Int J Softw Tools Technol Transfer (2013) 15:433–454
DOI 10.1007/s10009-012-0224-3

SYNTHESIS

Safety first: a two-stage algorithm for the synthesis of reactive systems

Saqib Sohail · Fabio Somenzi

Published online: 3 February 2012

Abstract In the game-theoretic approach to the synthesis of reactive systems, specifications are often expressed as ω-regular languages. Computing a winning strategy to an infinite game whose winning condition is an ω-regular language is then the main step in obtaining an implementation. Conjoining all the properties of a specification to obtain a monolithic game suffers from the doubly exponential determinization that is required. Despite the success of symbolic algorithms, the monolithic approach is not practical. Existing techniques achieve efficiency by imposing restrictions on the ω-regular languages they deal with. In contrast, we present an approach that achieves improvement in performance through the decomposition of the problem while still accepting the full set of ω-regular languages. Each property is translated into a deterministic ω-regular automaton explicitly while the two-player game defined by the collection of automata is played symbolically. Safety and persistence properties usually make up the majority of a specification. We take advantage of this by solving the game incrementally. Each safety and persistence property is used to gradually construct the parity game. Optimizations are applied after each refinement of the graph. This process produces a compact symbolic encoding of the parity game. We then compose the remaining properties and solve one final game after possibly solving smaller games to further optimize the graph. An implementation is finally derived from the winning strategies computed. We compare the results of our tool to those of the synthesis tool ANZU.

This work was supported in part by SRC contract 2010-TJ-1859.

S. Sohail (✉) · F. Somenzi
University of Colorado at Boulder, Boulder, USA
e-mail: Saqib.Sohail@Colorado.EDU; saqibbs@gmail.com

F. Somenzi
e-mail: Fabio@Colorado.EDU

Keywords LTL games · Parity games · Reactive systems · LTL synthesis · Safety first · Automatic synthesis

1 Introduction

The game-theoretic approach to the synthesis of reactive systems is the focus of renewed attention, thanks to the significant algorithmic advances of the last few years. While the doubly exponential bound established in Pnueli and Rosner's [37] seminal paper suggests that challenges to scalability will persist, there is increasing hope that synthesis algorithms may be applied to the design and diagnosis of intricate, safety-critical protocols.

In the game-theoretic approach to synthesis, the specification of a reactive system is converted into a two-player game. One player represents the environment to which the system must react; the other player represents the system itself. The game is a finite graph and a play is an infinite sequence of vertices. A winning strategy for the system player—if it exists—yields an implementation of the specification.

A system's intended behavior is often described by several simple properties, each given as either a formula in linear temporal logic (LTL) or as an ω-regular automaton. In a naive approach, all formulae and automata are reduced to one deterministic automaton, whose transition structure provides the parity game and whose acceptance condition is taken as the winning condition. This approach suffers from the high cost of determinization [25,38], which is prohibitive for even moderate-sized automata.

Several remedies have been proposed, ranging from restricting specifications to a subset of ω-regular properties that allows one to use algorithms more efficient than the general ones [36][1] to avoiding determinization through

[1] Synthesis of reactive (1) specifications is converted to a Streett (1) game.

alternate constructions [11,12,17,22,24]. In [42], we have presented an approach based on deriving a deterministic parity automaton for each property in the specification, thus avoiding in most cases the application of the expensive determinization procedure to large automata. The price to be paid is the switch from the solution of a parity game [20] to that of a more complex conjunctive generalized parity game—one equipped with multiple parity winning conditions that must be simultaneously satisfied [8]. In [42], we argued that this trade-off is advantageous because the most expensive operation—namely, the solution of the game—can be done symbolically, that is, by an algorithm that manipulates characteristic functions of sets rather than their members. (Determinization, on the other hand, is carried out by an explicit enumeration algorithm.)

Since the number of components of a conjunctive generalized parity winning condition sharply affects the time required to solve the game, one may conjoin some properties before translating them to deterministic automata as long as no blow up occurs. The choice of the properties to be merged is, however, heuristic. In contrast, in this paper we propose a two-stage approach to improve the solution of the generalized parity game. We prove that safety and persistence properties can be dealt with before the rest of the properties without affecting the algorithm's completeness. Safety and persistence properties make up the bulk of most specifications of reactive system. Their presence allows us to improve the worst-case runtime of the game solving algorithm and leads to significant time savings.

The rest of this paper is organized as follows. Section 2 recalls the notions on ω automata, temporal logic, and games that are relevant to this paper. Sections 3 and 4 describe the two-stage algorithm and our implementation. Section 5 discusses related work. Section 6 presents our experimental results and Sect. 7 concludes.

2 Automata, logic and games

2.1 Automata and logic

A finite automaton on ω-words $\langle \Sigma, Q, q_{\mathrm{in}}, \delta, \alpha \rangle$ (an ω-automaton) is defined by a finite alphabet Σ, a finite set of states Q, an initial state $q_{\mathrm{in}} \in Q$, a transition function $\delta : Q \times \Sigma \to 2^Q$ that maps a state and an input letter to a set of possible successors, and an acceptance condition α that describes a subset of Q^ω, that is, a set of infinite sequences of states. A *deterministic* automaton is such that $\delta(q, \sigma)$ is a singleton for all states $q \in Q$ and all letters $\sigma \in \Sigma$. A run of automaton M on ω-word $w = w_0 w_1 \ldots$ is a sequence $q_0, q_1, \ldots$ such that $q_0 = q_{\mathrm{in}}$, and for $i \geq 0$, $q_{i+1} \in \delta(q_i, w_i)$. A run is accepting iff (if and only if) it belongs to the set described by α, and a word is accepted iff it has an accepting run in M. The subset of Σ^ω accepted by M is the language of M.

Several types of acceptance conditions α are in use. We are concerned with Büchi [7], and parity [10,31] acceptance conditions. Both conditions are about the set of states $\mathsf{inf}(\rho)$ that occur infinitely often in a run ρ. A run ρ is accepting for a Büchi acceptance condition $F \subseteq Q$ iff $\mathsf{inf}(\rho) \cap F \neq \emptyset$. A parity acceptance condition assigns a *priority* to each state of the automaton. Letting $[k] = \{i \mid 0 \leq i < k\}$, a parity condition of index k is a function $\pi : Q \to [k]$. A run ρ is accepting iff $\max\{\pi(q) \mid q \in \mathsf{inf}(\rho)\}$ is odd; that is, iff the highest recurring priority is odd [8,9,19]. A deterministic parity automaton is of minimum index if there is no other such automaton for the same language with an acceptance condition of lower index. In the rest of the paper, the parity automata are assumed to be of minimum index [9].

Büchi and parity acceptance conditions may be *generalized*. A generalized Büchi condition consists of a collection $\mathcal{F} \subseteq 2^Q$ of Büchi conditions. A run ρ is accepting for a generalized Büchi condition iff it is accepting according to each $F \in \mathcal{F}$. A generalized parity condition may be either conjunctive or disjunctive and is given as a collection Π of priority functions. A run ρ is accepting according to a conjunctive (disjunctive) condition Π iff it is accepting according to each (some) $\pi \in \Pi$. Disjunctive and conjunctive generalized parity conditions are dual: if one of the two players has a winning condition of one type, the opponent has a winning condition of the other type.

An ω-automaton equipped with a Büchi acceptance condition is called a Büchi automaton; likewise for the other acceptance conditions. In this paper, we adopt popular three-letter abbreviations to designate different types of automata. The first letter of each abbreviation distinguishes non-deterministic (N) from deterministic (D) structures. The second letter denotes the type of acceptance condition: Büchi (B) or parity (P). The final letter indicates that the automata read infinite words (W) or infinite trees (T). As examples, NBW designates a non-deterministic Büchi automaton (on infinite words), while DPW is the acronym for a deterministic parity automaton (also on infinite words).

A subset $S \subseteq \Sigma^\omega$ is a *safety property* iff every word not in S has a prefix that cannot be extended to a word in S. A *liveness property* $L \subseteq \Sigma^\omega$ is such that every finite word in Σ^* can be extended to a word in L. Safety properties correspond to closed sets of the product topology of Σ^ω, while liveness properties correspond to dense sets [1].

NBWs, DPWs, and NPW are equally expressive: they accept all ω-regular languages. DBWs are less expressive; accordingly, determinization of Büchi automata is only possible in general by switching to a more powerful acceptance condition. Piterman has adapted Safra's procedure [38] so that it produces a DPW from an NBW [35]. The construction extends the well-known subset construction for automata on finite words. Rather than labeling each state of the deterministic automaton with a subset of states of the NBW, it

labels it with a tree of subsets. As a result, the upper bound on the number of states of the DPW derived from an NBW with n states is n^{2n+2}. This fast-growing function discourages determinization of large NBWs.

Linear Time Logic (LTL) [28,44] is a popular temporal logic for the specification of non-terminating reactive systems. LTL is a strict subset of ω-regular languages. LTL formulae are built from a set of atomic propositions, Boolean connectives, and basic temporal operators X (next), U (until), and R (releases). Derived operators G (always) and F (eventually) are usually included for convenience. Procedures exist (e.g., [14]) to translate an LTL formula into an NBW that accepts the language defined by the formula. On the other hand, DBWs are not sufficient to translate all of LTL. Sistla [41] has shown that LTL formulae in negation normal form that do not use the until operator define safety properties.

The classification of ω-regular languages was first proposed in [26]. The ω-regular languages are fully described by the sets G_1, F_1, $G_2 \cap F_2$, F_2, G_2 and $G_3 \cap F_3$ which are all very low in the Borel hierarchy. In [32] the authors named these classes as *safety, guarantee, obligation, persistence, recurrence* and *general reactivity*. The authors also provide a syntactic characterization of the LTL formulae belonging to each of these classes. The general reactivity class is layered into sub-classes denoted by general reactivity(n) for $n \geq 0$. The formulae belonging to general reactivity (1)[2] have the following form:

$$\bigwedge_{1 \leq k \leq m} (\mathsf{G}\,\mathsf{F}\, J_k^1) \rightarrow \bigwedge_{1 \leq k \leq n} (\mathsf{G}\,\mathsf{F}\, J_k^2).$$

where J_k^i for $i \in \{1, 2\}$ is a propositional formula.

Deterministic ω-regular automata can be used to define infinite games [43] in several ways. We consider turn-based and input-based two-player games, in which Player 0 (the *antagonist*) and Player 1 (the *protagonist*) move a token along the transitions of the game. If the resulting infinite sequence of states is an accepting play (run) of the game, Player 1 wins, otherwise Player 0 wins. In the rest of the paper, if the identity of the winner is not mentioned then Player 1 is understood.

When synthesizing a reactive system, the properties of the specification describe the constraints on its input and output signals. The input signals are controlled by the environment and the output signals are controlled by the system.

2.2 Input-based games versus turn-based games

Definition 1 An input-based game is

$$G = (\Sigma, Q, D, \delta, \alpha),$$

where Σ is the finite input alphabet, Q is the set of states, $D \subseteq Q \times \Sigma$ specifies the allowed input letters for each state, $\delta : D \rightarrow Q$ is the transition function such that $\forall q \in Q \,.\, \exists \sigma \in \Sigma \,.\, (q, \sigma) \in D$, and α is the winning condition. The alphabet Σ is the Cartesian product $\Sigma_{0d} \times \Sigma_1 \times \Sigma_{0p}$ of a disclosed antagonist alphabet Σ_{0d}, a protagonist alphabet Σ_1, and a private antagonist alphabet Σ_{0p}. Throughout this paper $\sigma_{0d} \in \Sigma_{0d}$, $\sigma_1 \in \Sigma_1$, $\sigma_{0p} \in \Sigma_{0p}$ and $\sigma \in \Sigma$. When the token is in state $q \in Q$, initially Player 0 chooses a letter σ_{0d} such that

$$\exists \sigma_1 \,.\, \exists \sigma_{0p} \,.(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \in D,$$

and discloses it to Player 1; then Player 1 chooses a letter σ_1 such that

$$\exists \sigma_{0p} \,.(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \in D,$$

and discloses it to Player 0; then Player 0 selects a letter σ_{0p} such that $(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \in D$; finally the token moves to $q' = \delta(q, (\sigma_{0d}, \sigma_1, \sigma_{0p}))$.

The specification of a reactive-system is naturally translated into an input-based game. The inputs of the reactive-system are partitioned into two groups, the first group of inputs encodes the letters of the alphabet Σ_{0d}, and the second group of inputs encodes the letters of the alphabet Σ_{0p}. The outputs encode the letters of the alphabet Σ_1. The input-based game and the winning strategy for Player 1 (if it exists) implement a reactive system that satisfies the specification.

Definition 2 A turn-based game is

$$G = (Q, Q_0, Q_1, \delta, \alpha),$$

where Q is the set of states, partitioned into Q_0 (antagonist states) and Q_1 (protagonist states), $\delta : Q \rightarrow 2^Q \backslash \emptyset$ is the transition function and α is the winning condition. In state $q \in Q_i$, Player i moves to a successor q' such that $q' \in \delta(q)$.

Turn-based games are games of perfect information; on the other hand, in an input-based game a player may have full, partial, or no advance knowledge of the other player's choices. The amount of information available to one player obviously affects its ability to win the game. In particular, if Player 1 has no advance knowledge of the opponent's moves, the synthesized reactive system is constrained to be of Moore type. Since this may be restrictive, in our input-based games we assume that Player 1 has partial advance knowledge of the other player's choices. The three-phase selection process allows us to synthesize Mealy controllers whose outputs may depend on some inputs, but not on others.

Definition 3 Turn-based game $\widehat{G} = \{\widehat{Q}, \widehat{Q}_0, \widehat{Q}_1, \hat{\delta}, \hat{\alpha}\}$ is the *associate* of input-based game $G = \{\Sigma, Q, D, \delta, \alpha\}$ if the following conditions hold. $\widehat{Q} = \widehat{Q}_0 \cup \widehat{Q}_1$ with

$$\widehat{Q}_0 = Q \cup \{(q, \sigma_{0d}, \sigma_1) \mid \exists \sigma_{0p} \,.(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \in D\},$$
$$\widehat{Q}_1 = \{(q, \sigma_{0p}, \sigma_1, \sigma_{0p}) \mid (q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \in D\} \cup$$
$$\{(q, \sigma_{0d}) | \exists \sigma_1 \,.\, \exists \sigma_{0p} \,.(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \in D\}.$$

[2] Also known as General Streett(1).

The states in $Q \subseteq \widehat{Q}$ are the *base* states of $\widehat{G}$. The transition function $\hat{\delta} : \widehat{Q} \to 2^{\widehat{Q}}$ is defined by:

$$\hat{\delta}(q) = \{(q, \sigma_{0d}) \mid \exists \sigma_1 \,.\, \exists \sigma_{0p} \,.(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \in D\},$$
$$\hat{\delta}((q, \sigma_{0d})) = \{(q, \sigma_{0d}, \sigma_1) \mid \exists \sigma_{0p} \,.(q, (\sigma_{0d}, \sigma_1, \sigma_1)) \in D\},$$
$$\hat{\delta}((q, \sigma_{0d}, \sigma_1))\{(q, \sigma_{0d}, \sigma_1, \sigma_{0p}) \mid (q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \in D\}$$
$$\hat{\delta}((q, \sigma_{0d}, \sigma_1, \sigma_{0p})) = \{\delta(q, (\sigma_{0d}, \sigma_1, \sigma_{0p}))\}.$$

Let $\gamma : \widehat{Q} \to Q$ be the projection function that maps each base state to itself and any other state of turn-based game $\widehat{G}$ to its first component. We write $\gamma(S)$ for the point-wise extension of γ to sets of states. Let $\Gamma : \widehat{Q}^\omega \to Q^\omega$ map a sequence of states of $\widehat{G}$ to the result of applying γ to every fourth state:

$$\Gamma(\hat{\rho})_n = \gamma(\hat{\rho}_{4n}).$$

Then,

$$\hat{\alpha} = \{\hat{\rho} \in \widehat{Q}^\omega | \Gamma(\hat{\rho}) \in \alpha\}$$

is the winning condition of turn-based game $\widehat{G}$. If $\widehat{G}$ is the associate of G, then we also say that G is the associate of $\widehat{G}$.

