Simulation Relations for Alternating Parity Automata and Parity Games*

Carsten Fritz and Thomas Wilke

Christian-Albrechts-Universität zu Kiel
{fritz, wilke}@ti.informatik.uni-kiel.de

Abstract. We adapt the notion of delayed simulation to alternating parity automata and parity games. On the positive side, we show that (i) the corresponding simulation relation can be computed in polynomial time and (ii) delayed simulation implies language inclusion. On the negative side, we point out that quotienting with respect to delayed simulation does not preserve the language recognized, which means that delayed simulation cannot be used for state-space reduction via merging of simulation equivalent states. As a remedy, we introduce finer, so-called biased notions of delayed simulation where we show quotienting does preserve the language recognized. We propose a heuristic for reducing the size of alternating parity automata and parity games and, as an evidence for its usefulness, demonstrate that it is successful when applied to the Jurdziński family of parity games.

1 Introduction

The motivation for studying simulation relations for automata is, in general, two-fold: First, simulation relations are an appropriate means for comparing the structure of automata. They formalize the idea that one automaton is capable of mimicking the behavior of another automaton. In other words, they are useful for identifying structural similarities of automata. This is also true in the context of transition systems and processes, in fact, simulation relations were introduced in a wider context [1]. Second, simulation relations have proved to be very useful for efficiently reducing the size of (finite-state) automata, the basic idea being to merge states which simulate each other. This method is also known as quotienting, and it is a well-established method for reducing the number of states of a given Büchi automaton, especially in the context of generating small Büchi automata from formulas in linear temporal logic, see [2, 3, 4, 5, 6, 7, 8, 9].

The objective of the work presented in this paper is to extend the notion of simulation to alternating parity automata (also known as alternating Rabin chain automata), see [10, 11], and to identify how simulation can be used for state-space reduction. From the simpler scenario with Büchi automata it is known, see [12, 13], that it is reasonable to distinguish different types of simulations: direct, delayed, and fair simulation. Direct simulation is less interesting from the

* Project funded by the Deutsche Forschungsgemeinschaft under no. 223228.

point of view of state-space reduction, because it yields the finest of the three relations and thus pays off the least. Fair simulation, on the other hand, is too coarse, for quotienting with respect to it may change the recognized language [13, 14]. That is why we focus on delayed simulation.

There are two major technical problems to overcome for delayed simulation. The first problem is that a priori it is not at all clear how the different priorities of the states of a parity automaton should be taken into account in a definition of simulation where delays are allowed. We try to give a definition as general as possible in the sense that the resulting simulation relation is as coarse as possible. Our approach is game-theoretic, just as in [13], and allows us to prove that our notion has the basic properties of a simulation relation. When it comes to quotienting, our definition, however, turns out to be too general: quotienting does not preserve the language recognized, which is then the second technical problem to overcome. We describe two ways of strengthening our notion of delayed simulation, so-called biased simulations, for which we can then show that quotienting still works.

To achieve our goal of developing an efficient heuristic for reducing the state spaces of alternating parity automata, we combine quotienting with respect to the biased relations with basic simplification methods and simplification methods involving our general delayed simulation relation. The heuristic we propose turns out to be successful when applied to parity games (a parity game is simply an alternating parity automaton over a unary alphabet): It reduces parity games which have been shown to be difficult instances for a certain game solving algorithm quite fast to games with just two positions, see [15].

The paper is structured as follows. In Sect. 2, we briefly describe our notation and terminology. In Sect. 3, we then introduce our general definition of delayed simulation for alternating parity automata, describe its main properties, and explain why quotienting does not work. In Sect. 4, we explain how the biased variants of the general delayed simulation are obtained and how they can be used for quotienting. Before we conclude, we describe our heuristic for state-space reduction in Sect. 5.

The proofs of most of the results are very involved; the reader is referred to [16] for details. For background on games and accepting/winning conditions, see [17].

2 Basic Notation and Terminology

An *infinite game* is a tuple

$$\mathcal{G} = (P, P_0, P_1, p_I, Z, W) \tag{1}$$

where P is a set of *positions*, $P_0 \subseteq P$ are the *positions of Player 0*, $P_1 \subseteq P$ are the *positions of Player 1* such that $P = P_0 \cup P_1$ and $P_0 \cap P_1 = \emptyset$, $p_I \in P$ is an *initial position*, $Z \subseteq P \times P$ is the set of *moves*, and $W \subseteq P^\omega$ is the *winning condition (for Player 0)*. For convenience, we require that the set of moves is *complete,* that is, for every $p \in P$ there is some $p' \in P$ such that $(p, p') \in Z$ is a move.

A *play* of such a game is an infinite path $\pi = p_0 p_1 p_2 \dots$ through the *game graph* (P, Z) starting in p_I. It is *winning* for Player 0 if $\pi \in W$.

