still has to be extended to global reasoning over parity games. Again, it would be of interest to study whether progress measures and their resulting reduction to satisfiability can be formulated as aspects within our framework.

Aspect A_3 and the under-approximation of winning regions in our design pattern use abstraction as a driving computational force. Abstraction is an established approach in game-based verification. We mention three-valued abstractions of 2-player games [5], where winning strategies for either player transfer soundly from the abstract to the concrete game. Similar results on transfer of winning strategies, within the formal framework of abstract interpretation [4], are presented in [14], where abstract game nodes are also equipped with a partial order.

7 Future work

We want to explore additional aspects and rank functions. We plan to give a more systematic account of interaction of these aspects in the context of the aforementioned "phase ordering problem" in compilers, as applied to this setting; preliminary results in that direction can be found in [16] already. Connections of our approach to complexity measures in directed graphs are of interest; we mention DAG-Width [3] and Entanglement [2]; for the latter, initial results are reported in [16]. We also mean to compare our approach to that of reducing parity games to SAT [11].

8 Conclusions

In this paper we proposed a design pattern for under-approximating winning regions of parity games in polynomial time and in a completely incremental manner. Patterns are being instantiated with k many aspects and a rank function suitable for that choice of aspects. We presented three aspects and a method for making our algorithms higher-order, all based on static analyses and the determinacy of parity games. We demonstrated that aspect interaction increases precision, due in part to an elimination of redundancies in parity games that lead to the consideration of lax parity games, which are as expressive as parity games. The utility of our approach has at least three sources. First, in local model checking [15] one is only interested in whether an initial state (read: designated node) satisfies a formula (read: is won by player 0), something our algorithms may well be able to decide. Second, our algorithms can be seen as pre-processors to existing algorithms since they can be applied directly to the residual sub-game induced by those nodes whose winning status is yet undecided, courtesy of invariants (A) and (P) for our design pattern. Third, the results of this paper and the implementation of that framework in [16] provide a workbench for designing and exploring algorithms for solving parity games, noting that the framework does not insist on aspects to run in polynomial time.

References

[1] A. Antonik, N. Charlton, and M. Huth. Computing safe winning regions of parity games in polynomial time. In: *Proc. of INFORMATION-MFCSIT'06*, pp. 340–344, University College Cork, 2006.

[2] D. Berwanger and E. Grädel. Entanglement - A measure for the complexity of directed graphs with applications to logic and games. In: *Proc. of LPAR'04*, LNCS 3452, pp. 209–223, Springer-Verlag, 2005.

[3] D. Berwanger, A. Dawar, P. Hunter, and S. Kreutzer. DAG-width and parity games. In: *Proc. of STACS'06*, LNCS 3884, pp. 524–436, Springer-Verlag, 2006.

[4] P. Cousot and R. Cousot. Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. In: *Proc. of POPL 1977*, pp. 238–252, 1977.

[5] L. de Alfaro, P. Godefroid, and R. Jagadeesan. Three-Valued Abstractions of Games: Uncertainty, but with Precision. In: *Proc. of LICS'04*, pp.170–179, IEEE Computer Society Press, 2004.

[6] A. Emerson and C. Jutla. Tree automata, mu-calculus and determinacy. In: *Proc. of the 32nd IEEE Symp. on Foundations of Computer Science*, pp. 368–377, 1991.

[7] E. Grädel, W. Thomas, and T. Wilke (Eds.). Automata, Logics, and Infinite Games — A Guide to Current Research, LNCS 2500, 385 pages. Springer Verlag, October 2002.

[8] H. Klauck. Algorithms for Parity Games. In [7].

[9] D. Kozen. Results on the propositional μ-calculus. *TCS* 27:333–354, 1983.

[10] R. Küsters. Memoryless Determinacy of Parity Games. In [7].

[11] M. Lange. Solving Parity Games by a Reduction to SAT. In *Proc. of the Workshop on Games in Design and Verification*, GDV'05, Edinburgh, Scotland, UK, 2005.

[12] M. Jurdziński. Deciding the winner in parity games is in UP $\cap$ co-UP. *Information Processing Letters*, 68(3):119–124, 1998.

[13] M. Jurdziński. Small progress measures for solving parity games. In *Proc. of the 17th Annual Symposium on Theoretical Aspects of Computer Science*, LNCS 1770, pp. 290–301, 2000.

[14] P. Stevens. Abstract Interpretation of games. In: *Proc. of VMCAI 1998*, Technical Report no. CS98-12, University of Venice, Italy.

[15] C. Stirling and D. Walker. Local Model Checking in the Modal Mu-Calculus. *TCS* 89(1): 161-177, 1991.

[16] H. Wang. Framework for Under-Approximating Solutions of Parity Games in Polynomial Time. *MEng Thesis*, Department of Computing, Imperial College London, 78 pages, June 2007.

[17] W. Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *TCS* 200:135-183, 1998.

A Eliminating dead ends and self-loops

Eight local transformations, four for each player, efficiently add nodes and edges to a parity game such that the winning status of old nodes won't change but no dead ends or self-loops remain. The eight transformations are given in Fig. A.1. All transformations for node v preserve all incoming edges and outgoing edges other than (v, v) of v (those edges that are not self-loops are not shown below; in particular, the fifth and sixth rule won't produce dead ends). All transformations for node v, except for the fifth and sixth, create two new nodes and three new edges each. Since these new nodes exhibit only two different patterns, sharing of these patterns means that only four new nodes in total are needed.