Note that $\widehat{G}$ is four-partite. If G is a game with generalized parity winning condition $\Pi = \{\pi_1, \pi_2, \ldots, \pi_k\}$ then $\hat{\alpha}$ can be written as $\hat{\Pi} = \{\hat{\pi}_1, \hat{\pi}_2, \ldots, \hat{\pi}_k\}$, where

$$\forall q \in \widehat{Q} \,.\, \hat{\pi}_k(q) = \pi_k(\gamma(q)).$$

Later in this section (Theorem 1) we shall see that associate $\widehat{G}$ can be solved instead of the input-based game G.

The existence and computation of winning strategies are central problems in the study of infinite games. A strategy is a function that defines the letter or successor of the current state a player should choose at each move. A strategy for Player i in a turn-based game can be defined equivalently as either a function $\tau_i : Q^* \times Q_i \to Q$, or as a function $\tau_i : S_i \times Q_i \to S_i \times Q$. (We use the second form later in this section.) The set S_i is Player i's *memory*, which contains an initial element $\tilde{s}_i$. According to the cardinality of S_i, strategies are classified as *infinite memory*, *finite memory*, and *memoryless* (or *positional*). A strategy τ_i is *winning* for Player i from a given state of the game iff victory is secured from that state regardless of the opponent's choices as long as Player i plays according to τ_i. In an input-based game, a strategy for Player 0 can be defined equivalently as either a pair of functions:

$$\tau_0^1 : Q^* \times Q \to \Sigma_{0d} \quad \text{and}$$
$$\tau_0^2 : Q^* \times Q \times \Sigma_{0d} \times \Sigma_1 \to \Sigma_{0p},$$

or as a pair of functions:

$$\tau_0^1 : S_0 \times Q \to \Sigma_{0d} \quad \text{and}$$
$$\tau_0^2 : S_0 \times Q \times \Sigma_{0d} \times \Sigma_1 \to S_0 \times \Sigma_{0p}.$$

A strategy for Player 1 can be defined equivalently as a function: $\tau_1 : Q^* \times Q \times \Sigma_{0d} \to \Sigma_1$ or as function: $\tau_1 : S_1 \times Q \times \Sigma_{0d} \to S_1 \times \Sigma_1$. (We use the second form later in this section.)

Definition 4 Let $G = (\Sigma, Q, D, \delta, \alpha)$ be an input-based game and $\widehat{G} = (\widehat{Q}, \widehat{Q}_0, \widehat{Q}_1, \hat{\delta}, \hat{\alpha})$ its associate game. Given Player 1's strategy $\hat{\tau}_1 : S_1 \times \widehat{Q}_1 \to S_1 \times \widehat{Q}_0$ for $\widehat{G}$, then

$$\tau_1 : S_1 \times Q \times \Sigma_{0d} \to S_1 \times \Sigma_1$$

is Player 1's *associate* strategy of $\hat{\tau}_1$ defined as follows. Let

$$\hat{\tau}_1(s_1^1, (q, \sigma_{0d})) = (s_1^2, (q, \sigma_{0d}, \sigma_1)),$$
$$\hat{\tau}_1(s_1^2, (q, \sigma_{0d}, \sigma_1, \sigma_{0p})) = (s_1^3, q'),$$

then

$$\tau_1(s_1^1, q, \sigma_{0d}) = (s_1^3, \sigma_1).$$

Given Player 0's strategy $\hat{\tau}_0 : S_0 \times \widehat{Q}_0 \to S_0 \times \widehat{Q}_1$ for $\widehat{G}$, then

$$\tau_0^1 : S_0 \times Q \to \Sigma_{0d}$$
$$\tau_0^2 : S_0 \times Q \times \Sigma_{0d} \times \Sigma_1 \to S_0 \times \Sigma_{0p}$$

is Player 0's *associate* strategy of $\hat{\tau}_0$ defined as follows. Let

$$\hat{\tau}_0(s_0^1, q) = (s_0^2, (q, \sigma_{0d})) \quad \text{and}$$
$$\hat{\tau}_0(s_0^2, (q, \sigma_{0d}, \sigma_1)) = (s_0^3, (q, \sigma_{0d}, \sigma_1, \sigma_{0p})),$$

then

$$\tau_0^1(s_0^1, q) = \sigma_{0d} \quad \text{and}$$
$$\tau_0^2(s_0^1, q, \sigma_{0d}, \sigma_1) = (s_0^3, \sigma_{0p}).$$

Given Player 1's strategy $\tau_1 : S_1 \times Q \times \Sigma_{0d} \to S_1 \times \Sigma_1$ for G,

$$\hat{\tau}_1 : S_1 \times \widehat{Q}_1 \to S_1 \times \widehat{Q}_0$$

is Player 1's *associate* strategy of τ_1 for $\widehat{G}$ defined as follows. Let

$$\tau_1(s_1^1, q, \sigma_{0d}) = (s_1^2, \sigma_1),$$

then

$$\hat{\tau}_1(s_1^1, (q, \sigma_{0d})) = (s_1^1, (q, \sigma_{0d}, \sigma_1)),$$

while the strategy for a state $q \in Q \times \Sigma_{0d} \times \Sigma_1 \times \Sigma_{0p}$ is determined by the transition function δ as follows:

$$\hat{\tau}_1(s_1^1, (q, \sigma_{0d}, \sigma_1, \sigma_{0p})) = (s_1^2, \delta(q, (\sigma_{0d}, \sigma_1, \sigma_{0p}))).$$

Given Player 0's strategy $\tau_0^1 : S_0 \times Q \to \Sigma_{0d}$ and $\tau_0^2 : S_0 \times \Sigma_{0d} \times \Sigma_1 \to S_0 \times \Sigma_{0p}$ for G,

$$\hat{\tau}_0 : S_0 \times \widehat{Q}_0 \to S_0 \times \widehat{Q}_1$$

is Player 0's *associate* strategy of (τ_0^1, τ_0^2) for $\widehat{G}$ defined as follows. Let

$$\tau_0^1(s_0^1, q) = \sigma_{0d},$$
$$\tau_0^2(s_0^1, q, \sigma_{0d}, \sigma_1) = (s_0^2, \sigma_{0p}),$$

then

$\hat{\tau}_0(s_0^1, q) = (s_0^1, (q, \sigma_{0d}))$ and
$\hat{\tau}_0(s_0^1, (q, \sigma_{0d}, \sigma_1)) = (s_0^2, (q, \sigma_{0d}, \sigma_1, \sigma_{0p}))$.

If τ_i is Player i's strategy for G, then $\text{AS}(\tau_i)$ is Player i's associate strategy for the associate game $\widehat{G}$. If $\hat{\tau}_i$ is Player i's strategy for $\widehat{G}$ then $\text{AS}(\hat{\tau}_i)$ is Player i's associate strategy for the associate game G. Note that a strategy and its associate use the same memory. In the rest of the section, if ρ is a play of G then $\rho_{l,m}$ is a segment of ρ such that $\rho_{l,m} = \rho_l, \rho_{l+1}, \ldots, \rho_{m-1}, \rho_m$.

We now show that the strategy $\text{AS}(\tau_i)$ is winning for Player i iff τ_i is. The following two lemmas prove that using associate strategies preserves the outcome of a play.

Lemma 1 *Let $G = (\Sigma, Q, D, \delta, \alpha)$ be an input-based game and $\widehat{G} = (\widehat{Q}, \widehat{Q}_0, \widehat{Q}_1, \hat{\delta}, \hat{\alpha})$ be its associate game. Let $\hat{\rho}$ be the play of $\widehat{G}$ starting from base state $q \in Q$ when, for $i \in \{0, 1\}$, Player i plays according to strategy $\hat{\tau}_i$. If ρ is the play of G starting from q when Player i plays according to strategy $\text{AS}(\hat{\tau}_i)$, then $\rho = \Gamma(\hat{\rho})$.*

Proof For the play ρ of G, let ρ^{S_i} be the corresponding sequence of Player i's memory states. For the play $\hat{\rho}$ of $\widehat{G}$, let $\hat{\rho}^{S_i}$ be the corresponding sequence of Player i's memory states. We prove by induction on length of the play ρ and the corresponding sequence of Player i's memory states ρ^{S_i} that $\rho = \Gamma(\hat{\rho})$ and $\forall n \in N \,.\, \rho_n^{S_i} = \hat{\rho}_{4n}^{S_i}$. Since the initial states of the runs $\hat{\rho}$ and ρ are the same, while Player i's memory is always initialized to $\tilde{s}_i$, the base case holds.

For the inductive step, let $\rho_{0,n} = \Gamma(\hat{\rho}_{0,4n})$ and $\forall x \,.(0 \leq x \leq n) \rightarrow \rho_x^{S_i} = \hat{\rho}_{4x}^{S_i}$. Let

$\hat{\rho}_{4n} = \rho_n = u$
$\hat{\rho}_{4n}^{S_0} = \rho_n^{S_0} = s_0^1$ and
$\hat{\rho}_{4n}^{S_1} = \rho_n^{S_1} = s_1^1$.

Suppose Player i plays according to

$\hat{\tau}_0(s_0^1, u) = (s_0^2, (u, \sigma_{0d}))$,
$\hat{\tau}_1(s_1^1, (u, \sigma_{0d})) = (s_1^2, (u, \sigma_{0d}, \sigma_1))$,
$\hat{\tau}_0(s_0^2, (u, \sigma_{0d}, \sigma_1)) = (s_0^3, (u, \sigma_{0d}, \sigma_1, \sigma_{0p}))$ and
$\hat{\tau}_1(s_1^2, (u, \sigma_{0d}, \sigma_1, \sigma_{0p})) = (s_1^3, v)$.

then

$\hat{\rho}_{4n+1} = (u, \sigma_{0d}), \quad \hat{\rho}_{4n+1}^{S_0} = s_0^2, \quad \hat{\rho}_{4n+1}^{S_1} = s_1^1,$
$\hat{\rho}_{4n+2} = (u, \sigma_{0d}, \sigma_1), \quad \hat{\rho}_{4n+2}^{S_0} = s_0^2, \quad \hat{\rho}_{4n+2}^{S_1} = s_1^2,$
$\hat{\rho}_{4n+3} = (u, \sigma_{0d}, \sigma_1, \sigma_{0p}), \quad \hat{\rho}_{4n+3}^{S_0} = s_0^3, \quad \hat{\rho}_{4n+3}^{S_1} = s_1^2$

and

$\hat{\rho}_{4n+4} = v, \quad \hat{\rho}_{4n+4}^{S_0} = s_0^3, \quad \hat{\rho}_{4n+4}^{S_1} = s_1^3.$

Let $\tau_i = \text{AS}(\hat{\tau}_i)$. By Definition 4

$\tau_0^1(s_0^1, u) = \sigma_{0d}$,
$\tau_1(s_1^1, u, \sigma_{0d}) = (s_1^3, \sigma_1)$ and
$\tau_0^2(s_0^1, u, \sigma_{0d}, \sigma_1) = (s_0^3, \sigma_{0p})$.

By Definition 3, $\delta(u, (\sigma_{0d}, \sigma_1, \sigma_{0p})) = v$. Hence $\rho_{n+1} = v$, $\rho_{n+1}^{S_0} = s_0^3$ and $\rho_{n+1}^{S_1} = s_1^3$. Thus we have $\rho_{0,n+1} = \Gamma(\hat{\rho}_{0,4n+4})$ and $\forall x (0 \leq x \leq n+1) \rightarrow \rho_x^{S_i} = \hat{\rho}_{4x}^{S_i}$. Therefore, we can conclude $\rho = \Gamma(\hat{\rho})$ which by definition of $\hat{\alpha}$ implies that $\hat{\rho}$ is winning iff ρ is winning. □

Lemma 2 *Let $G = (\Sigma, Q, D, \delta, \alpha)$ be an input-based game and $\widehat{G} = (\widehat{Q}, \widehat{Q}_0, \widehat{Q}_1, \hat{\delta}, \hat{\alpha})$ be its associate game. Let ρ be the play of G starting from $q \in Q$ when, for $i \in \{0, 1\}$, Player i plays according to strategy τ_i. If $\hat{\rho}$ is the play of $\widehat{G}$ starting in q when Player i plays according to strategy $\text{AS}(\tau_i)$, then $\Gamma(\hat{\rho}) = \rho$.*

Proof For the play ρ of G, let ρ^{S_i} be the corresponding sequence of Player i's memory states. For the play $\hat{\rho}$ of $\widehat{G}$, let $\hat{\rho}^{S_i}$ be the corresponding sequence of Player i's memory states. We prove on length of the play $\hat{\rho}$ and the corresponding sequence of Player i's memory states $\hat{\rho}^{S_i}$ that $\rho = \Gamma(\hat{\rho})$ and $\forall n \in N \rightarrow \rho_n^{S_i} = \hat{\rho}_{4n}^{S_i}$. Since the starting states of both runs ρ and $\hat{\rho}$ are the same, while the memory is always initialized to $\tilde{s}_i$, the base case holds. For the inductive step, let $\Gamma(\hat{\rho}_{0,4n}) = \rho_{0,n}$ and $\forall x \,.(0 \leq x \leq n) \rightarrow \hat{\rho}_{4x}^{S_i} = \rho_x^{S_i}$. Let

$\rho_n = \hat{\rho}_{4n} = u$
$\rho_n^{S_0} = \hat{\rho}_{4n}^{S_0} = s_0^1$ and
$\rho_n^{S_1} = \hat{\rho}_{4n}^{S_1} = s_1^1$.

Suppose

$\tau_0^1(s_0^1, u) = \sigma_{0d}$,
$\tau_1(s_1^1, u, \sigma_{0d}) = (s_1^2, \sigma_1)$,
$\tau_0^2(s_0^1, u, \sigma_{0d}, \sigma_1) = (s_0^2, \sigma_{0p})$ and
$\delta(u, (\sigma_{0d}, \sigma_1, \sigma_{0p})) = v$,

then $\rho_{n+1} = v$, $\rho^{S_0} = s_0^2$ and $\rho^{S_1} = s_1^2$. Let $\hat{\tau}_i = \text{AS}(\tau_i)$. Then, by Definition 4

$\hat{\tau}_0(s_0^1, u) = (s_0^1, (u, \sigma_{0d}))$,

which means that

$\hat{\rho}_{4n+1} = (u, \sigma_{0d}), \quad \hat{\rho}_{4n+1}^{S_0} = s_0^1$ and $\hat{\rho}_{4n+1}^{S_1} = s_1^1$.

Then, by Definition 4

$\hat{\tau}_1(s_1^1, (u, \sigma_{0d})) = (s_1^1, (u, \sigma_{0d}, \sigma_1))$,

which means that

$\hat{\rho}_{4n+2} = (u, \sigma_{0d}, \sigma_1), \quad \hat{\rho}_{4n+2}^{S_0} = s_0^1$ and $\hat{\rho}_{4n+2}^{S_1} = s_1^1$.

Then, by Definition 4

$\hat{\tau}_0(s_0^1, (u, \sigma_{0d}, \sigma_1)) = (s_0^2, (v, \sigma_{0d}, \sigma_1, \sigma_{0p}))$,

which means that

$$\hat{\rho}_{4n+3} = (u, \sigma_{0d}, \sigma_1, \sigma_{0p}),\ \hat{\rho}^{S_0}_{4n+3} = s^2_0 \quad \text{and} \quad \hat{\rho}^{S_1}_{4n+3} = s^1_1.$$

Finally, by Definition 4

$$\hat{\tau}_0(s^1_1, (u, \sigma_{0d}, \sigma_1, \sigma_{0p})) = (s^2_1, v),$$

which means that

$$\hat{\rho}_{4n+4} = v, \quad \hat{\rho}^{S_0}_{4n+4} = s^2_0 \quad \text{and} \quad \hat{\rho}^{S_1}_{4n+4} = s^2_1.$$

Thus we have $\Gamma(\hat{\rho}_{0,4n+4}) = \rho_{0,n+1}$ and $\forall x\,.(0 \le x \le n+1) \rightarrow \hat{\rho}^{S_i}_{4x} = \rho^{S_i}_x$. Therefore, we can conclude $\Gamma(\hat{\rho}) = \rho$, which by definition of $\hat{\alpha}$ implies that $\hat{\rho}$ is winning iff ρ is winning. □

If τ_i is a strategy for Player i in an input-based game G, then Lemma 3 proves that $\text{AS}(\text{AS}(\tau_i)) = \tau_i$. On the other hand if $\hat{\tau}_i$ is a strategy for Player i in the associate game $\widehat{G}$, then Lemma 4 proves that even though $\text{AS}(\text{AS}(\hat{\tau}_i))$ may differ from $\hat{\tau}_i$, the two strategies are interchangeable.