A *parity game* is an infinite game as in (1) where W is specified indirectly by a *priority function* $\Omega \colon P \to \omega$ which is required to have a finite image. The winning condition associated with Ω, denoted $W(\Omega)$, is the set which contains a sequence $p_0 p_1 p_2 \dots$ if $\min\{m \mid \exists^\infty i(\Omega(p_i) = m)\}$ (which is well-defined because Ω is required to have a finite image) is even. That is, Player 0 wins if the minimum priority occurring infinitely often is even.

An *alternating parity automaton (APA)* is a tuple

$$\mathcal{Q} = (Q, \Sigma, q_I, \Delta, E, U, \Omega) \tag{2}$$

where Q is a finite set of *states,* Σ is an *alphabet*, $q_I \in Q$ is the *initial state*, $\Delta \subseteq Q \times \Sigma \times Q$ is the *transition relation*, $E \subseteq Q$ is the set of *existential states*, $U \subseteq Q$ is the set of *universal states* such that $E \cup U = Q$ and $E \cap U = \emptyset$, and $\Omega \colon Q \to \omega$ is the *priority function.* Without loss of generality, we require that Δ is complete, that is, for every $q \in Q$, $a \in \Sigma$ there must exist a state $q' \in Q$ such that $(q, a, q') \in \Delta$.

Acceptance of an APA is best explained using games. Given an APA $\mathcal{Q}$ as in (2) and an ω-word $w_0 w_1 w_2 \dots \in \Sigma^\omega$, the *word game*

$$G(\mathcal{Q}, w) = (P, P_0, P_1, p_I, Z, \Omega') \tag{3}$$

is the parity game where $P = Q \times \omega$, $P_0 = E \times \omega$, $P_1 = U \times \omega$, $p_I = (q_I, 0)$, $Z = \{((q, i), w_i, (q', i + 1) \mid i \in \omega \wedge (q, w_i, q') \in \Delta\}$ and $\Omega'((q, i)) = \Omega(q)$.

In this game Player 0 and Player 1 are called *Automaton* and *Pathfinder*, respectively, following the terminology from [18]. The word w is *accepted* by $\mathcal{Q}$ if Automaton has a winning strategy in $G(\mathcal{Q}, w)$. The set of all words accepted by $\mathcal{Q}$ is denoted $L(\mathcal{Q})$.

We will use the following ordering on natural numbers (also used in, e. g., [19]), which reflects the parity winning condition. The *reward order* $\preceq$ is the total order on ω defined by $m \preceq n$ if and only if m is even and n is odd, or m and n are even and $m \leq n$, or m and n are odd and $n \leq m$. That is, $0 \prec 2 \prec 4 \prec \dots \prec 5 \prec 3 \prec 1$. When $n \prec m$, we will say n *is better than* m, while terms like *minimum* and *smaller than* will always be used w. r. t. the standard order $\leq$.

3 Delayed Simulation for the Parity Condition

On a very abstract level, delayed simulation can be explained as follows. A state s simulates a state q *directly*, if everything that can be done starting from q can be mimicked step-by-step starting from s. Here, mimicking means that the simulating step must be as good as (with respect to acceptance) the simulated step. For *delayed* simulation, direct simulation is relaxed: A step need not be simulated directly, but a finite delay is allowed.

Delays lead to “pending obligations”, every one of which has to be fulfilled eventually. Our definition of delayed simulation will therefore have mechanisms for keeping track of pending obligations and for checking that pending obligations have been fulfilled.

3.1 Formal Definition of Delayed Simulation

Suppose $\mathcal{Q}$ and $\mathcal{S}$ are APA's of the form $\mathcal{Q} = (Q, \Sigma, q_I, \Delta, E^{\mathcal{Q}}, U^{\mathcal{Q}}, \Omega^{\mathcal{Q}})$ and $\mathcal{S} = (S, \Sigma, s_I, \Delta^{\mathcal{S}}, E^{\mathcal{S}}, U^{\mathcal{S}}, \Omega^{\mathcal{S}})$. We want to define what it means for $\mathcal{Q}$ to be simulated by $\mathcal{S}$ in a delayed fashion. We do this by first describing an infinite game, named *simulation game* and denoted $\mathcal{G}^{de}(\mathcal{Q}, \mathcal{S})$, where Player 0 and Player 1 are called *Duplicator* and *Spoiler*, respectively. We then say that $\mathcal{S}$ simulates $\mathcal{Q}$ if Duplicator wins the game.

The idea of the game, similar to other game-based definitions of simulation, see, e.g., [13], is as follows. Throughout the game, there is one pebble on a state of $\mathcal{Q}$ and another pebble on a state of $\mathcal{S}$. The game proceeds in rounds in such a way that in every round first Spoiler chooses a letter and then Spoiler and Duplicator move the pebbles along transitions labeled with the chosen letter according to rules which depend on the modes of the states the pebbles are on. The outcome of a play are two infinite sequences of states in $\mathcal{Q}$ and $\mathcal{S}$, respectively. Duplicator wins the play if each obligation built up during the course of the game is eventually fulfilled. For an easier formal description, information about the pending obligations will be built into the game graph.