Example A.1 We illustrate this elimination process by means of an example. The parity game G in Fig. A.2 has $\mathsf{Win}_0(G) = \{v_2, v_3, v_4, v_6\}$ and $\mathsf{Win}_1(G) = \{v_0, v_1, v_5\}$. It turns into the parity game without dead ends and self-loops in Fig. A.3 such that the winning regions are the same for the original nodes. The two new nodes with priority 1, and all their incoming nodes are won by player 1. Dually, the two new nodes with priority 0 and all its incoming nodes are won by player 0.

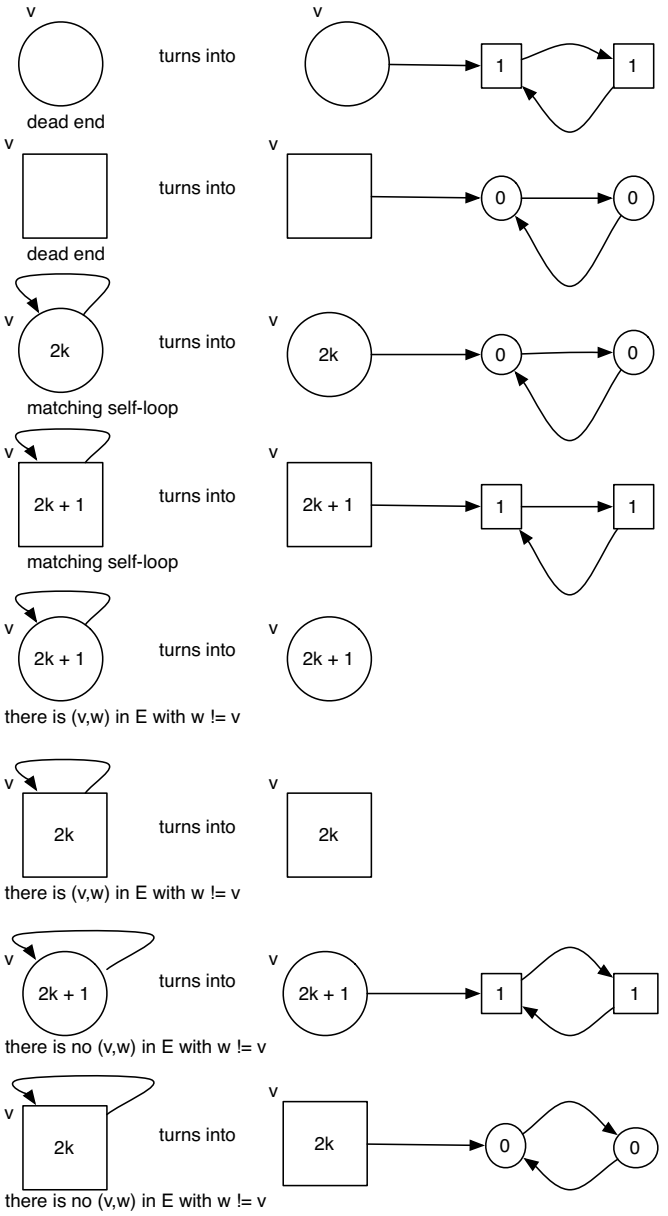

Fig. A.1. Eight transformation rules for converting parity games into equivalent ones without any dead ends or self-loops. Those edges that are not self-loops are not shown; in particular, the fifth and sixth rule won't produce dead ends.

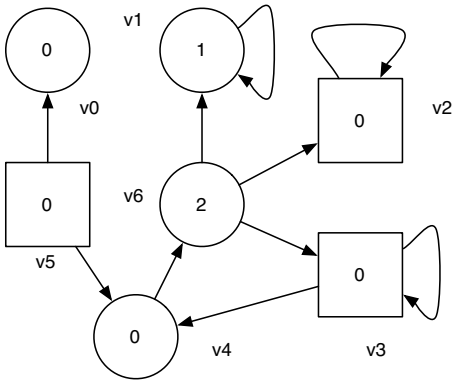

Fig. A.2. A parity game with dead end and self-loops.

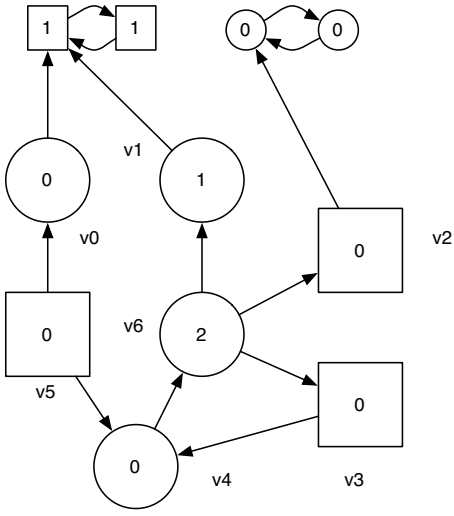

Fig. A.3. The parity game obtained from the one in Fig. A.2 by applying the rewrite rules in Fig. A.1.