Lemma 3 *Let $G = (\Sigma, Q, D, \delta, \alpha)$ be an input-based game and $\widehat{G} = (\widehat{Q}, \widehat{Q}_0, \widehat{Q}_1, \hat{\delta}, \hat{\alpha})$ be its associate game. If τ_i is the strategy of Player i in G, then $\text{AS}(\text{AS}(\tau_i)) = \tau_i$.*

Proof Suppose

$$\tau^1_0(s^1_0, u) = \sigma_{0d},$$
$$\tau_1(s^1_1, u, \sigma_{0d}) = (s^2_1, \sigma_1) \quad \text{and}$$
$$\tau^2_0(s^1_0, u, \sigma_{0d}, \sigma_1) = (s^2_0, \sigma_{0p}).$$

By Definition 4

$$\text{AS}(\tau_0)(s^1_0, u) = (s^1_0, (u, \sigma_{0d})),$$
$$\text{AS}(\tau_1)(s^1_1, (u, \sigma_{0d})) = (s^1_1, (u, \sigma_{0d}, \sigma_1)),$$
$$\text{AS}(\tau_0)(s^1_0, (u, \sigma_{0d}, \sigma_1)) = (s^2_0, (u, \sigma_{0d}, \sigma_1, \sigma_{0p})) \quad \text{and,}$$
$$\text{AS}(\tau_1)(s^1_1, (u, \sigma_{0d}, \sigma_1, \sigma_{0p})) = (s^1_1, v).$$

By Definition 4

$$\text{AS}(\text{AS}(\tau^1_0))(s^1_0, u) = \sigma_{0d},$$
$$\text{AS}(\text{AS}(\tau_1))s^1_1, u, \sigma_{0d}) = (s^2_1, \sigma_1) \quad \text{and}$$
$$\text{AS}(\text{AS}(\tau^2_0))(s^1_0, u, \sigma_{0d}, \sigma_1) = (s^2_0, \sigma_{0p}).$$

Hence $\text{AS}(\text{AS}(\tau_i)) = \tau_i$. □

Lemma 4 *Let $G = (\Sigma, Q, D, \delta, \alpha)$ be an input-based game and $\widehat{G} = (\widehat{Q}, \widehat{Q}_0, \widehat{Q}_1, \hat{\delta}, \hat{\alpha})$ be its associate. Let $\hat{\rho}$ be the play of $\widehat{G}$ starting in $q \in \widehat{Q}$ when Player k plays according to $\hat{\tau}_k$. For $i \in \{0, 1\}$, let $j = \neg i$, and let ρ^i be the play of $\widehat{G}$ starting in $q \in \widehat{Q}$ by Player i playing according to $\hat{\tau}_i$ and Player j playing according to $\text{AS}(\text{AS}(\tau'_j))$. Then, $\rho^0 = \rho^1 = \hat{\rho}$.*

Proof For the play $\hat{\rho}$ of $\widehat{G}$, let $\hat{\rho}^{S_i}$ be the corresponding sequence of Player i's memory states. For the play ρ^0 of $\widehat{G}$, let ρ^{0,S_i} be the corresponding sequence of Player i's memory states. Likewise, for the play ρ^1 of $\widehat{G}$, let ρ^{1,S_i} be the corresponding sequence of Player i's memory states. We prove by induction on the length of the runs and the corresponding sequence of Player i's memory states that for $p \in \{0, 1\}$, $\hat{\rho} = \rho^p$ and $\forall n \in N\,.\,\hat{\rho}^{S_i}_{4n} = \rho^{p,S_i}_{4n}$.

Since both runs start in the same state, while the memory is always initialized to $\tilde{s}_i$, the base case holds. For the inductive step, let $\hat{\rho}_{0,4n} = \rho^p_{0,4n}$ and $\forall x\,.\,0 \le x \le n \rightarrow \hat{\rho}^{S_i}_{4x} = \rho^{p,S_i}_{4x}$. Suppose we have $\hat{\rho}_{4n} = u$, $\hat{\rho}^{S_0}_{4n} = s^1_0$ and $\hat{\rho}^{S_1}_{4n} = s^1_1$; then $\rho^i_{4n} = u$, $\rho^{p,S_0}_{4n} = s^1_0$ and $\rho^{p,S_1}_{4n} = s^1_1$. Suppose

$$\hat{\tau}_0(s^1_0, u) = (s^2_0, (u, \sigma_{0d})),$$
$$\hat{\tau}_1(s^1_1, (u, \sigma_{0d})) = (s^2_1, (u, \sigma_{0d}, \sigma_1)),$$
$$\hat{\tau}_0(s^2_0, (u, \sigma_{0d}, \sigma_1)) = (s^3_0, (u, \sigma_{0d}, \sigma_1, \sigma_{0p})) \quad \text{and,}$$
$$\hat{\tau}_1(s^2_1, (u, \sigma_{0d}, \sigma_1, \sigma_{0p})) = (s^3_1, v).$$

This implies that

$$\hat{\rho}_{4n+1} = (u, \sigma_{0d}), \quad \hat{\rho}^{S_0}_{4n+1} = s^2_0, \quad \hat{\rho}^{S_0}_{4n+1} = s^1_1$$
$$\hat{\rho}_{4n+2} = (u, \sigma_{0d}, \sigma_1), \quad \hat{\rho}^{S_0}_{4n+2} = s^2_0, \quad \hat{\rho}^{S_1}_{4n+2} = s^2_1$$
$$\hat{\rho}_{4n+3} = (u, \sigma_{0d}, \sigma_1, \sigma_{0p}), \quad \hat{\rho}^{S_0}_{4n+3} = s^3_0, \quad \hat{\rho}^{S_1}_{4n+3} = s^2_1$$

and

$$\hat{\rho}_{4n+4} = v, \quad \hat{\rho}^{S_0}_{4n+4} = s^3_0, \quad \hat{\rho}^{S_1}_{4n+4} = s^3_1.$$

Definition 4

$$\text{AS}(\hat{\tau}^1_0)(s^1_0, u) = \sigma_{0d},$$
$$\text{AS}(\hat{\tau}_1)(s^1_1, u, \sigma_{0d}) = (s^3_1, \sigma_1) \quad \text{and}$$
$$\text{AS}(\hat{\tau}^2_0)(s^1_0, u, \sigma_{0d}, \sigma_1) = (s^3_0, \sigma_{0p}).$$

By Definition 4

$$\text{AS}(\text{AS}(\hat{\tau}_0))(s^1_0, u) = (s^1_0, (u, \sigma_{0d})),$$
$$\text{AS}(\text{AS}(\hat{\tau}_1))(s^1_1, (u, \sigma_{0d})) = (s^1_1, (u, \sigma_{0d}, \sigma_1)),$$
$$\text{AS}(\text{AS}(\hat{\tau}_0))(s^1_0, (u, \sigma_{0d}, \sigma_1)) = (s^3_0, (u, \sigma_{0d}, \sigma_1, \sigma_{0p})$$

and

$$\text{AS}(\text{AS}(\hat{\tau}_1))(s^1_1, (u, \sigma_{0d}, \sigma_1, \sigma_{0p})) = (s^3_1, v).$$

If Player 1 plays according to $\text{AS}(\text{AS}(\hat{\tau}_1))$ and Player j plays according to $\hat{\tau}_0$, then we have

$$\rho^1_{4n+1} = (u, \sigma_{0d}), \quad \rho^{1,S_0}_{4n+1} = s^1_0, \quad \rho^{1,S_1}_{4n+1} = s^1_1,$$
$$\rho^1_{4n+2} = (u, \sigma_{0d}, \sigma_1), \quad \rho^{1,S_0}_{4n+2} = s^1_0, \quad \rho^{1,S_1}_{4n+2} = s^1_1,$$
$$\rho^1_{4n+3} = (u, \sigma_{0d}, \sigma_1, \sigma_{0p}), \quad \rho^{1,S_0}_{4n+3} = s^3_0, \quad \rho^{1,S_1}_{4n+3} = s^1_1,$$

and

$$\rho^1_{4n+4} = v, \quad \rho^{1,S_0}_{4n+4} = s^3_0, \quad \rho^{1,S_1}_{4n+1} = s^3_1.$$

Hence $\rho^1_{0,4n+4} = \rho'_{4n+4}$. Similarly, if Player 0 plays according to $\text{AS}(\text{AS}(\hat{\tau}_0))$ and Player 1 plays according to $\hat{\tau}_1$, then we have $\hat{\rho}_{0,4n+4} = \rho^0_{0,4n+4}$. Therefore, $\rho'_{0,4n+4} = \rho^i_{0,4n+4}$. Hence we conclude that $\hat{\rho} = \rho^i$. □

Finally, the following theorem proves that a winning strategy for Player i in G can be translated to a winning strategy for the associate game $\widehat{G}$.

Theorem 1 *A strategy $\hat{\tau}_i$ for Player i in $\widehat{G}$ is winning iff the strategy $\text{AS}(\hat{\tau}_i)$ for Player i in G is winning. Conversely, a strategy τ_i for Player i in G is winning iff the strategy $\text{AS}(\tau_i)$ for Player i in $\widehat{G}$ is winning.*

Proof For $i \in \{0, 1\}$, let $j = \neg i$, let $\hat{\tau}_i$ be a winning strategy for Player i in $\widehat{G}$. Let Player i play according to $\text{AS}(\hat{\tau}_i)$ in G and suppose it loses to Player j's strategy τ_j. Let $\hat{\rho}$ be the play of $\widehat{G}$ produced by $\text{AS}(\text{AS}(\hat{\tau}_i))$ and $\text{AS}(\tau_j)$. From Lemma 2, $\hat{\rho}$ is a losing play for Player i. From Lemma 4, $\text{AS}(\text{AS}(\hat{\tau}_i))$ and $\hat{\tau}_i$ produce an identical play when Player j uses a fixed strategy. Since $\hat{\tau}_i$ is a winning strategy, we have reached a contradiction. Therefore, our assumption is wrong and there does not exist a winning strategy for Player j in G when Player i plays according to $\text{AS}(\hat{\tau}_i)$. Hence $\text{AS}(\hat{\tau}_i)$ is a winning strategy for Player i in G.

Similarly, it is proved that τ_i for Player i in G is winning iff the strategy $\text{AS}(\tau_i)$ for Player i in $\widehat{G}$ is winning. □

The existing literature focuses on turn-based games. We will leverage results for turn-based games in the rest of this section and Sect. 3. In Sect. 4.1, we prove that it is not necessary to explicitly play the associate turn-based game. We implicitly play the associate turn-based game using the representation of the input-based game.

2.3 From LTL games to parity games

If the winning condition of an infinite game is given as an LTL formula on the states of the game, then an *LTL game* is obtained. Such a game can be solved by translating the formula into a deterministic ω-automaton and composing it with the graph of the given game. Not all LTL formulae have an equivalent DBW. Therefore, determinization to a more powerful type of automaton is required in general. With Piterman's improvement of Safra's construction [35,38], the parity condition is the natural choice.

Since determinization is expensive, there have been attempts to circumvent it based on the notion of fair simulation [15]. An ω-automaton $P = (\Sigma, Q^P, q^P_{\text{in}}, \delta^P, \alpha^P)$ *fair simulates* another ω-automaton $A = (\Sigma, Q^A, q^A_{\text{in}}, \delta^A, \alpha^A)$ with the same alphabet if Player 1 (protagonist) has a winning strategy for the following turn-based game: Initially, the antagonist token is placed on q^A_{in} and the protagonist token is placed on q^P_{in}. At each turn, let $p \in Q^P$ be the state with the protagonist token and let $a \in Q^A$ be the state with the antagonist token. Player 0 chooses a letter $\sigma \in \Sigma$ and moves the A token to one of the states in $\delta_A(a, \sigma)$. Player 1 then moves the P token to one of the states in $\delta_P(p, \sigma)$. Player 1 wins if either the run of A is not in α^A or the run of P is in α^P. A winning strategy for Player1 is a function $\tau : (Q^A \times Q^P \times \Sigma)^+ \to Q^P$ that is consistent with δ^P ($\forall a \in Q^A . \forall p \in Q^P . \forall \sigma . \tau(a, p, \sigma) \in \delta^P(p, \sigma)$) and that guarantees victory regardless of the opponent's choices.

In [42, Theorem 1] it is proved that an NBW cannot fair simulate a DPW with (minimum) index greater than two. The following theorem is a generalization of [42, Theorem 1]; a DPW of (minimum) index k, cannot be fair simulated by any NPW with index less than k.

Theorem 2 *Let $L \subseteq \Sigma^\omega$ be an ω-regular language. Let $N = (\Sigma, Q^N, \delta^N, \pi^N)$ be an NPW accepting L with $\pi^N : Q^N \to [k]$ and $D = (\Sigma, Q^D, \delta^D, \pi^D)$ be a DPW accepting L with $\pi^D : Q^D \to [k']$. Let k' be the minimum number of colors for L and $k < k'$. Then N does not fair simulate D.*

Proof Suppose a fair simulation strategy $\tau : Q^N \times Q^D \times \Sigma \to Q^N$ exists for N. (W.l.o.g. τ is positional because the winning condition is the disjunction of two parity conditions.) Consider the DPW $C = (\Sigma, Q, \delta, \pi)$, where

$$Q = Q^N \times Q^D$$
$$\delta((q^N, q^D), \sigma) = (\tau(q^N, q^D, \sigma), \delta^D(q^D, \sigma))$$
$$\pi((q^N, q^D)) = \pi^N(q^N).$$

Suppose $w \in \Sigma^\omega$ is accepted by C. Let ρ be the play of N on the word w when obeying the strategy τ. The play ρ is the projection on Q^N of the play of C, moreover it is accepting because C accepts. Therefore, $w \in L$. Conversely, if $w \in L$, its play in D is accepting. Since τ is a simulation strategy for N, the play of N on w according to τ is also accepting. Hence the play of C on w is also accepting and C is a DPW that accepts L such that $\pi : Q \to [k]$. This, however, contradicts the assumption that $k' > k$ is minimum for L. Hence τ does not exits. □

In [18], the authors present a method in which the NBW need not be determinized. A *shift automaton*, which offers the ability to rectify a non-deterministic choice, is composed with the NBW. If the resulting three-color NPW embeds (hence, fair simulates) a DPW for the given property, then the NPW can be used to solve the game. Although the approach of [18] avoids determinization, it is not complete (in particular, for properties for which the minimum index of a DPW is three) and the NPW is roughly the same size of the DPW obtained by the procedure [35].

In [17], the authors proved that for the purpose of synthesis of reactive system, the translation of a property from the specification to a non-deterministic *good for games (GFG)* parity automaton can be composed with the graph of the given game when the non-deterministic automaton fair simulates the deterministic translation of the property. Although the approach of [17] avoids determinization, the GFG parity automaton may be larger than its deterministic counterpart.

Secondly, the GFG parity automaton cannot use fewer colors than its deterministic counterpart. Thus, avoiding determinization may incur an additional cost during game play.

2.4 Solving parity games

We consider parity games, and their generalized counterpart. All these games are *determinate* [30]; that is, from each state of the game if a player has no winning strategy then the opponent has one. Non-generalized parity and disjunctive generalized parity games admit memoryless strategies. Conjunctive generalized parity games admit finite memory strategies [45].