Formally, $\mathcal{G}^{de}(\mathcal{Q}, \mathcal{S}) = (P, P_{\mathsf{Du}}, P_{\mathsf{Sp}}, p_I, Z, W)$ with components defined as follows. The **positions** contain information on where the pebbles are on, whether Spoiler has already chosen a letter and if so, which letter, what the pending obligations are, and who is going to move next and where (if that cannot be deduced from the other components). We use the elements of $K = \Omega^q(Q) \cup \Omega^s(S) \cup \{\surd\}$ to describe the pending obligations in terms of a priority to be met by Duplicator, where the check mark stands for "all obligations fulfilled".

The set of all positions is the union of P^0, P^1, and P^2 defined by

$$P^0 = Q \times S \times K \ , \qquad P^1 = Q \times S \times K \times \Sigma \ ,$$
$$P^2 = Q \times S \times K \times \Sigma \times \{\mathsf{Sp}, \mathsf{Du}\} \times \{\mathcal{Q}, \mathcal{S}\} \ ,$$

where positions in P^i describe configurations of the game at the beginning of a round, after Spoiler has chosen a letter, and after one of the players has moved and one player is still to move, respectively; a position in P^2 explicitly specifies who is to move and which pebble.

The **initial position** is determined by: If $\Omega^{\mathcal{Q}}(q_I) \prec \Omega^{\mathcal{S}}(s_I)$, then the initial position is given by $p_I = (q_I, s_I, \min\{\Omega^{\mathcal{Q}}(q_I), \Omega^{\mathcal{S}}(s_I)\})$, else it is $p_I = (q_I, s_I, \surd)$.

Depending on the modes of the states the pebbles are on at the beginning of a round, the players **move** according to the following table:

q	s	1st player plays on		2nd player plays on	
$E^{\mathcal{Q}}$	$E^{\mathcal{S}}$	Sp	$\mathcal{Q}$	Du	$\mathcal{S}$
$E^{\mathcal{Q}}$	$U^{\mathcal{S}}$	Sp	$\mathcal{Q}$	Sp	$\mathcal{S}$
$U^{\mathcal{Q}}$	$E^{\mathcal{S}}$	Du	$\mathcal{S}$	Du	$\mathcal{Q}$
$U^{\mathcal{Q}}$	$U^{\mathcal{S}}$	Sp	$\mathcal{S}$	Du	$\mathcal{Q}$

For instance, if both, q and s, are existential, Spoiler moves first on $\mathcal{Q}$ and then Duplicator moves on $\mathcal{S}$.

The table explains why P_{Du} is defined by

$$P_{\mathsf{Du}} = (U^{\mathcal{Q}} \times E^{\mathcal{S}} \times K \times \Sigma) \cup (Q \times S \times K \times \Sigma \times \{\mathsf{Du}\} \times \{\mathcal{Q}, \mathcal{S}\})$$

and that P_{Sp} is defined to be the set of the remaining positions.

The important part of a move is how the pending obligations are updated. To describe this, we define a function $\gamma \colon \omega \times \omega \times (\omega \cup \{\surd\}) \to \omega \cup \{\surd\}$ as follows. First, we set $\gamma(i, j, \surd) = \surd$ if $j \preceq i$, and $\gamma(i, j, \surd) = \min\{i, j\}$ otherwise. When the third argument is a natural number k, we set

$$\gamma(i, j, k) = \begin{cases} \surd \ , \ \text{if} \begin{cases} j \preceq i, \ i \text{ odd}, \ i \leq k, \text{ or} \\ j \preceq i, \ j \text{ even}, \ j \leq k, \end{cases} \\ \min\{i, j, k\} \ , \ \text{otherwise.} \end{cases} \tag{4}$$

For instance, the first clause says that if Duplicator is supposed to meet k, q has priority i, s has priority j, then it is enough when $j \preceq i$ and $i \leq k$ for an odd i.

That is, we store an obligation $\min\{i, j\}$ if the priority i of $\mathcal{Q}$ is better than the priority j of $\mathcal{S}$. One possibility to meet this obligation in a future round is that the priority i of $\mathcal{Q}$ in that round is odd and less than or equal to the obligation (especially, i is at most as good as the obligation) while at the same time, j is at least as good as i. In that case, the stored obligation no longer is a witness for a, so to say superior acceptance behavior of $\mathcal{Q}$ as compared to $\mathcal{S}$, which means it can be erased. Symmetrically, the obligation can be met by a small even value of j in $\mathcal{S}$.