In [42], we have presented an algorithm that takes as input a collection of LTL formulae and NBWs over an alphabet $\Sigma = \Sigma_0 \times \Sigma_1$. The input is converted into a conjunctive generalized parity game with one parity function for each formula and automaton. At each turn, Player 0 chooses a letter from Σ_0 and Player 1 chooses a letter from Σ_1. The objective of Player 1 is to satisfy the conjunctive generalized parity acceptance condition. A winning strategy for Player 1 from the initial state of the parity automaton thus corresponds to an implementation of a reactive system that reads inputs from alphabet Σ_0 and produces outputs from alphabet Σ_1. The reactive system satisfies all the linear-time properties given as LTL formulae or Büchi automata from its initial state. If no such winning strategy exists, there exists no implementation of the given specification. The "classical" algorithm described in [8] is used to compute winning strategies for generalized parity conditions. This algorithm is based on [16], which in turn extends Zielonka's algorithm for parity conditions [45].

Zielonka's algorithm is representative of a class of procedures used to compute winning strategies of a parity game. Though other algorithms have better worst-case complexity, it has the advantage of lending itself to efficient symbolic implementation. Zielonka's algorithm, when applied to the solution of Büchi and co-Büchi games (regarded as two-color parity games), is as efficient as dedicated (symbolic) algorithms. It is simpler—and therefore easier to present succinctly—than the procedure of [16] or the "classical" procedure of [8]; however, it is close enough to those algorithms that its discussion sheds light on them as well.

The pseudo code shown in Fig. 1 extends the one given in [19] with the computation of the strategies. The reader is referred to [19] for a detailed explanation of its operation. Here we focus on strategy computation, which is related to our approach. The algorithm of Fig. 1 takes as input a parity game G, whose set of vertices Q is partitioned into Q_0 and Q_1, and a parity condition; it returns the winning positions of each player and two sets of transitions T_0 and T_1 of the parity game G such that the transitions in T_i belong to winning strategies of Player i. Specifically, T_i contains at least a

```
1  algorithm ZIELONKA(G)
2  if G = (∅, ∅) then return (∅, ∅, ∅, ∅)
3  d ← d(G); A ← A_d(G)
4  i ← d mod 2; j ← ¬i
5  (A_i, Δ_i) ← attr_i(G, A)
6  (U_0, U_1, T_0, T_1) ← ZIELONKA(G \ A_i)
7  if (U_j = ∅) then
8      U_i ← V(G)
9      T_i ← Δ_i ∪ T_i ∪ {(u, v) ∈ E(G) | u ∈ A ∩ Q_i}
10 else
11     (A_j, Δ_j) ← attr_j(G, U_j)
12     (U_0, U_1, W_0, W_1) ← ZIELONKA(G \ A_j)
13     U_j ← V(G) \ U_i
14     T_i = W_i
15     T_j = T_j ∪ Δ_j ∪ W_j
16 fi
17 return (U_0, U_1, T_0, T_1)
```

Fig. 1 Zielonka's algorithm for parity games. In the code, $d(G)$ is the largest color of G and $A_d(G)$ is the set of vertices of G with color d

transition from each vertex in $U_i \cap Q_i$, and any strategy for Player i that is consistent with T_i is a winning strategy from all winning positions of Player i. A strategy σ is consistent with T_i iff $\forall u \in (U_i \cap Q_i) . (u, \sigma(u)) \in T_i$.

A key step in the algorithm of Fig. 1 is the computation of the i-attraction in G of a set of vertices A, that is, the set of vertices $\mathrm{attr}_i(G, A)$ of G from which Player i can force a visit of some vertex in A.

Definition 5 Given a turn-based game $G = (Q, Q_0, Q_1, \delta, \alpha)$ and the set of states $A \subseteq Q$, the set $\mathcal{A}_i \subseteq Q$ is the i-attraction of A in G if it is the least set such that

1. $A \subseteq \mathcal{A}_i$,
2. $\forall q \in (Q_i \setminus \mathcal{A}_i) . \delta(q) \subseteq (Q \setminus \mathcal{A}_i)$ and
3. $\forall q \in (Q_{\neg i} \setminus \mathcal{A}_i) . \delta(q) \not\subseteq \mathcal{A}_i$.

The computation of $\mathrm{attr}_i(G, A)$ is a least fixpoint computation, in which $\mathcal{A}_i$ is initialized to A and the set of attraction transitions Δ_i is initially empty. Vertices of G in Q_i (controlled by Player i) are added if they have at least one transition into a vertex already acquired to $\mathrm{attr}_i(G, A)$, while vertices of G in Q_j are added if all their transitions lead to vertices already acquired to $\mathrm{attr}_i(G, A)$. When a vertex in Q_i is added to $\mathcal{A}_i$, the transitions from it to vertices already in $\mathrm{attr}_i(G, A)$ are added to Δ_i.

At Line 6, the algorithm recurs on the subgame induced by the removal of states in $\mathcal{A}_i$ from the game G.

Definition 6 A turn-based game $G^S = (Q^S, Q_0^S, Q_1^S, \delta^S, \Pi^S)$ is the subgame of turn-based game $G = (Q, Q_0, Q_1, \delta, \Pi)$ induced by $Q^S \subseteq Q$ if

1. $Q_i^S = Q_i \cap Q^S$,

2. $\forall u, v \in Q^S \,.\, v \in \delta^S(u) \;\leftrightarrow\; v \in \delta(u)$ and
3. Π^S is the restriction of Π to Q^S.

If U_j returned by the recursive call is empty, then Player i wins from all vertices of the current G whenever the play stays in G. In this case, T_i is obtained as the union of three sets: The set Δ_i produced by the attraction computation at Line 5, the set of transitions for Player i returned by the recursive call at Line 6, and the transitions from vertices in $A \cap Q_i$.

If U_j is not empty at Line 7, then Player j wins from every vertex in U_j using the transitions computed in the recursive call at Line 6. Moreover, Player j wins from every vertex in $\mathcal{A}_j$ from which she can force a visit of U_j. To force such a visit, she uses the transitions in Δ_j. (Note that if any vertex is added to $\mathcal{A}_j$ by the attraction computation, at least one such vertex is from A.) The second recursive call (Line 12) returns winning positions and transitions for the residual subgame. For Player i these are the only winning positions and transitions in G. For Player j, the resulting positions and transitions are added to those computed at Lines 6 and 12.

Figure 2 shows a state-based parity game with two colors. Squares denote vertices in Q_1 and circles denote vertices in Q_0. Zielonka's algorithm returns all vertices of the graph in U_1. The bold arrows denote transitions in T_1. Since both transitions out of vertices d and g are in T_1, there are four distinct memoryless winning strategies for Player 1.

Note that transitions (e, b) and (f, c) are not in T_1. A strategy that used both of them would have to use memory to prevent the play from cycling through b, f, c, and e without visiting a. Exclusion of (e, b) and (f, c) from T_1 prevents such cycles, but also rules out eight memoryless strategies that include one but not the other. The computation of attractions imposes a preorder on the vertices in the fixpoint: $v \preceq u$ iff v is added to $\mathrm{attr}_i(G, A)$ no later than u. (It is a preorder, because breadth-first computation of the attraction may add multiple vertices simultaneously.) A transition (u, v) between two vertices of $\mathrm{attr}_i(G, A)$ is in T_1 iff $v \prec u$.

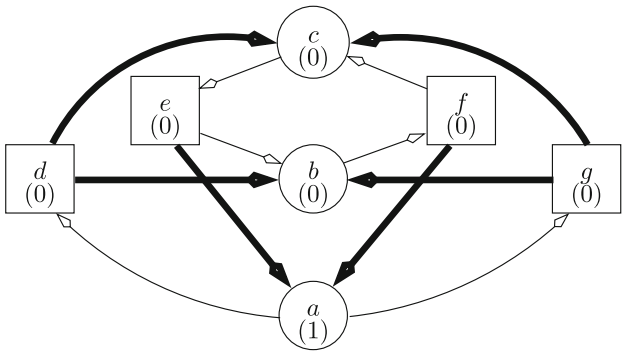

Fig. 2 A parity game. Vertex colors are shown in *parentheses*

Game solving algorithms based on attraction computations—not just Zielonkas's—therefore only compute a subset of all memoryless strategies. Even in algorithms not explicitly based on attraction computations like the one of [20], strategy computation relies on the order in which vertices are added to the set of winning positions to select transitions. Even though we have used Zielonka's algorithm for the purpose of illustration, this observation applies more generally.

3 Solving games incrementally

A player wins a game with a conjunctive generalized parity winning condition iff it has a strategy that simultaneously works for each parity condition. If one could compute all winning strategies for one parity condition, one would then be able to play the conjunctive game incrementally by initializing the set of candidate strategies to all possible strategies and then successively removing all strategies that are not winning for each parity condition. The surviving strategies would be winning for the conjunctive game.

An incremental approach is particularly attractive when symbolic algorithms are used to solve the games, because it allows the parity game to be restructured between steps. However, the simple scheme outlined above does not work because algorithms for parity games only compute a subset of the winning strategies. For instance, algorithms for parity games only compute a subset of the memoryless winning strategies, as shown in the previous section. In this section, we show that the subset of strategies computed is not sufficient to solve conjunctive generalized parity games in an iterative fashion. We prove, however, that it is possible to decompose the solution in stages when some of the parity conditions correspond to safety or persistence properties.

Consider the parity game shown in Fig. 3 with the conjunctive winning condition $\{\pi_1, \pi_2\}$, where π_1 assigns color 1 to b and color 0 to the other vertices, while π_2 assigns color 1 to c and color 0 to the other vertices. Player 1 moves from vertex a. (That is, $Q_0 = \{b, c\}$ and $Q_1 = \{a\}$.) Note that the parity conditions are effectively Büchi conditions corresponding to the LTL formula $(\mathsf{G}\,\mathsf{F}\,b) \wedge (\mathsf{G}\,\mathsf{F}\,c)$.

Suppose the algorithm of Fig. 1 is used to compute the winning positions according to π_1. For the graph G in Fig. 3, the attractor of $\{b\}$ for Player 1 is computed as follows at Line 5. Initially b is in the attractor and the set of transitions is empty; a is then added to $\mathrm{attr}_1(G, \{b\})$ and the transition

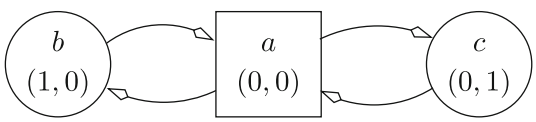

Fig. 3 A simple generalized parity game

from a to b is added to Δ_1. Finally c is added to the attractor and the transition from c to a is added to the set of transitions. Since $G \backslash \mathcal{A}_1$ is empty, the recursive call returns immediately with $U_0 = \emptyset$. Therefore all positions are winning for Player 1. The transition from b to a is added to Δ_1 to produce T_1. It is clear that there is no winning strategy for π_2 when only transitions from T_1 are allowed from vertices in Q_1, though there exist strategies to win both π_1 and π_2 on G. One such strategy uses one bit of memory to alternate between the two transitions out of a.

Though in general one cannot solve generalized parity games by incrementally applying Zielonka's algorithm—one must resort instead to algorithms that can deal with multiple winning conditions like the ones of [8]—there are cases of practical relevance when it is possible to decompose the computation, namely when some of the winning conditions correspond to *safety* or *persistence* properties [32].

Definition 7 A *safety winning condition* for a parity game $G = (Q, Q_0, Q_1, \delta, \pi)$ is a function $\pi : Q \to \{0, 1\}$ such that for every state $u \in Q$ such that $\pi(u) = 0$ there is no state $v \in \delta(u)$ such that $\pi(v) = 1$.

The winning plays according to such a condition form a closed set of the product topology of Σ^ω recognized by a finite automaton. Hence they form an ω-regular safety property. Closed sets form the F_1 class in the Borel hierarchy [26]. Conversely, an ω-regular safety property is accepted by a DPW with a safety winning condition [2].

While most translators guarantee that a syntactically safe property [41] expressed in LTL is translated into a DPW with a safety acceptance condition (a safety DPW), pathological safety properties [23] may not be recognized as such. This, however, is not a significant drawback, because such properties are quite rare in practice.[3]

Definition 8 A *persistence winning condition* for a parity game $G = (Q, Q_0, Q_1, \delta, \pi)$ is a function $\pi : Q \to \{1, 2\}$.

The winning plays according to a persistence winning condition form a regular F_2 set in the Borel hierarchy. Many commonly-used properties are recognized by weak automata. A weak automaton can be interpreted as a Büchi automaton such that every strongly connected component of the state graph is either included in the accepting states F or has null intersection with it [34]. It is known that non-deterministic weak word automata (NWW) and deterministic co-Büchi word automata (DCW) are equally expressive [29]. Therefore, an NWW translates into a DPW with a persistence acceptance condition.

Definition 9 For game $G = (Q, Q_0, Q_1, \delta, \alpha)$, $G[\alpha \leftarrow \alpha']$ denotes the game $(Q, Q_0, Q_1, \delta, \alpha')$.

[3] Even mildly pathological cases like $(p \,\mathsf{U}\, q) \vee \mathsf{G}\, p$ are usually recognized as safety properties by translators.

Definition 10 Given game $G = (Q, Q_0, Q_1, \delta, \alpha)$, α' is equivalent to α with respect to G if a play is winning in G iff it is winning in $G[\alpha \leftarrow \alpha']$.

Since $F_1 \subset F_2$, we could limit our discussion to persistence winning conditions. If a winning condition satisfies Definition 7 then we can convert it to a winning condition that satisfies Definition 8 as described in the following lemma.

Lemma 5 *Let $\Pi = \{\pi_1, \ldots, \pi_k\} (k > 1)$ be a conjunctive winning condition for a parity game $G = (Q, Q_0, Q_1, \delta, \Pi)$. Suppose that π_k is a winning condition such that no cycle in G contains both states of even color and states of odd color; then there exists a persistence winning condition π'_k such that $\Pi' = \{\pi_1, \ldots, \pi'_k\}$ is equivalent to Π with respect to G.*

Proof The lack of cycles of mixed parity in G implies that π_k is equivalent (with respect to G) to $\pi'_k : Q \to \{1, 2\}$ such that $\pi'_k(q) = 2$ iff $\pi_k(q)$ is even. Therefore, Π' is equivalent to Π. □

Note that all properties in $F_2 \cap G_2$ of the Borel hierarchy (the *obligation* properties, a superset of the safety properties) are accepted by DPWs of minimum index that satisfy the condition of Lemma 5 and use colors 0 and 1. For all of them, it is easy to "upgrade" their DPWs to persistence acceptance conditions. The DPW of minimum index computed by the procedure of [9] for a property in $F_2 \backslash G_2$, on the other hand, may use color 0 beside 1 and 2. If, however, all states of a maximal strongly connected component (SCC) have color 0, then they can all be colored 2; otherwise, if all the cycles in that SCC involve at least one state of color 1 or 2, then its 0 states can be colored 1. If states colored 0 remain, then the property accepted by the automaton is not in F_2. Therefore, every DPW for a persistence property produced by the procedure of [9] can be equipped with a persistence acceptance condition.

In spite of Lemma 5, practical reasons that will become apparent suggest that we distinguish the case of safety conditions from the more general one of persistence conditions. It is immediate that a set $\{\pi_1, \ldots, \pi_k\}$ of safety conditions can be replaced by a single safety condition π such that $\pi(q) = \prod_{1 \le i \le k} \pi_i(q)$. The details of how this observation is put to use are in the algorithm of Fig. 4.

A conjunctive parity winning condition can be converted to an equivalent one thanks to the following observation.

Theorem 3 *Let $\Pi = \{\pi_1, \ldots, \pi_k\}$ $(k > 1)$ be a conjunctive winning condition for parity game $G = (\Sigma, Q, D, \delta, \Pi)$. Suppose that the largest odd color in the co-domain of π_k is m and the largest odd color in the co-domain of π_{k-1} is l. Then Π is equivalent to $\Pi' = \{\pi_1, \pi_2, \ldots, \pi'_{k-1}, \pi_k\}$ with respect to G, where, for each $q \in Q$,*

$$\pi'_{k-1}(q) = \begin{cases} l+1 & \text{if } \pi_k(q) = m+1 \\ \pi_{k-1}(q) & \text{otherwise.} \end{cases}$$

1 **algorithm** PERSISTENCE-FIRST($\{G^1, G^2, \ldots, G^k\}$)
2 **foreach** $i = 1, 2, \ldots, k$
2.1 **if** $i = 1$ **then** $G \leftarrow G^0 \times G^1$
else $G \leftarrow G \times G^i$
2.2 let π be the last component of Π
2.3 $(U, U_1, T, T_1) \leftarrow$ ZIELONKA($G[\Pi \leftarrow \pi]$)
2.4 **if** $i = 1$ **then** $U \leftarrow U_0$ and $T \leftarrow T_0$
else $U \leftarrow U \cup U_0$ and $T \leftarrow T \cup T_0$
2.5 $G \leftarrow$ SIMPLIFY(G, U_1)
end
3 **if** $|\Pi| = 0$ **then** $(U_0, U_1, T_0, T_1) \leftarrow (\emptyset, Q, \emptyset, \delta)$
else if $|\Pi| = 1$ **then** $(U_0, U_1, T_0, T_1) \leftarrow$ ZIELONKA(G)
else $(U_0, U_1, T_0, S) \leftarrow$ CONJPARITYWIN(G)
4 $U \leftarrow U \cup U_0$ and $T \leftarrow T \cup T_0$
5 **if** $S =$ NULL **then** $S \leftarrow \{\emptyset\} \times T_1$
6 **return** (U, U_1, T, S)

Fig. 4 An incremental algorithm for conjunctive parity games. $S : M \times (U_1 \cap Q_1) \rightarrow (U_1 \cap Q_0)$ is a strategy, where M is the finite memory. G^0 is the identity game such that $G^0 \times G^i = G^i$

Proof Let A^k_{m+1} be the states of G with color $m+1$ with respect to π_k. Suppose a play ρ is winning according to π_k and π_{k-1} then

$$\inf(\rho) \cap A^k_{m+1} = \emptyset \text{ and } \inf(\rho) \cap A^{k-1}_{l+1} = \emptyset.$$

From definition of π'_{k-1} we have

$$\forall q \in \inf(\rho) \,.\, \pi_{k-1}(q) = \pi'_{k-1}(q)$$

implying that

$$\max(\pi_{k-1}(\inf(\rho))) = \max(\pi'_{k-1}(\inf(\rho))).$$

Hence $\max(\pi'_{k-1}(\inf(\rho)))$ is odd. Therefore ρ is winning according to π'_{k-1} (and π_k).

Conversely, a play ρ' winning according to π'_{k-1} and π_k is such that

$$\inf(\rho') \cap A'^{k-1}_{l+1} = \emptyset \text{ and } \inf(\rho') \cap A^k_{m+1} = \emptyset.$$

From definition of π'_{k-1} we have

$$\forall q \in \inf(\rho') \,.\, \pi'_{k-1}(q) = \pi_{k-1}(q).$$

Hence $\max(\pi_{k-1}(\inf(\rho')))$ is odd. Therefore ρ' is winning according to π_{k-1} (and π_k). □

Note that for any state q that satisfies $\mathsf{F}\, A^k_{m+1}$, $\pi_k(q)$ can be set to $m+1$.

Corollary 1 *Let $\Pi = \{\pi_1, \ldots, \pi_k\}$ $(k > 1)$ be a conjunctive winning condition for parity game $G = (\Sigma, Q, D, \delta, \Pi)$. Suppose that π_k is a persistence winning condition and that the largest odd color in the co-domain of π_{k-1} is m. Then Π is equivalent to $\Pi' = \{\pi_1, \ldots, \pi'_{k-1}\}$ with respect to G, where, for each $q \in Q$,*

$$\pi'_{k-1}(q) = \begin{cases} m+1 & \text{if } \pi_k(q) \text{ is } 2 \\ \pi_{k-1}(q) & \text{otherwise.} \end{cases}$$

Proof Let A^k_2 be the states of G with color 2 with respect to π_k. It follows from Theorem 3 that a play ρ winning according to π_k and π_{k-1} is winning according to π'_{k-1}.

Conversely, suppose ρ' is winning according to π'_{k-1} then

$$\inf(\rho') \cap A'^{k-1}_{m+1} = \emptyset.$$

Since

$$A^k_2 \cup A^{k-1}_{m+1} = A'^{k-1}_{m+1}$$

we have

$$\inf(\rho') \cap A^k_2 = \emptyset.$$

Thus we have ρ' is winning according to π_k. Therefore Π' is equivalent to $\{\pi_1, \pi_2, \ldots, \pi'_{k-1}, \pi_k\}$, which, by Theorem 3, is equivalent to Π. □

As an example, consider the LTL games defined by a graph Γ, which encodes permissible moves, and an objective $\varphi \wedge \psi$, where φ is one of

$$\varphi_1 = \mathsf{G}(\texttt{favorable} \rightarrow \mathsf{F}\,\texttt{win})$$
$$\varphi_2 = \mathsf{G}(\texttt{favorable} \rightarrow \mathsf{G}\,\neg\texttt{lose})$$
$$\varphi_3 = \texttt{favorable} \rightarrow (\texttt{favorable}\,\mathsf{U}\,\texttt{win}),$$

and ψ is either

$$\psi_1 = \mathsf{F}(\texttt{win} \vee \texttt{lose})$$
$$\text{or} \quad \psi_2 = \mathsf{F}\,\mathsf{G}(\texttt{win} \vee \texttt{lose}).$$

In these LTL formulae, `favorable`, `win`, and `lose` are atomic propositions labeling the states of Γ. The initial translation of $\varphi \wedge \psi$ determines the NBWs for φ and ψ and composes them to produce a DPW with conjunctive acceptance condition $\Pi = \{\pi_\varphi, \pi_\psi\}$. Note that φ_1 is a recurrence (G_2) property, φ_2 is a safety (F_1) property, φ_3 is an obligation $(F_2 \cap G_2)$ property, ψ_1 is a guarantee (G_1) property, and ψ_2 is a persistence (F_2) property. In all six games, Π can be reduced to an equivalent non-generalized acceptance condition with at most three colors.

Repeated application of Lemma 5 and Corollary 1 eliminates all safety and persistence winning conditions (except one if there are no other winning conditions) with a maximum increase of one color in one of the surviving winning conditions (if all remaining winning conditions have an odd color as maximum priority). Repeated application of Theorem 3 may further reduce the number of used colors (hence possibly the index) of one or more parity winning conditions. The "classical" algorithm of [8] runs in

$$O(m \cdot n^{2d}) \cdot \binom{d}{d_1, d_2, \ldots, d_k}, \tag{1}$$

where $n = |Q|$, $m = |E|$, $\pi_i : Q \rightarrow [d_i + 1]$ is the ith component of the winning condition, k is the number of components, and $d = \sum_{1 \le i \le k} d_i$. The simplification of the conjunctive condition afforded by Corollary 3 and Theorem 1 improves the bound by reducing k and, in most cases, also d.

Our objective is to find winning strategies for the conjunctive player (Player 1). A state which is losing with respect to any parity winning condition $\pi \in \Pi$ is also losing with respect to the conjunctive parity winning condition Π. This suggests a way to play conjunctive generalized parity games iteratively. After eliminating all the safety and persistence winning conditions we can play each parity winning condition individually and for each of them, restrict G to the winning states, until Player 1 can win each parity winning condition individually in the pruned game G. Only at this point all conditions are considered simultaneously.

The iterative approach just outlined operates on a turn-based game G. In practice, G is built through the composition of several input-based non-generalized parity games, each coming from the translation of a property into a DPW.

Definition 11 Given two input-based games $G^1 = (\Sigma, Q^1, D^1, \delta^1, \alpha^1)$ and $G^2 = (\Sigma, Q^2, D^2, \delta^2, \alpha^2)$ then $G = (\Sigma, Q, \delta, \alpha)$ is the composition of G^1 and G^2, written $G^1 \times G^2$, if

$$Q = Q^1 \times Q^2,$$
$$D = \{((q^1, q^2), \sigma) | (q^1, \sigma) \in D^1 \wedge (q^2, \sigma) \in D^2\},$$
$$\delta((u^1, u^2,), \sigma) = (\delta^1(u^1, \sigma), \delta^2(u^2, \sigma)).$$

For $i \in \{1, 2\}$, let $\gamma_i : Q_1 \times Q_2 \to Q_i$ map a state of the composed game to its ith component:

$$\gamma_i((u^1, u^2)) = u^i \text{ for } i \in \{1, 2\}.$$

For a play ρ, we write $\gamma_i(\rho)$ for $\gamma_i(\rho_0), \gamma_i(\rho_1), \ldots$ for $i \in \{1, 2\}$. A play $\rho \in \alpha$ iff

$$\forall i \in \{1, 2\} \,.\, \gamma_i(\rho) \in \alpha^i.$$

If G^1 and G^2 are games with generalized parity winning conditions $\Pi^1 = \{\pi_1^1, \pi_1^1, \ldots, \pi_{n_1}^1\}$ and $\Pi^2 = \{\pi_1^2, \pi_2^2, \ldots, \pi_{n_2}^2\}$, respectively, then α can be written as $\Pi = \{\pi_1, \pi_2, \ldots, \pi_{n_1+n_2}\}$, where

$$\pi_k((q^1, q^2)) = \begin{cases} \pi_k^1(q^1) & 1 \le k \le n_1 \\ \pi_{k-n_1}^2(q^2) & n_1 < k \le n_1 + n_2. \end{cases}$$

Figure 4 shows the application of Theorem 3 and Corollary 1 during the construction of G. If π from line 2.2 is a safety or persistence winning condition then the SIMPLIFY procedure removes π from Π. In case of a persistence winning condition Π is modified as described in Corollary 1. If π was neither a safety nor a persistence winning condition then modifications to Π are made if Theorem 3 is applicable. When G is restricted to the set of winning states with respect to π, some of the winning states become unreachable. The unreachable states are removed and the losing states are replaced with one representative losing state. This often improves the symbolic encoding of G. The iterative approach discussed earlier is still applicable inside CONJPARITYWIN.

The procedure of Fig. 4 partitions Q into the set of winning and losing states. Player 0 can win from any state in U by following any positional strategy in T. Player 1 can win from any state in U_1 by following any memoried strategy in S. We can use line 2.3 because a losing state with respect to $\pi \in \Pi$ is losing with respect to Π. The SIMPLIFY procedure modifies Π according to Theorem 3 and Corollary 1. G is restricted to winning reachable states because no winning strategy can visit unreachable states or losing states.

Since safety properties often form the bulk of a specification, the ability to treat them incrementally is significant. Safety properties are a special case of persistence properties as no modification of other components of the winning condition is necessary. This special case was already discussed in [6].

The complexity of "classical" algorithm of [8] is given in (1). If π_k is a safety condition, solving the game in two stages leads to a better bound for the second stage, $O(m \cdot n^{2d-2}) \cdot \binom{d-1}{d_1, \ldots, d_{k-1}}$, while the first stage runs in $O(m \cdot n^2)$. In practice, in the second stage, the number of transitions may decrease, and the removal of losing positions for π_1 may reduce the number of colors in the remaining conditions. This may further speed up execution. Similar considerations apply when π_k is a persistence condition.

The separation of safety and liveness specifications is part of the approach of [36], which synthesizes from reactive(1) specifications that are made up of assumptions and guarantees. The system has to satisfy the guarantees when the environment satisfies its assumptions. The specification has the form

$$\left(I_e \wedge S_e \wedge \bigwedge_i \mathsf{GF}\, L_e^i\right) \to \left(I_s \wedge S_s \wedge \bigwedge_i \mathsf{GF}\, L_s^i\right), \quad (2)$$

where $I_e(I_s)$ are constraints on initial states of environment (system), similarly $S_e(S_s)$ and $L_e(S_s)$ are safety and liveness constraints. The specification is translated from (2) to the following form

$$(I_e \wedge S_e \to I_s \wedge S_s) \wedge \left(\bigwedge_i \mathsf{GF}\, L_e^i \to \bigwedge_i \mathsf{GF}\, L_s^i\right). \quad (3)$$

The realizability of (3) is checked by solving a GR(1) game.

In [21], the translation from (2) to (3) is proven to be correct only when the environment is *well-separated* from the system. A well-separated environment has the ability to satisfy the assumptions without any cooperation from the system. In [21], it is proven that the specification in the form of (2) can be rewritten as follows:

$$(I_e \to I_s) \wedge (S_s \,\mathsf{W}\, \neg S_e) \wedge \left(I_e \wedge S_e \wedge \bigwedge_i \mathsf{GF}\, L_e^i \to \bigwedge_i \mathsf{GF}\, L_s^i\right), \quad (4)$$

where W is the weak until operator. The specifications in the forms of (2) or (4) are equi-realizable even when the environment is not well-separated.

While our approach deals with unrestricted LTL, it benefits significantly if the specification is in the form of (3) or (4). Each representative set I_e, S_e, I_s and S_s is a conjunction of safety properties. For (3), the incremental approach starts by considering each property in $I_e \wedge S_e$ by reversing the roles of the system and environment (system becomes Player 0 and environment becomes Player 1). This allows us to restrict the game to the states where the environment satisfies its assumptions. The states where the environment can not satisfy the assumptions are winning for the system because the system has a winning strategy or in the case of a well-separated environment any strategy for the system is a winning strategy. In the next stage, the incremental approach considers each property in $I_s \wedge S_s$ followed by the reactive(1) property $\bigwedge_i \mathsf{GF}\, L_e^i \rightarrow \bigwedge_i \mathsf{GF}\, L_s^i$.

For (4), the incremental approach considers each property in I_e first by reversing the roles of the two players, then each property in I_s is considered, followed by $(S_s\, \mathsf{W} \neg S_e)$ which is a safety property. Finally, the reactive(1) property $I_e \wedge S_e \wedge \bigwedge_i \mathsf{GF}\, L_e^i \rightarrow \bigwedge_i \mathsf{GF}\, L_s^i$ is solved.

4 Implementation

4.1 Solving input-based games

As discussed in Sect. 2.2, a specification is naturally translated into an input-based game $G = (\Sigma, Q, D, \delta, \Pi)$ with $D = Q \times \Sigma$. The input-based G can be transformed into a turn-based game $\widehat{G} = (\Sigma, \widehat{Q}, \widehat{Q}_0, \widehat{Q}_1, \widehat{\delta}, \widehat{\Pi})$, for instance using Definition 3. The turn-based game has significantly larger state space because the size of state space is $O(|Q \times \Sigma|)$. When deploying symbolic algorithms for playing games, more variables are required to encode the states. It can be observed from Definition 3 that the base states that appear in a sequence are sufficient to decide if the sequence is winning. This suggests that we collapse the four transitions from a base state to another base state into one transition so as to play the associate game $\widehat{G}$ without explicitly creating it. In this section, we discuss the details of this process in the context of symbolic graph algorithms.

For a symbolic implementation, the number of variables appearing in the characteristic functions is important; among other things, the search for a good BDD variable order is affected and this has a negative impact on the performance of the synthesis process. In our approach we try to avoid promoting all the input variables to state variables as in [11,12,36].

We adapt both Zielonka's algorithm for non-generalized parity winning conditions and its extension to conjunctive parity winning conditions [8] to play directly the input-based games. The *input-based Zielonka* (ib-Z) algorithm requires appropriate definition of i-attraction of a set and of subgame when an i-attraction is removed from an input-based game. We prove the correctness of our adaptation by relating the run of ib-Z on the input-based game G to the run of Zielonka's algorithm on the associate game $\widehat{G}$.

The key observation is that tracking the base states of $\widehat{G}$ is sufficient if the input choices of each player are properly constrained. These constraints hold in the original game and the subgames that ib-Z recursively examines. (The proper notion of a subgame of input-based games is established in Definition 13.) We build up to the main result by proving that the i-attractions computed in G and $\widehat{G}$ agree on the base states and that the subgames encountered while solving the associate game $\widehat{G}$ are related to the associates of the subgames encountered while solving G.