The set Z of all moves is the union of the sets Z^0, Z^1, and Z^2 defined below, where $2\mathrm{pl}(q, s)$ and $2\mathrm{au}(q, s)$ are determined by the above table (last two columns):

$$\begin{aligned} Z^0 &= \{((q, s, k), (q, s, k, a)) \mid (q, s, k) \in P^0 \wedge a \in \Sigma\} \ , \\ Z^1 &= \{((q, s, k, a), (q', s, k, a, 2\mathrm{pl}(q, s), 2\mathrm{au}(q, s)) \mid q \in E^{\mathcal{Q}} \wedge (q, a, q') \in \Delta^{\mathcal{Q}}\} \\ &\quad \cup \{((q, s, k, a), (q, s', k, a, 2\mathrm{pl}(q, s), 2\mathrm{au}(q, s)) \mid q \in U^{\mathcal{Q}} \wedge (s, a, s') \in \Delta^{\mathcal{S}}\}, \\ Z^2 &= \{((q, s, k, a, x, \mathcal{Q}), (q', s, \gamma(\Omega(q'), \Omega(s), k))) \mid x \in \{\mathsf{Sp}, \mathsf{Du}\} \wedge (q, a, q') \in \Delta^{\mathcal{Q}}\} \\ &\quad \cup \{((q, s, k, a, x, \mathcal{S}), (q, s', \gamma(\Omega(q), \Omega(s'), k))) \mid x \in \{\mathsf{Sp}, \mathsf{Du}\} \wedge (s, a, s') \in \Delta^{\mathcal{S}}\}. \end{aligned}$$

A play $\pi = p_0 p_1 p_2 \dots$ of the delayed simulation game is a win for Duplicator iff there are infinitely many i such that the check mark occurs in the third component of p_i (every obligation is eventually satisfied), that is, the winning condition can be expressed as a Büchi condition and, of course, as a parity condition.

If Duplicator has a winning strategy in $\mathcal{G}^{de}(\mathcal{Q}, \mathcal{S})$, we write $\mathcal{Q} \leq_{de} \mathcal{S}$ and say "$\mathcal{S}$ de-simulates $\mathcal{Q}$". By abuse of notation, when q and q' are states of the same automaton $\mathcal{Q}$, we write $q \leq_{de} q'$ and say q' de-simulates q if $\mathcal{Q}[q] \leq_{de} \mathcal{Q}[q']$ where $\mathcal{Q}[q]$ and $\mathcal{Q}[q']$ are obtained from $\mathcal{Q}$ by making q and q', respectively, the initial state.

3.2 Basic Properties of $\leq_{de}$

Delayed simulation for APA's has a number of useful and important properties, which we now describe. We start with some definitions.

We say that a relation $\leq$ between automata *implies language containment* if $\mathcal{Q} \leq \mathcal{S}$ implies $L(\mathcal{Q}) \subseteq L(\mathcal{S})$. This is one of the basic properties one expects of a simulation relation.

The *dual* of an APA $\mathcal{Q}$, denoted $\tilde{\mathcal{Q}}$, is obtained from $\mathcal{Q}$ by exchanging the roles of existential and universal states and replacing Ω by $\Omega + 1$. (Observe that $\tilde{\tilde{\mathcal{Q}}}$ is the same as $\mathcal{Q}$ modulo reducing all priorities by 2. Also, $L(\tilde{\mathcal{Q}}) = \Sigma^\omega \setminus L(\mathcal{Q})$.)

Theorem 1 (properties of delayed simulation)
1. *On APA's, the relation* $\leq_{de}$
   (a) is a preorder (reflexive and transitive) and
   (b) implies language containment.
2. *The relation* $\leq_{de}$ *can be computed in time* $O(n^3 l^2 m)$ *and space* $O(mnl)$ *on an APA with* n *states,* m *transitions, and* l *priorities.*
3. *For APA's* $\mathcal{Q}$ *and* $\mathcal{S}$*, we have* $\mathcal{Q} \leq_{de} \mathcal{S}$ *iff* $\tilde{\mathcal{S}} \leq_{de} \tilde{\mathcal{Q}}$.

The proof of the transitivity of $\leq_{de}$ is quite technical; it involves a notion of strategy composition, in analogy to what is explained in [20]. Since the delayed simulation game has a Büchi winning condition, it can be solved using the approach of [12], which gives the desired bound on the running time and space.

3.3 Quotienting is a Problem for $\leq_{de}$

Recall that a major motivation for studying simulation relations is state-space reduction, the basic idea being that states that simulate each other are merged and thus incurring a reduction in the number of states. This process is usually referred to as *quotienting.*

More precisely, let $\leq$ be a preorder on the state space of an automaton $\mathcal{Q}$ and $\equiv$ the corresponding equivalence relation defined by $q \equiv q'$ iff $q \leq q'$ and $q' \leq q$. Then the states of a quotient of $\mathcal{Q}$ with respect to $\leq$ are the equivalence classes of $\equiv$. But this does not fully determine a quotient. In addition, one has to specify: how the equivalence classes are connected by transitions, for any automaton; how the the set of equivalence classes is partitioned into universal and existential classes, for any alternating automaton; how priorities are assigned to equivalence classes, for any parity automaton. The overall objective is to define a quotient in such a way that it is simulation equivalent to the given automaton—we call this a *simulation preserving quotient*—and recognizes the same language. (Formally, two APA's $\mathcal{Q}$ and $\mathcal{S}$ are simulation equivalent if $\mathcal{Q} \leq \mathcal{S}$ and $\mathcal{S} \leq \mathcal{Q}$.)