The computation of i-attractions in input-based games relies on the *pre-image* operator EX. We define the EX operator so that it quantifies target states from the transition relation, but does not quantify the inputs σ. For input-based game $G = (\Sigma, Q, D, \delta, \Pi)$ the pre-image of $T \subseteq Q$ is defined by

$$\mathsf{EX}_G\, T = \{(u, \sigma) \in D \mid \delta(u, \sigma) \in T\}.$$

When the context is clear, we write EX instead of EX_G.

Definition 12 Given an input-based game $G = (\Sigma, Q, D, \delta, \Pi)$ and the set of states $T \subseteq Q$, let

$$\begin{aligned}
\mathsf{MX}_0\, T = \{q \in Q \mid \exists \sigma_{0d}.(\exists \sigma_1.\exists \sigma_{0p}.(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \in D) \\
\wedge \big(\forall \sigma_1.(\exists \sigma_{0p}.(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \in \mathsf{EX}\, T) \\
\vee (\forall \sigma_{0p}.(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \notin D)\big)\}, \\
\mathsf{MX}_1\, T = \{q \in Q \mid \forall \sigma_{0d}.(\forall \sigma_1.\forall \sigma_{0p}.(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \notin D) \\
\vee \big(\exists \sigma_1.(\forall \sigma_{0p}.(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \notin (D \setminus \mathsf{EX}\, T)) \\
\wedge (\exists \sigma_{0p}.(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \in D)\big)\}.
\end{aligned}$$

The set $T_i \subseteq Q$ is the i-attraction of T in the input-based game G (written $\mathrm{attr}_i(G, T)$ as for turn-based games) if it is the least set Z such that

1. $T \subseteq Z$ and
2. $(Q \setminus Z) \cap \mathsf{MX}_i\, Z = \emptyset$.

Lemma 6 *Let $\widehat{G} = (\widehat{Q}, \widehat{Q}_0, \widehat{Q}_1, \widehat{\delta}, \widehat{\Pi})$ be the associate game of the input-based game $G = (\Sigma, Q, D, \delta, \Pi)$. Let $T \subseteq Q$ be the set of base states and let $\widehat{T} = T$. Let $\mathcal{T}_i$ be the states of G that can be forced by Player i into T in one or less steps and $\widehat{\mathcal{T}}_i$ be the states of $\widehat{G}$ that can be forced by Player i into $\widehat{T}$ in four or less steps then $\mathcal{T}_i = Q \cap \widehat{\mathcal{T}}_i$.*

Proof Let $\widehat{\mathsf{MX}}_i\, S$ be the set of states that Player i can control to S in one step. Since $\widehat{G}$ is four-partite, whenever T is a set of base states we have the following:

$$\widehat{\mathsf{MX}}_i^0 T = T,$$
$$\widehat{\mathsf{MX}}_i^1 T = \{(q, \sigma_{0d}, \sigma_1, \sigma_{0p}) \in \widehat{Q} \mid \delta((q, \sigma_{0d}, \sigma_1, \sigma_{0p})) \in T\},$$
$$\widehat{\mathsf{MX}}_0^2 T = \{(q, \sigma_{0d}, \sigma_1) \in \widehat{Q} \mid \exists \sigma_{0p} . (q, \sigma_{0d}, \sigma_1, \sigma_{0p}) \in \widehat{\mathsf{MX}}_0^1 T\},$$
$$\widehat{\mathsf{MX}}_0^3 T = \{(q, \sigma_{0d}) \in \widehat{Q} \mid \forall \sigma_1 . (q, \sigma_{0d}, \sigma_1) \notin (\widehat{Q} \backslash \widehat{\mathsf{MX}}_0^2 T)\},$$
$$\widehat{\mathsf{MX}}_0^4 T = \{q \in Q \mid \exists \sigma_{0d} . (q, \sigma_{0d}) \in \widehat{\mathsf{MX}}_0^3 T\},$$
$$\widehat{\mathsf{MX}}_1^2 T = \{(q, \sigma_{0d}, \sigma_1) \in \widehat{Q} \mid \forall \sigma_{0p} . (q, \sigma_{0d}, \sigma_1, \sigma_{0p}) \notin (\widehat{Q} \backslash \widehat{\mathsf{MX}}_1^1 T)\},$$
$$\widehat{\mathsf{MX}}_1^3 T = \{(q, \sigma_{0d}) \in \widehat{Q} \mid \exists \sigma_1 . (q, \sigma_{0d}, \sigma_1) \in \widehat{\mathsf{MX}}_1^2\},$$
$$\widehat{\mathsf{MX}}_1^4 T = \{q \in Q \mid \forall \sigma_{0d} . (q, \sigma_{0d}) \notin (\widehat{Q} \backslash \widehat{\mathsf{MX}}_1^3 T)\}.$$

From Definition 3 we have

$$\forall q . \forall \sigma_{0d} . \forall \sigma_1 . \forall \sigma_{0p} . (q, \sigma_{0d}, \sigma_1, \sigma_{0p}) \in \widehat{Q} \leftrightarrow (q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \in D. \tag{5}$$

Then it follows from the definitions of $\mathsf{EX}\, T$ in G and $\widehat{\mathsf{MX}}_i T$ in $\widehat{G}$ that:

$$\forall (q, \sigma_{0d}, \sigma_1, \sigma_{0p}) \in \widehat{Q} .$$
$$(q, \sigma_{0d}, \sigma_1, \sigma_{0p}) \in \widehat{\mathsf{MX}}_i^1 T \leftrightarrow (q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \in \mathsf{EX}\, T.$$

We can therefore rewrite $\widehat{\mathsf{MX}}_1^4 T$ as follows:

$$\widehat{\mathsf{MX}}_1^4 T = \{q \in Q \mid \forall \sigma_{0d} . (\forall \sigma_1 . \forall \sigma_{0p} . (q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \notin D) \vee \big(\exists \sigma_1 (\forall \sigma_{0p} . ((q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \notin (D \backslash \mathsf{EX}\, T))) \wedge (\exists \sigma_{0p} . (q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \in D)\big)\},$$

which leads to the conclusion that $\mathsf{MX}_1\, T = \widehat{\mathsf{MX}}_1^4 T$. Since

$$\widehat{\mathcal{T}}_i = \widehat{\mathsf{MX}}_1^{0-4} T, \text{ where}$$
$$\widehat{\mathsf{MX}}_1^{i-j} T = \widehat{\mathsf{MX}}_1^i T \cup \cdots \cup \widehat{\mathsf{MX}}_1^j T,$$

and only $\widehat{\mathsf{MX}}_1^0 T$ and $\widehat{\mathsf{MX}}_1^4 T$ contain base states, we conclude that $\mathcal{T}_1 = Q \cap \widehat{\mathcal{T}}_1$. Similarly, we can show that $\mathcal{T}_0 = Q \cap \widehat{\mathcal{T}}_0$. Therefore $\mathcal{T}_i = Q \cap \widehat{\mathcal{T}}_i$. □

Corollary 2 *Let $\widehat{G} = (\widehat{Q}, \widehat{Q}_0, \widehat{Q}_1, \widehat{\delta}, \widehat{\Pi})$ be the associate of input-based game $G = (\Sigma, Q, D, \delta, \Pi)$. Let $T \subseteq Q$ be a set of base states. Then*

$$\bigcup_{n \geq 0} h^n(\emptyset) = \bigcup_{n \geq 0} g^n(\emptyset), \quad \textit{where}$$
$$h(X) = T \vee \mathsf{MX}_i\, X \quad \textit{and}$$
$$g(X) = T \vee \widehat{\mathsf{MX}}_i^4 X.$$

Proof By induction on n. Since $h^0(\emptyset) = g^0(\emptyset) = \emptyset$, the base case holds. Suppose $g^{n-1}(\emptyset) = h^{n-1}(\emptyset)$; thanks to Lemma 6 we have $h^n(\emptyset) = g^n(\emptyset)$. □

Let $\widehat{S} \subseteq \widehat{Q}$ be a set of states of $\widehat{G}$ and let $\overline{S} \subseteq \widehat{S}$ be the set of non-base states such that

$$\overline{S} = \{q \in \widehat{S} \mid \gamma(q) \notin \widehat{S}\}.$$

Then $\widehat{S}$ is an *i-based* set if

$$\forall q \in \overline{S} . q \in \widehat{\mathsf{MX}}_i^{1-3}(\widehat{S} \backslash \overline{S}).$$

Let

$$\tilde{S} = \{q \in \widehat{S} \backslash \overline{S} \mid q \neq \gamma(q)\}.$$

Then $\widehat{S} = \overline{S} \cup (\widehat{S} \cap Q) \cup \tilde{S}$. Note that an i-closed set (one such that Player i can prevent the play from escaping it) is also i-based.

Lemma 7 *Let $\widehat{G} = (\widehat{Q}, \widehat{Q}_0, \widehat{Q}_1, \widehat{\delta}, \widehat{\Pi})$ be the associate of input-based game $G = (\Sigma, Q, D, \delta, \Pi)$. Let $\widehat{S} \subseteq \widehat{Q}$ be a set of states in $\widehat{G}$. Then $\gamma(q) \notin \gamma(\widehat{S})$ implies that if q can be attracted to $\widehat{S}$ by Player i then q can also be attracted to $\gamma(\widehat{S})$ by Player i.*

Proof From Definition 3 it follows that if there is a path between states q and q' in $\widehat{Q}$ such that $\gamma(q) \neq \gamma(q')$, then that path must visit $\gamma(q')$. Therefore q can only be forced to visit any state in $\widehat{S}$ after visiting $\gamma(\widehat{S})$. Therefore q is attracted to $\gamma(\widehat{S})$. □

Lemma 8 *Let $\widehat{G} = (\widehat{Q}, \widehat{Q}_0, \widehat{Q}_1, \widehat{\delta}, \widehat{\Pi})$ be the associate of input-based game $G = (\Sigma, Q, D, \delta, \Pi)$. Let $\widehat{T} \subseteq \widehat{Q}$ be an i-based set and $\widehat{P} = Q \cap \widehat{T}$. Let $\widehat{T}_i$ be $\mathrm{attr}_i(\widehat{G}, \widehat{T})$ and $\widehat{P}_i$ be $\mathrm{attr}_i(\widehat{G}, \widehat{P})$. Then $Q \cap \widehat{P}_i = Q \cap \widehat{T}_i$.*

Proof Consider $q \in Q$. If $q \in \widehat{T}$, then q trivially belongs to both attractions. If, on the other hand, $q \notin \widehat{T}$, it must be $\gamma(q) \notin \gamma(\widehat{T})$. Suppose $q \in \widehat{T}_i$. Then Player i can force a visit of $\widehat{T}$ from q. By definition of i-based set, Player i can force a visit of $\widehat{T} \backslash \overline{T}$ from every state of $\overline{T}$. Therefore, Player i can force a visit of $\widehat{T} \backslash \overline{T}$ from q. Since $\gamma(\widehat{T} \backslash \overline{T}) = \widehat{T} \cap Q$, Lemma 7 implies that $Q \cap \widehat{T}_i \subseteq Q \cap \widehat{P}_i$. The inclusion in the opposite direction follows from the monotonicity of attractions. □

We now establish the correspondence between i-attractions in G and $\widehat{G}$.

Lemma 9 *Let $\widehat{G} = (\widehat{Q}, \widehat{Q}_0, \widehat{Q}_1, \widehat{\delta}, \widehat{\Pi})$ be the associate of input-based game $G = (\Sigma, Q, D, \delta, \Pi)$. Let $\widehat{T} \subseteq \widehat{Q}$ be an i-based set and $T = Q \cap \widehat{T}$. Then*

$$T_i = Q \cap \widehat{T}_i,$$

where $\widehat{T}_i = \mathrm{attr}_i(\widehat{G}, \widehat{T})$ and $T_i = \mathrm{attr}_i(G, T)$.

Proof Let $f(X) = \widehat{T} \vee \widehat{\mathsf{MX}}_i X$, $\phi(X) = T \vee \widehat{\mathsf{MX}}_i X$, and $g(X) = T \vee \widehat{\mathsf{MX}}_i^4 X$. These functions are monotonic over

finite lattices; therefore they are continuous. Thanks to Corollary 2, we have the following:

$$\widehat{T}_i = \bigcup_{n\geq 0} f^n(\emptyset) \quad \text{and} \quad T_i = \bigcup_{n\geq 0} g^n(\emptyset).$$

Moreover, from Lemma 8 and the fact that $\widehat{T}$ is i-based,

$$Q \cap \bigcup_{n\geq 0} f^n(\emptyset) = Q \cap \bigcup_{n\geq 0} \phi^n(\emptyset).$$

While in general $\widehat{\mathsf{MX}}_i$ does not distribute over union, $\widehat{G}$ is four-partite, therefore:

$$\widehat{\mathsf{MX}}_i(S_1 \cup S_2) = \widehat{\mathsf{MX}}_i S_1 \cup \widehat{\mathsf{MX}}_i S_2$$

if every state in S_1 is of different type from every state in S_2. Hence,

$$\bigcup_{n\geq 0} \phi^n(\emptyset) = \bigcup_{n\geq 0} \bigcup_{0\leq j<4} \widehat{\mathsf{MX}}_i^j g^{\lfloor \frac{n+3-j}{4} \rfloor}(\emptyset).$$

This can be rewritten as

$$\bigcup_{n\geq 0} \phi^n(\emptyset) = \bigcup_{0\leq j<4} \bigcup_{n\geq 0} \widehat{\mathsf{MX}}_i^j g^n(\emptyset).$$

Only the first of the four components ($j = 0$) has a non-null intersection with the set of base states. Therefore

$$Q \cap \bigcup_{n\geq 0} f^n(\emptyset) = Q \cap \bigcup_{n\geq 0} \phi^n(\emptyset) = \bigcup_{n\geq 0} g^n(\emptyset).$$

Hence $T_i = Q \cap \widehat{T}_i$. □

As shown in Fig. 1, Zielonka's algorithm recurs on the subgame $\widehat{G}^S$ induced by the removal of $\widehat{T}_i$ from $\widehat{G}$. In Fig. 5, $\widehat{T}_0 \neq \widehat{T}_1$ but $T_0 = T_1$ which indicates that the subgame $\widehat{G}^S$ is not necessarily the associate of the subgame of G obtained by removing T_i. Two issues confront us. The first is that the game graph of a subgame $G\backslash T_i$ is not the subgraph induced by $Q\backslash T_i$. The input-based subgame must account for the fact that even when two base states of a turn-based game belong to a subgame, some of the paths connecting them may not be entirely contained in it. In Fig. 5, the subgame of G induced by removing T_0 must not contain the edge $(q, (1, 1, 1), t)$ even though both q and t belong to the subgame $(G\backslash T_0)$. The following is the proper definition of the induced subgame with respect to the removal of i-attraction T_i from G.