Several quotients have been discussed in the literature for various types of automata and simulation relations, for instance, *naive quotients*, where every transition in the given automaton induces a transition in the quotient (representative-wise), *minimax quotients*, and *semi-elective quotients*, see, for instance, [20].

Unfortunately, it turns out to be quite difficult to find a working definition of a simulation preserving quotient with respect to delayed simulation. In fact, it is

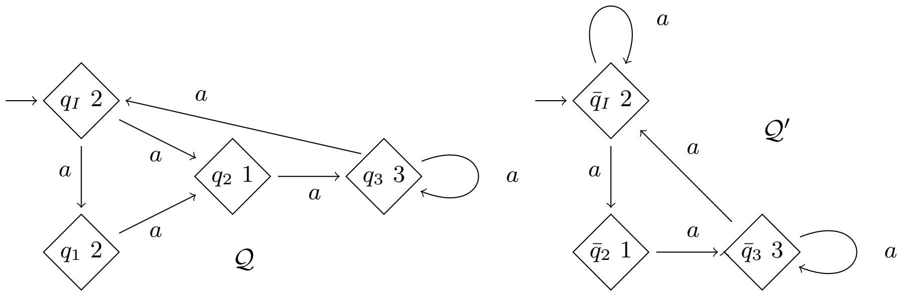

Fig. 1. An APA $\mathcal{Q}$ and a naive delayed simulation quotient $\mathcal{Q}'$

not at all clear how such a quotient should be defined, as we will argue in what follows. Note that in all the examples below not even the language is preserved.

For a start, consider the APA $\mathcal{Q}$ in Figure 1. As usual, existential states are shown as diamonds (there are no universal states). The labels of the states give the name and the priority of a state. It is easy to check that $q_I \equiv_{de} q_1 <_{de} q_2 <_{de} q_3$. The automaton $\mathcal{Q}'$ in Figure 1 is a naive quotient of $\mathcal{Q}$ where the equivalence class of a state q with respect to $\equiv$ is denoted $\bar{q}$.

Now observe that $L(\mathcal{Q})$ is empty, while $L(\mathcal{Q}') = \{a^\omega\}$. Further, note that $|\Sigma| = 1$ (the automaton is merely a parity game), all states have the same mode, only states in the same strongly connected component (SCC) are merged, and there are only three different priorities, which shows the situation is not complicated at all.

For $\mathcal{Q}$, one gets a simulation-equivalent quotient if the transition $(\bar{q}_I, a, \bar{q}_I)$ is removed from $\mathcal{S}$. This might suggest to use a minimax quotient as advocated in [20] for alternating Büchi automata and direct simulation. But note that, in general, minimax quotients do not even work for alternating Büchi automata and delayed simulation. In addition, the semi-elective quotient introduced in [20] for alternating Büchi automata and delayed simulation does not work for alternating parity automata, because in semi-elective quotients, transitions originating from existential states must not be removed, which means the semi-elective quotient of $\mathcal{Q}$ does not preserve the language.

The studies on alternating Büchi automata might suggest an approach in which existential states retain only their maximal transitions provided their priority is even. This would be correct for $\mathcal{Q}$, but we have other examples that exclude this approach as well, see [16].

4 Biased Delayed Simulations

We have just seen that our delayed simulation relation makes it difficult to merge equivalent states. As a remedy, we present two simulation relations, denoted $\leq^e_{de}$ and $\leq^o_{de}$, and corresponding quotienting constructions that are simulation preserving.

4.1 Definition and Basic Properties

The relations $\leq_{de}^{e}$ and $\leq_{de}^{o}$ are finer than $\leq_{de}$. They are defined just as $\leq_{de}$ with the only difference that the function γ which accumulates the pending obligations is replaced by variants, γ^e and γ^o, respectively. These functions coincide with γ except for the following cases:

– if $j \preceq i$, i odd, $i \leq k$, and (j odd or $k < j$), then $\gamma(i,j,k)^e = k$, and
– if $j \preceq i$, j even, $j \leq k$, and (i even or $k < i$), then $\gamma(i,j,k)^o = k$.

That is, in the case of $\leq_{de}^{e}$ (e reminiscent of "even") , once the value of the priority memory is not $\surd$, it will change back to the value $\surd$ only if this is triggered by a small even priority in the simulating automaton, while small odd priorities in the simulated automaton are ignored.

We call $\leq_{de}^{e}$ and$\leq_{de}^{o}$ the *even-biased delayed simulation relation* and *odd-biased delayed simulation relation*, respectively.

These two new relations have all basic properties of a simulation relation:

Theorem 2 (properties of biased delayed simulations)
1. *On APA's, the simulation relations $\leq_{de}^{e}$ and $\leq_{de}^{o}$*
   (a) *are preorders (reflexive and transitive),*
   (b) *imply language containment, and*
   (c) *are at least as fine as $\leq_{de}$.*
2. *The relations $\leq_{de}^{e}$ and $\leq_{de}^{o}$ can be computed in time $O(n^3 l^2 m)$ and space $O(mnl)$ on an APA with n states, m transitions, and l priorities.*
3. *For APA's $\mathcal{Q}$ and $\mathcal{S}$, we have $\mathcal{Q} \leq_{de}^{e} \mathcal{S}$ iff $\tilde{\mathcal{S}} \leq_{de}^{o} \tilde{\mathcal{Q}}$.*

The proofs are similar to the proofs of the assertions in Theorem 1, but in some places even more complicated because of the asymmetry in the two definitions.

We note that, in general, $\leq_{de}^{e}$ and $\leq_{de}^{o}$ are strictly finer than $\leq_{de}$, and even the reflexive-transitive closure of $\leq_{de}^{l} \cup \leq_{de}^{r}$ is strictly finer than $\leq_{de}$.

4.2 Quotienting

We next present the quotients which can be used with the biased delayed simulation relations; they are enhanced versions of the semi-elective quotient introduced in [20] for alternating Büchi automata.

Let $\mathcal{Q}$ be an APA as in (2), $\leq$ a preorder on its state space, and $\equiv$ the corresponding equivalence relation. The *min semi-elective quotient* of $\mathcal{Q}$ with respect to $\leq$ is

$$\mathcal{Q}^{se}_{min} = (Q/{\equiv}, \Sigma, \bar{q}_I, \Delta^{\min}, E^{\min}, U^{\min}, \Omega^{\min}) \tag{5}$$

defined by

$$\Delta^{\min} = \{(\bar{q}, a, \bar{q}') \mid (q, a, q') \in \Delta, q \in E\} \cup \{(\bar{q}, a, \bar{q}') \mid \bar{q} \subseteq U \wedge q' \in \min_a(q)\} \ ,$$
$$U^{\min} = \{\bar{q} \mid \bar{q} \subseteq U\} \ , \qquad E^{\min} = \{\bar{q} \mid \bar{q} \cap E \neq \emptyset\} \ ,$$
$$\Omega^{\min} \colon \bar{q} \mapsto \min\{\Omega(q') \mid q' \in \bar{q}\} \ ,$$

where $\min_a(q) = \{q' \mid (q,a,q') \in \Delta \wedge \forall q''((q,a,q'') \in \Delta \wedge q'' \le q' \rightarrow q' \le q'')\}$, that is, $\min_a(q)$ denotes the set of minimum a-successors of q. The *max semi-elective quotient* is defined in the same way with the only exceptions that min is replaced by max and the roles of U and E are exchanged.

We next apply these quotients to our biased delayed simulations. Given an APA $\mathcal{Q}$, the *even semi-elective quotient* of $\mathcal{Q}$ is the min semi-elective quotient with respect to $\le^e_{de}$, and the *odd semi-elective quotient* of $\mathcal{Q}$ is the max semi-elective quotient with respect to $\le^o_{de}$, denoted $\mathcal{Q}^e_{de}$ and $\mathcal{Q}^o_{de}$, respectively.

Theorem 3 (even and odd semi-elective quotients) *For every APA $\mathcal{Q}$, the even and odd semi-elective quotients of $\mathcal{Q}$ are simulation preserving, in particular, $L(\mathcal{Q}) = L(\mathcal{Q}^e_{de}) = L(\mathcal{Q}^o_{de})$.*

The proof of this theorem is similar to the proof of the correctness of other quotients, but, technically, it is more involved.

We mention that, in general, $\mathcal{Q} \not\equiv^o_{de} \mathcal{Q}^e_{de}$ and $\mathcal{Q} \not\equiv^e_{de} \mathcal{Q}^o_{de}$. Therefore, there is no obvious way to combine the two quotients or to perform one after the other without recomputing one of the biased simulation relations.

5 A Simulation-Based Simplification Algorithm

We conclude by describing a heuristic for reducing the number of states of a given APA. There are three parts to this heuristic: the quotienting procedures from the previous section; simplification methods based on the general delayed simulation relation from Sect. 3; basic simplification methods.

5.1 Basic Simplification Methods

There are two basic simplification methods. To describe them, let $\mathcal{Q}$ be an APA as in (2) and let $q \rightsquigarrow q'$ denote that there is a path from q to q' (with any labeling).