Definition 13 An input-based game $G^{s_i} = (\Sigma, Q^{s_i}, D^{s_i}, \delta^{s_i}, \Pi^{s_i})$ is a subgame of input-based game $G = (\Sigma, Q, D, \delta, \Pi)$ induced by removing from G an i-attraction T_i, if

1. $\mathcal{T} = T_i \times \Sigma$
2. $Q^{s_i} = Q\backslash T_i$,
3. $D^{s_1} = D\backslash\big(\mathcal{T} \cup \mathsf{EX}\, T_1 \cup \{(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \mid \exists\sigma'_1 . \forall\sigma'_{0p} . (q, (\sigma_{0d}, \sigma'_1, \sigma'_{0p})) \notin (D\backslash \mathsf{EX}\, T_1)\}\big)$,
   $D^{s_0} = D\backslash\big(\mathcal{T} \cup \{(q, (\sigma_{0d}, \sigma_1, \sigma_{0d})) \mid \exists\sigma'_{0p} . (q, (\sigma_{0d}, \sigma_1, \sigma'_{0d})) \in \mathsf{EX}\, T_0\}\big)$,

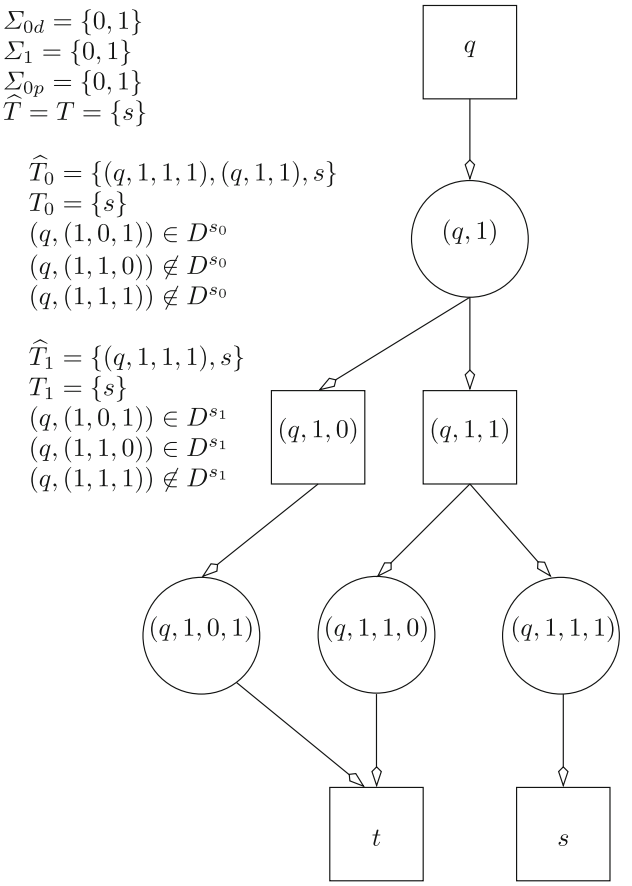

Fig. 5 The subgame of an input-based game induced by the removal of states in i-attraction depends on the value of i

4. $\forall u, v \in Q^{s_i} . \forall (u, \sigma) \in D^{s_i} .$
   $\delta^{s_i}(u, \sigma) = v \leftrightarrow \delta(u, \sigma) = v,$
5. Π^{s_i} is the restriction of Π to Q^{s_i}.

The second issue that we confront is that the subgame $\widehat{G}^S$ is not necessarily the associate of any subgame of input-based game G. However, there exists a subgame $\widehat{G}^s$ of $\widehat{G}^S$ that is the associate of the corresponding subgame of G. Let B_u be the states of $\widehat{G}^S$ unreachable from the base states $Q \cap \widehat{Q}^S$; then $\widehat{G}^s$, the subgame of $\widehat{G}^S$ obtained by removing the states in B_u, is the subgame of $\widehat{G}$ *reach-reduced* from Q^S. Lemma 10 and 11 prove that $\widehat{G}^s$ is the associate of G^S when $\widehat{G}^S$ and G^S are obtained by removing corresponding attractions from associate games.

No play starting in a state in $\widehat{Q}^S\backslash B_u$ visits any state in B_u. Thus, no strategy in $\widehat{G}^s$ relies on states in B_u, and a winning strategy for Player i in $\widehat{G}^s$ is a winning strategy in $\widehat{G}^S$ for the same player from the same state. The computation of strategies for states in B_u involves attraction computations. This means that the ability to derive strategies for one game from the strategies of its associate established in Theorem 1 extends to subgames induced by corresponding i-attractions.

We now prove that the subgames encountered during the run of ib-Z on an input-based game are the associates of the reach-reduced subgames encountered when playing of the turn-based game.

Lemma 10 *Let* $\widehat{G} = (\widehat{Q}, \widehat{Q}_0, \widehat{Q}_1, \widehat{\delta}, \widehat{\Pi})$ *be the associate of input-based game* $G = (\Sigma, Q, D, \delta, \Pi)$ *and* $\widehat{T}_1$ *be* $\mathrm{attr}_1(\widehat{G}, \widehat{T})$. *If* $\widehat{G}^s$ *is the subgame of* $\widehat{G}$ *reach-reduced from* $\widehat{Q}\backslash\widehat{T}_1$ *then it is the associate of* $G^{s_1} = (\Sigma, Q^{s_1}, D^{s_1}, \delta^{s_1}, \Pi^{s_1})$, *where* G^{s_1} *is the subgame of* G *induced by* $Q\backslash T_1$, *where* $T_1 = Q \cap \widehat{T}_i$ *is the corresponding* 1*-attraction of* $T = Q \cap \widehat{T}$ *in* G.

Proof The parent game $\widehat{G}$ is the associate game of G, thus the base states of $\widehat{G}$ are the states of G. To prove that $\widehat{G}^s$ is the associate of G^{s_1}, we first show that the base states of $\widehat{G}^s$ are the states of G^{s_1}. According to the definition of reach-reduced game, the base states of $\widehat{G}^s$ are $Q\backslash\widehat{T}_1$. Since, $Q^{s_1} = Q\backslash T_1$ and $T_1 = Q \cap \widehat{T}_1$, the base states of $\widehat{G}^s$ are the states of G^{s_1}. From Definition 3 we know that (5) relates the non-base states of the associate game and the set of state and input-label pairs. We show that (5) also holds for G^{s_1} and $\widehat{G}^s$.

We first show that

$$\forall q \,.\, \forall\sigma_{0d} \,.\, \forall\sigma_1 \,.\, \forall\sigma_{0p} \,.(q, \sigma_{0d}, \sigma_1, \sigma_{0p}) \in \widehat{Q}^s \\ \rightarrow (q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \in D^{s_1}. \tag{6}$$

Suppose $(q, \sigma_{0d}, \sigma_1, \sigma_{0p}) \in \widehat{Q}^s$, but $(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \notin D^{s_1}$. Since $(q, \sigma_{0d}, \sigma_1, \sigma_{0p}) \in \widehat{Q}^s$, the base state q is in Q^s and therefore $q \notin T_1$. This implies that

$$(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \notin \mathcal{T}.$$

From Definition 3, $\hat{\delta}((q, \sigma_{0d}, \sigma_1, \sigma_{0p}))$ is a singleton and its element is a base state; this implies that $\hat{\delta}((q, \sigma_{0d}, \sigma_1, \sigma_{0p})) \subseteq \widehat{Q}^s$, therefore $\delta((q, (\sigma_{0d}, \sigma_1, \sigma_{0p}))) \in Q^s$ which means $(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \in \mathsf{EX}\, Q^{s_1}$. Since $Q^{s_1} \cap T_1 = \emptyset$,

$$(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \notin \mathsf{EX}\, T_1.$$

Since $\widehat{G}^s$ is reach-reduced, $(q, \sigma_{0d}, \sigma_1, \sigma_{0p})$ is reachable from q in $\widehat{G}^s$; therefore Player 1 cannot force a visit of T_1 from q in G. Since $(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \in D$, from Definition 12 we conclude that

$$(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \notin \{(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \mid \\ \exists\sigma'_1 \,.\, \forall\sigma'_{0p} \,.(q, (\sigma_{0d}, \sigma'_1, \sigma'_{0p})) \notin (D \backslash \mathsf{EX}\, T_1)\}\big).$$

Therefore $(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \in D^{s_1}$, our assumption is wrong and (6) holds.

We now show that

$$\forall q \,.\, \forall\sigma_{0d} \,.\, \forall\sigma_1 \,.\, \forall\sigma_{0p} \,.(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \in D^{s_1} \\ \rightarrow (q, \sigma_{0d}, \sigma_1, \sigma_{0p}) \in \widehat{Q}^s. \tag{7}$$

Suppose $(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \in D^{s_1}$, but $(q, \sigma_{0d}, \sigma_1, \sigma_{0p}) \notin \widehat{Q}^s$. If the base state q is not in $\widehat{Q}^s$ then $q \in \widehat{T}_1$, which implies $q \in T_1$. Therefore $(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \in \mathcal{T}$. Hence $(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \notin D^{s_1}$.

If the base state q is in $\widehat{Q}^s$, then either $(q, \sigma_{0d}, \sigma_1, \sigma_{0p}) \in \widehat{T}_1$ or $(q, \sigma_{0d}, \sigma_1, \sigma_{0p})$ is unreachable from any base state in $\widehat{Q}^s$. If $(q, \sigma_{0d}, \sigma_1, \sigma_{0p}) \in \widehat{T}_1$ then $\delta((q, (\sigma_{0d}, \sigma_1, \sigma_{0p}))) \in T_1$ because $\hat{\delta}((q, \sigma_{0d}, \sigma_1, \sigma_{0p})) \subseteq T_1$ is a singleton. Therefore $(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \in \mathsf{EX}\, T_1$. Hence $(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \notin D^{s_1}$.

In the latter case, $(q, \sigma_{0d}, \sigma_1, \sigma_{0p})$ is unreachable from any base state in $\widehat{Q}^s$ because $(q, \sigma_{0d}) \in \widehat{T}_1$. Since $(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \in D$, from Definition 12 we conclude that

$$(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \in \{(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \mid \\ \exists\sigma'_1 \,.\, \forall\sigma'_{0p} \,.(q, (\sigma_{0d}, \sigma'_1, \sigma'_{0p})) \notin (D \backslash \mathsf{EX}\, T_1)\}\big).$$

Hence $(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \notin D^{s_1}$. All the cases contradict our assumption and therefore (7) holds. Since both (6) and (7) are true, we conclude that (5) holds.

From Definition 3, the set of non-base states in $\widehat{G}^s$ can be determined directly from D^{s_1}. State $(q, \sigma_{0d}, \sigma_1, \sigma_{0p})$ is only reachable from base state q through (q, σ_{0d}) and then $(q, \sigma_{0d}, \sigma_1)$. Since (5) holds, the state $(q, \sigma_{0d}, \sigma_1, \sigma_{0p})$ is in $\widehat{G}^s$ and $\forall q \in \widehat{Q}^s \,.\, \hat{\delta}^s(q) \neq \emptyset$,

$$\begin{aligned} \widehat{Q}^s = Q^s &\cup \{(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \in D^{s_1} \mid (q, \sigma_{0d})\} \\ &\cup \{(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \in D^{s_1} \mid (q, \sigma_{0d}, \sigma_1)\} \\ &\cup \{(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \in D^{s_1} \mid (q, \sigma_{0d}, \sigma_1, \sigma_{0p})\}\,. \end{aligned}$$

The states of $\widehat{G}^s$ and the states of G^{s_1} are in correspondence with each other according to Definition 3. Since $\widehat{G}$ is the associate of G, both Π^{s_1} and δ^{s_1} are restrictions of Π and δ with respect to Q^{s_1}, while both $\widehat{\Pi}^s$ and $\widehat{\delta^s}$ are restrictions of $\widehat{\Pi}$ and $\widehat{\delta}$ with respect to $\widehat{Q}^s$. We can conclude that $\widehat{G}^s$ is the associate of G^{s_1}. □

Lemma 11 *Let* $\widehat{G} = (\widehat{Q}, \widehat{Q}_0, \widehat{Q}_1, \widehat{\delta}, \widehat{\Pi})$ *be the associate of input-based game* $G = (\Sigma, Q, D, \delta, \Pi)$ *and* $\widehat{T}_0$ *be* $\mathrm{attr}_0(\widehat{G}, \widehat{T})$. *If* $\widehat{G}^s$ *is the subgame of* $\widehat{G}$ *reach-reduced from* $\widehat{Q}\backslash\widehat{T}_0$ *then it is the associate of* $G^{s_0} = (\Sigma, Q^{s_0}, D^{s_0}, \delta^{s_0}, \Pi^{s_0})$, *where* G^{s_0} *is the subgame of* G *induced by* $Q\backslash T_0$, *where* $T_0 = Q \cap \widehat{T}_0$ *is the corresponding* 0*-attraction of* $T = Q \cap \widehat{T}$ *in* G.

Proof The parent game $\widehat{G}$ is the associate game of G, thus the base states of $\widehat{G}$ are the states of G. To prove that $\widehat{G}^s$ is the associate of G^{s_0}, we first show that the base states of $\widehat{G}^s$ are the states of G^{s_0}. According to the definition of reach-reduced game, the base states of $\widehat{G}^s$ are $Q\backslash\widehat{T}_0$. Since, $Q^{s_0} = Q\backslash T_0$ and $T_0 = Q \cap \widehat{T}_0$, the base states of $\widehat{G}^s$ are the states of G^{s_0}. From Definition 3 we know that (5) relates the non-base states of the associate game and the set of state and input-label pairs. We show that (5) also holds for G^{s_0} and $\widehat{G}^s$.

We first show that

$$\forall q \,.\, \forall\sigma_{0d} \,.\, \forall\sigma_1 \,.\, \forall\sigma_{0p} \,.(q, \sigma_{0d}, \sigma_1, \sigma_{0p}) \in \widehat{Q}^s \\ \rightarrow (q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \in D^{s_0}. \tag{8}$$

Suppose $(q, \sigma_{0d}, \sigma_1, \sigma_{0p}) \in \widehat{Q}^s$, but $(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \notin D^{s_0}$. Since $(q, \sigma_{0d}, \sigma_1, \sigma_{0p}) \in \widehat{Q}^s$, the base state q is in Q^s. Therefore $q \notin T_0$. This implies that $(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \notin \mathcal{T}$.

From Definition 3, $\hat{\delta}((q, \sigma_{0d}, \sigma_1, \sigma_{0p}))$ is a singleton and its element is a base state; this implies that $\hat{\delta}((q, \sigma_{0d}, \sigma_1, \sigma_{0p})) \subseteq \widehat{Q}^s$, therefore $\delta((q, (\sigma_{0d}, \sigma_1, \sigma_{0p}))) \in Q^s$ which means $(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \in \mathsf{EX}\, Q^{s_0}$. Since $Q^{s_0} \cap T_0 = \emptyset$, $(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \notin \mathsf{EX}\, T_0$.

Since $\widehat{G}^s$ is reach-reduced, $(q, \sigma_{0d}, \sigma_1, \sigma_{0p})$ is reachable from q in $\widehat{G}^s$; therefore Player 0 cannot force a visit of T_0 from q in G. Since $(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \in D$, from Definition 12 we conclude that

$$(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \notin \{(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \mid \exists\, \sigma'_{0p} .(q, (\sigma_{0d}, \sigma_1, \sigma'_{0d})) \in \mathsf{EX}\, T_0\}).$$

Therefore $(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \in D^{s_0}$, our assumption is wrong and (8) holds.

We now show that

$$\forall q \,.\, \forall \sigma_{0d} \,.\, \forall \sigma_1 \,.\, \forall \sigma_{0p} \,.(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \in D^{s_1} \rightarrow (q, \sigma_{0d}, \sigma_1, \sigma_{0p}) \in \widehat{Q}^s. \tag{9}$$

Suppose $(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \in D^{s_0}$, but $(q, \sigma_{0d}, \sigma_1, \sigma_{0p}) \notin \widehat{Q}^s$. If the base state q is not in $\widehat{Q}^s$ then $q \in \widehat{T}_0$, which implies that $q \in T_0$. Therefore $(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \in \mathcal{T}$. Hence $(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \notin D^{s_0}$.

If the base state $q \in \widehat{Q}^s$, then either $(q, \sigma_{0d}, \sigma_1, \sigma_{0p}) \in \widehat{T}_0$ or $(q, \sigma_{0d}, \sigma_1, \sigma_{0p})$ is unreachable from any base state in $\widehat{Q}^s$. If $(q, \sigma_{0d}, \sigma_1, \sigma_{0p}) \in \widehat{T}_0$ then $\delta((q, (\sigma_{0d}, \sigma_1, \sigma_{0p}))) \in T_0$ because $\hat{\delta}((q, \sigma_{0d}, \sigma_1, \sigma_{0p})) \subseteq T_0$ is a singleton. Therefore $(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \in \mathsf{EX}\, T_0$. Hence $(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \notin D^{s_0}$.

In the latter case, $(q, \sigma_{0d}, \sigma_1, \sigma_{0p})$ is unreachable from any base state in $\widehat{Q}^s$ because $(q, \sigma_{0d}, \sigma_1) \in \widehat{T}_0$. Since $(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \in D$, from Definition 12 we conclude that

$$(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \in \{(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \mid \exists \sigma'_{0p} .(q, (\sigma_{0d}, \sigma_1, \sigma'_{0d})) \in \mathsf{EX}\, T_0\}).$$

Hence $(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \notin D^{s_0}$. All the cases contradict our assumption and therefore (8) holds. Since both (8) and (9) are true, we conclude that(5) holds.