Reachability reduction. Remove all states q where $q_I \not\rightsquigarrow q$.

Normalization. Repeatedly redefine Ω by $\Omega(q) := \Omega(q) - 2$ for a state q such that $\Omega(q) \ge 2$ and there is no state q' with $\Omega(q') = \Omega(q) - 1$ in the same SCC.

Clearly, these methods are correct, that is, they do not change the language of the given APA. Also, they can be implemented quite efficiently.

5.2 Delayed Simplifications

Next, we describe simplification methods based on delayed simulation. Let $\bar{q}$ be the equivalence class of a state q with respect to the equivalence relation $\equiv_{de}$ corresponding to $\le_{de}$.

Homogenization. Redefine Ω by $\Omega(q) = \min\{\Omega(q') \mid q \equiv_{de} q'\}$.

Stretching. For every equivalence class C of $\equiv_{de}$ choose $\mu(C) \in C$ such that there exists no $q' \in C$ satisfying $\mu(C) \rightsquigarrow q' \not\rightsquigarrow \mu(C)$, that is, a maximum representative with respect to reachability. Replace every transition (q,a,q') by $(q,a,\mu(\bar{q}'))$.

0-1-minimaxing. For every $q \in E$ with $\Omega(q) = 0$, remove every transition (q, a, q') where $q' \notin \max_a(q)$. Symmetrically, for every $q \in U$ with $\Omega(q) = 1$, remove every transition (q, a, q') where $q' \notin \min_a(q)$.

In addition, repeatedly remove a transition (q, a, q') with $q \in E$ if there exists $q'' \neq q'$ such that (q, a, q'') is a transition, $q' \leq_{de} q''$, and $\Omega(q'') = 0$. Symmetrically, repeatedly remove a transition (q, a, q') with $q \in U$ if there exists $q'' \neq q'$ such that (q, a, q'') is a transition, $q' \geq_{de} q''$ and $\Omega(q'') = 1$.

Reachability minimaxing. For every $q \in E$ remove a transition (q, a, q') if there exists a transition (q, a, q'') such that $q' \leq_{de} q''$ and $q'' \not\rightsquigarrow q$. Symmetrically, for every $q \in U$ remove a transition (q, a, q') if there exists a transition (q, a, q'') such that $q' \geq_{de} q''$ and $q'' \not\rightsquigarrow q$.

We prove that all these methods are correct. It is easy to see that they can be implemented quite efficiently, once $\leq_{de}$ has been computed.

5.3 Heuristic for State-Space Reduction

The simplification methods described above can be combined in many different reasonable ways to reduce the number of states of a given APA, but they only allow to remove states and edges. For a good state-space reduction heuristic, we need to combine them with quotienting as described in the previous section. One reasonable way to do this, which has proved to be useful, is the following:

State-Space Reduction Heuristic SSRH for an APA $\mathcal{Q}$

1. Choose a cut-off threshold $t > 0$.
2. Normalize the APA.
3. Compute $\leq_{de}$ and perform the delayed simplifications from Subsection 5.2 in the same order as described. Delete unreachable states whenever possible (reachability reduction).
4. Compute $\leq_{de}^{e}$ and $\leq_{de}^{o}$ and pass to the even or the odd semi-elective quotient, whichever results in fewer states. (See remark at the end of Sect. 4.)
5. Let $t := t - 1$. If $t > 0$ and the number of states has been reduced, then go to (2), that is, start all over again, else stop.

Parity games are APA's over a unary alphabet. Thus they are useful for testing how well SSRH performs. In general, one cannot expect that SSRH, even when restricted to parity games, will give perfect results, because the computational complexity of determining the winner in a parity game is still unknown. The best one knows is that it belongs to **UP** $\cap$ **co-UP**.

We have studied how SSRH performs when applied to the Jurdziński family of parity games, $\{H_{m,n}\}_{m,n}$, which have proved to be hard for Jurdziński's algorithm for solving parity games, see [15].

Our algorithm performs well on this family:

Theorem 4 *For every choice of m and n and for $t = 2$, on input $H_{m,n}$, SSRH yields a parity game with two positions in a number of steps polynomial in the input size.*

6 Conclusion

We have shown how the concept of delayed simulation can be adapted to alternating parity automata, that it is hardly useful for state-space reduction via quotienting, but that it can also be modified appropriately in order to arrive at a useful state-space reduction heuristic.

One interesting question that we do not know how to answer is whether there is an intermediate relation $\leq$ such that $(\leq_{de}^{e} \cup \leq_{de}^{o}) \subseteq \leq \subseteq \leq_{de}$ and where quotienting is simulation preserving. Also, it would be useful to have a rigorous proof that quotienting with respect to $\leq_{de}$ does not work; one would have to come up with an APA $\mathcal{Q}$ such that no APA with fewer states accepts the same language, but where two different states simulate each other.

References

1. Milner, R.: An algebraic definition of simulation between programs. In Cooper, D.C., ed.: Proc. 2nd Internat. Joint Conf. on Artificial Intelligence, London, UK, William Kaufmann (1971) 481–489
2. Etessami, K., Holzmann, G.: Optimizing Büchi automata. In Palamidessi, C., ed.: 11th Int. Conf. on Concurrency Theory (CONCUR 2000), University Park, PA, USA. Vol. 1877 of LNCS, Springer, Berlin (2000) 153–167
3. Etessami, K.: (Temporal massage parlor) available at `http://www.bell-labs.com/project/TMP/`.
4. Somenzi, F., Bloem, R.: Efficient Büchi automata from LTL formulae. In Emerson, E.A., Sistla, A.P., eds.: Computer Aided Verification, 12th Internat. Conf., CAV 2000, Chicago, IL, USA. Vol. 1855 of LNCS, Springer, Berlin (2000) 248–263
5. Gurumurthy, S., Bloem, R., Somenzi, F.: Fair simulation minimization. In Brinksma, E., Guldstrand Larsen, K., eds.: Computer Aided Verification, 14th Internat. Conf., CAV 2002, Copenhagen, Denmark. Vol. 2404 of LNCS, Springer, Berlin (2002) 610–623
6. Bloem, R.: (Wring: an LTL to Buechi translator) available at `http://www.ist.tugraz.at/staff/bloem/wring.html`.
7. Fritz, C.: Constructing Büchi automata from linear temporal logic using simulation relations for alternating Büchi automata. In Ibarra, O.H., Dang, Z., eds.: Implementation and Application of Automata, 8th Internat. Conf., CIAA 2003, Santa Barbara, CA, USA. Vol. 2759 of LNCS, Springer, Berlin (2003) 35–48
8. Fritz, C., Teegen, B.: (LTL $\rightarrow$ NBA (improved version)) available at `http://www.ti.informatik.uni-kiel.de/~fritz/ABA-Simulation/ltl.cgi`.
9. Fritz, C.: Concepts of automata construction from LTL. In Sutcliffe, G., Voronkov, A., eds.: LPAR. Vol. 3835 of LNCS, Springer (2005) 728–742
10. Mostowski, A.W.: Regular expressions for infinite trees and a standard form of automata. In: Computation Theory. Vol. 208 of LNCS. Springer (1984) 157–168
11. Emerson, E.A., Jutla, C.S.: Tree automata, mu-calculus and determinacy (extended abstract). In: Proc. 32nd Ann. Symp. on Foundations of Computer Science (FoCS '91), San Juan, Puerto Rico, IEEE Computer Society Press (1991) 368–377
12. Etessami, K., Wilke, Th., Schuller, R.A.: Fair simulation relations, parity games, and state space reduction for büchi automata. In Orejas, F., Spirakis, P.G., van Leeuwen, J., eds.: ICALP. Vol. 2076 of LNCS, Springer (2001) 694–707

13. Etessami, K., Wilke, Th., Schuller, R.A.: Fair simulation relations, parity games, and state space reduction for Büchi automata. SIAM J. Comput. **34**(5) (2005) 1159–1175
14. Henzinger, T.A., Rajamani, S.K.: Fair bisimulation. In Graf, S., Schwartzbach, M.I., eds.: TACAS. Vol. 1785 of LNCS, Springer (2000) 299–314
15. Jurdziński, M.: Small progress measures for solving parity games. In Reichel, H., Tison, S., eds.: STACS 2000, 17th Ann. Symp. on Theoretical Aspects of Computer Science, Lille, France. Vol. 1770 of LNCS, Springer, Berlin (2000) 290–301
16. Fritz, C.: Simulation-Based Simplification of omega-Automata. PhD thesis, Technische Fakultät der Christian-Albrechts-Universität zu Kiel (2005) available at `http://e-diss.uni-kiel.de/diss_1644/`.
17. Grädel, E., Thomas, W., Thomas Wilke, eds.: Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001]. In Grädel, E., Thomas, W., Thomas Wilke, eds.: Automata, Logics, and Infinite Games. Vol. 2500 of LNCS, Springer (2002)
18. Gurevich, Y., Harrington, L.: Trees, automata, and games. In: 14th ACM Symp. on the Theory of Computing, San Francisco, CA, USA, ACM Press (1982) 60–65
19. Vöge, J., Jurdziński, M.: A discrete strategy improvement algorithm for solving parity games. In Emerson, E.A., Sistla, A.P., eds.: CAV. Vol. 1855 of LNCS, Springer (2000) 202–215
20. Fritz, C., Wilke, Th.: Simulation relations for alternating Büchi automata. Theoretical Computer Science **338**(1–3) (2005) 275–314
21. Calude, C., Calude, E., Khoussainov, B.: Finite nondeterministic automata: Simulation and minimality. Theor. Comput. Sci. **242**(1-2) (2000) 219–235