From Definition 3, the set of non-base states in $\widehat{G}^s$ can be determined directly from D^{s_0}. State $(q, \sigma_{0d}, \sigma_1, \sigma_{0p})$ is only reachable from base state q through (q, σ_{0d}) and then $(q, \sigma_{0d}, \sigma_1)$. Since $\forall q \in \widehat{Q}^s \,.\, \hat{\delta}^s(q) \neq \emptyset$,

$$\begin{aligned}\widehat{Q}^s = Q^s &\cup \{(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \in D^{s_1} \mid (q, \sigma_{0d})\} \\ &\cup \{(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \in D^{s_1} \mid (q, \sigma_{0d}, \sigma_1)\} \\ &\cup \{(q, (\sigma_{0d}, \sigma_1, \sigma_{0p})) \in D^{s_1} \mid (q, \sigma_{0d}, \sigma_1, \sigma_{0p})\}\,.\end{aligned}$$

The states of $\widehat{G}^s$ and the states of G^{s_0} are in correspondence with each other according to Definition 3. Since $\widehat{G}$ is the associate of G, both Π^{s_0} and δ^{s_0} are restrictions of Π and δ with respect to Q^{s_0}, while both $\widehat{\Pi}^s$ and $\widehat{\delta}^s$ are restrictions of $\widehat{\Pi}$ and $\widehat{\delta}$ with respect to $\widehat{Q}^s$. We can conclude that $\widehat{G}^s$ is the associate of G^{s_0}. □

The algorithm of ib-Z is obtained from Zielonka's algorithm in Fig. 1 by replacing Definition 5 with Definition 12 and Definition 6 with Definition 13. We now prove that the adapted algorithm correctly computes Player i's winning and losing states.

Theorem 4 *For an input-based game $G = (\Sigma, Q, D, \delta, \pi)$* ib-Z *returns $W_0 = Q \cap \widehat{W}_0$ and $W_1 = Q \cap \widehat{W}_1$ where $\widehat{W}_0$ and $\widehat{W}_1$ are the set of states returned by Zielonka's algorithm for the associate turn-based game $\widehat{G} = (\widehat{Q}, \widehat{Q}_0, \widehat{Q}_1, \hat{\delta}, \hat{\pi})$.*

Proof We prove by induction that ib-Z returns Player i's winning and losing base states. If G is an empty game then $\widehat{G}$ is also empty; then, on Line 2, ib-Z returns $W_i = W_j = \emptyset$ in agreement with the result of Zielonka's algorithm on $\widehat{G}$. Therefore the base case holds trivially.

If G is a non-empty game then so is $\widehat{G}$. Both ib-Z and Zielonka's algorithm recur on subgames of G and $\widehat{G}$ respectively. We now show the correspondence between the two subgames. At Line 3 let $T \subseteq Q$ be the set of states colored k with respect to π and $\widehat{T} \subseteq \widehat{Q}$ be the set of states colored k with respect to $\widehat{\pi}$. From Definition 3 we know that $T = Q \cap \widehat{T}$. The set of states $\widehat{T}$ is i-based because $\overline{T} = \emptyset$. At Line 5 let T_i be the i-attraction of T in G and $\widehat{T}_i$ be the i-attraction of $\widehat{T}$ in $\widehat{G}$. Thanks to Lemma 9, $T_i = Q \cap \widehat{T}_i$. At Line 6 ib-Z recurs on the subgame $G^{s_i} = (\Sigma, Q^{s_i}, D^{s_i}, \delta^{s_i}, \pi^{s_i})$ where $Q^{s_i} = Q \setminus T_i$. Since $T_i = Q \cap \widehat{T}_i$ and thanks to Lemmas 10 and 11, the reach-reduced subgame $\widehat{G}^s = (\widehat{Q}^s, \widehat{Q}^s_0, \widehat{Q}^s_1, \hat{\delta}^s, \hat{\pi}^s)$ induced by $\widehat{Q} \setminus \widehat{T}_i$ is the associate of G^{s_i}. On the other hand, Zielonka's algorithm recurs on the subgame $\widehat{G}^S = (\widehat{Q}^S, \widehat{Q}^S_0, \widehat{Q}^S_1, \hat{\delta}^S, \hat{\pi}^S)$ induced by $\widehat{Q} \setminus \widehat{T}_i$. Since Player i's winning and losing base states in $\widehat{G}^s$ are the same as the winning and losing base states in $\widehat{G}^S$, we can invoke the inductive hypothesis and conclude that at Line 7, $U_i = Q \cap \widehat{U}_i$ and $U_j = Q \cap \widehat{U}_j$.

If at Line 7 $U_j = \emptyset$ then $Q \cap \widehat{U}_j = \emptyset$. Since every path through a non-base state must visit some base state, it follows that $Q \cap \widehat{U}_j = \emptyset$ implies $\widehat{U}_j = \emptyset$. Therefore the Line 7 evaluates to true in Zielonka's algorithm as well. Hence, all states of both G and $\widehat{G}$ are winning for Player i. ib-Z returns $W_i = Q = Q \cap \widehat{Q}$ and $W_j = Q \cap \widehat{Q} = \emptyset$ in agreement with the results of Zielonka's algorithm on $\widehat{G}$.

If at Line 7 $U_j \neq \emptyset$ then $\widehat{U}_j \neq \emptyset$ because we have established $U_j = Q \cap \widehat{U}_j$. The set $\widehat{U}_j$ is j-based because $\widehat{U}_j$ is j-closed in $\widehat{G}$. At Line 12, with argument similar to the one for Line 6, we can again invoke the inductive hypothesis, hence $U_i = Q \cap \widehat{U}_i$ and $U_j = Q \cap \widehat{U}_j$. At Line 13 ib-Z assigns all the base states in $Q \setminus U_i$ to Player j in agreement with Zielonka's algorithm. Therefore $U_i = Q \cap \widehat{U}_i$ and $U_j = Q \cap \widehat{U}_j$ is true after Line 13.

At Line 17 ib-Z returns $W_i = Q \cap \widehat{W}_i = Q \cap \widehat{U}_i$ and $W_j = Q \cap \widehat{W}_j = Q \cap \widehat{U}_j$. Therefore ib-Z correctly identifies the set of Player i's winning and losing states for G. □

By extending Lemma 9 and Theorem 4 we can show that the strategies for both players computed by ib-Z are related to those returned by Zielonka's algorithm through Definition 4.

4.2 Framework

Our implementation takes as input a set of LTL properties and NBWs. In addition, it takes as input a partial model of the environment in Verilog. Each LTL property is translated into an NBW. Safety properties are identified at this point, by checking Definition 7 on the NBWs. If the check passes then we use the subset construction method [40] for determinization to a DPW. The rest of NBWs are converted into DPWs of minimal index. We use an extension of Wring [39] for translating an LTL formula into an NBW and then into a DPW. The composition of the DPWs and the Verilog model define a game and a conjunctive generalized parity winning condition as described in Sect. 3.

A winning strategy for Player 1 in this game—if it exists—is a realization of the specifications. The game has an initial vertex, corresponding to the initial states of the various DPWs. The winning set of Player 1 must contain this vertex.

Alternatively our implementation can also handle a specification as described at the end of Sect. 3. As discussed earlier, PERSISTENCE- FIRST described in Fig. 4 restricts the game to the Player i's winning states where Player i is obligated to satisfy the acceptance condition of the game. After the algorithm returns, one strategy is extracted from the computed set of strategies. The algorithm outlined in [5] is used to heuristically attempt to extract one strategy with low implementation cost, which is written to the output in the form of a Verilog module.

5 Related work

In this section, we discuss approaches to game solving that promote some form of decomposition or incrementality and compare these ideas to our work. For a broader discussion the reader is referred to [42].

In [36], the specification conforming to restricted LTL is translated into a Generalized Streett(1) game. LTL properties that translate into DBWs can be added to the specification by adding the acceptance conditions of the DBWs to the liveness specification that has the following shape:

$$\bigwedge_{1 \le k \le m} (\mathsf{G}\,\mathsf{F}\, J_k^1) \rightarrow \bigwedge_{1 \le k \le n} (\mathsf{G}\,\mathsf{F}\, J_k^2). \tag{10}$$

The transition relation of the DBWs is expressed as safety constraints. Specifications which cannot be directly translated into a General Strett(1) game can sometimes be hand-modified to be translated into a General Streett(1) game; but such modifications are not always possible.

The "Safraless" approach of [22,24] is another example of multi-stage synthesis process. When the specification is a safety property, the universal co-Büchi tree automaton produced as intermediate stage can be made weak; this speeds up synthesis. However, the "Safraless" approach does not separate the safety component of the specification. Rather, in [22] it is extended so that synthesizing the conjunction of specifications can take advantage of the computations performed for the individual specifications. Our approach benefits from the weakness of the specification when it applies to some properties only, thanks to Theorem 3.

The "Safraless" approach of [11,12] translates the specification to a universal co-Büchi word automaton (UCW). Instead of determinizing UCW, a counter with a bound K is added to each state of UCW and the resulting automaton is denoted by UKCW. For every accepting run of UKCW the accepting states are visited K or fewer times. A UKCW is a safety automaton and it can easily be determinized through the well known subset construction. UKCW is interpreted as a two player game between the system and the environment. Since L(UKCW) $\subseteq$ L(UCW), a game has to be played for every $K \geq 0$ until the system is able to win the game or it is proven that the specification is unrealizable. In [12] the authors convert each LTL formula in the specification to a UKCW interpreted as a two player game. It is also shown that for some $K \geq 0$ if the system wins all these games then the specification is realizable and an implementation can be synthesized. Their approach is shown to be practical for small examples but it remains to be seen how it will perform on examples such as the Amba bus specification [5,13], both in terms of runtime and quality of implementation.

The authors of [6] study a partial *permissivity* ordering on non-deterministic strategies. They show that safety games have maximal strategies that are memoryless, and that every game that has a maximal memoryless strategy is equivalent to a safety game. For other types of games, they define a *permissive* strategy as one that allows the behaviors of all memoryless strategies. They show how to build such permissive strategies with memory bounded by the size of the game graph. In our approach, we account for all the strategies for persistent winning conditions (not just the memoryless ones) without introducing memory.

The authors of [17] propose an incremental approach in which a sequence of *good-for-games* automata is built until a winning strategy is found or specification is proved unrealizable. The algorithm does not lend itself to efficient symbolic implementation. Since no experimental results were

published it is difficult to assess the effectiveness of that approach.

The authors of [33] also translate each property in the specification to DPW but they avoid the explicit determinization phase by employing symbolic methods for determinization. The DPWs are composed together to form a game with a conjunctive parity winning condition. They too use [8] to solve the conjunctive parity game but without simplification of the conjunctive parity winning condition through Theorem 3. Their symbolic approach to determinization is an example of an algorithm that allocates one BDD variable to each state as to represent collections of subsets by the characteristic functions of their occurrence vectors (as also required by the algorithms of [17,24]). Therefore, their approach is limited in most cases to automata of hundred states or less. For such reasons we prefer an explicit implementation of Piterman's improved determinization procedure [35] to the approach of [17,33].

6 Experiments

The approach described in Sects. 3 and 4 has been implemented in Vis [3] as an extension of the technique described in [42]. In this section we report on experiments conducted on examples coming from [42] (RRobin) and [4] (Amba Bus). Crane is the controller of a traveling crane [27].

Table 1 illustrates the benefits of the two-stage approach to playing the generalized parity games described in Sect. 3 and then compares it to the approach of Anzu [4]. SF is the approach of Sect. 3 in which every property is converted into a DPW. Pre-SF is the approach in which properties of the following type

$$\mathsf{G}\left(\bigvee x^1 \wedge \ldots x^j \wedge \mathsf{X}(x^1 \wedge \ldots x^j)\right) \tag{11}$$

are conjoined together. The resulting property is treated as an implicit transition relation. The translation of this implicit transition relation to a valid transition relation still requires some care as observed in the following example.

Example 1 System controls $\{m, g_0, g_1\}$ and the environment controls $\{h\}$. System does not snoop anything (the system makes its selection first and environment completes the move).

$$\mathsf{G}(h \rightarrow (g_0 \rightarrow \mathsf{X}(\neg m))), \quad \mathsf{G}(h \rightarrow (g_1 \rightarrow \mathsf{X}(m))),$$
$$\mathsf{G}(h \rightarrow (g_0 \rightarrow \mathsf{X}(g_0))) \quad \text{and} \quad \mathsf{G}(h \rightarrow (g_1 \rightarrow \mathsf{X}(g_1))).$$

The system must make sure that g_0 and g_1 are always mutually exclusive. The mutual exclusion property is implicitly stated in the above LTL formulae.

Table 1 Experimental results

Model	Safety		Parity		Properties	Latches			Time (s)			Reduced		Time (s)
	E	S	E	S	Anzu	SF	Pre-SF	Anzu	SF	Pre-SF	Anzu	Opt	Pre+Opt	No-Opt
RRobin	0	10	0	0	20	7	n/a	6	0.16	n/a	0.2	8	n/a	0.19
Crane	2	12	2	2	n/a	11	n/a	n/a	6.3	n/a	n/a	3	n/a	6.7
GenBuf2	13	19	2	3	51	40	n/a	18	4.03	n/a	0.38	26	n/a	7.64
GenBuf3	15	25	2	4	61	49	n/a	21	6.49	n/a	0.6	31	n/a	14.09
GenBuf4	17	32	2	5	78	55	n/a	24	11.04	n/a	0.99	36	n/a	24.49
GenBuf5	18	43	2	6	79	61	n/a	27	15.48	n/a	2.11	45	n/a	49.20
GenBuf6	20	52	2	7	96	71	n/a	29	28.28	n/a	4.74	47	n/a	215.79
Amba2	3	17	2	3	56	37	19	24	6.87	3.4	2.39	6	3	4.83
Amba3	4	22	2	4	68	42	23	30	14.2	4.6	44.67	9	4	31.80
Amba4	5	26	2	5	80	48	25	34	109.9	10.8	35.30	6	5	578.8
Amba5	6	31	2	6	93	56	28	39	139.7	15.7	224.06	11	6	345.7
Amba6	7	34	2	7	105	55	31	43	301.1	20.6	1011.7	11	8	789.9
Amba7	8	38	2	8	117	61	33	48	965.6	71.7	1758.5	10	8	1955.6
Amba8	9	41	2	9	129	67	35	52	875.3	131.7	2034.9	12	9	2375.6
Amba9	10	44	2	10	141	77	38	57	1439.6	275.5	7861.2	14	10	8128.7
Amba10	11	48	2	11	153	81	41	61	3727.6	233.3	28319.8	16	12	11234.6
Amba11	12	51	2	12	165	87	43	65	3154.0	224.5	8403.3	22	13	TO
Amba12	13	55	2	13	177	92	46	69	6641.2	742.3	49138.7	21	15	TO
Amba13	14	60	2	14	189	98	47	73	32562.4	504.9	13163.4	25	15	TO
Amba14	15	64	2	15	200	105	49	77	12202.2	1006.7	17104.9	19	16	TO
Amba15	16	69	2	16	212	–	52	–	TO	2281.4	TO	–	17	TO

Table 2 Experimental results of improved AMBA specification

Model	Safety		Parity		Properties	Latches		Time (s)		*Reduced*
	E	S	E	S	ANZU	SF	ANZU	SF	ANZU	SF
AMBA2	15	16	2	3	55	48	23	10.17	1.67	7
AMBA3	17	21	2	4	68	54	29	31.53	6.75	7
AMBA4	19	26	2	5	79	65	33	83.74	20.83	9
AMBA5	21	31	2	6	92	72	38	218.22	158.40	10
AMBA6	23	35	2	7	103	78	42	1075.99	412.28	11

The implicit transition relation is translated to a game $\mathcal{G}$ such that every state of $\mathcal{G}$ is winning for the system. The game $\mathcal{G}$ is language equivalent to the game obtained from the incremental approach of SF. [4] The game $\mathcal{G}$ is significantly smaller than the game obtained from the incremental approach. SF then operates on the remaining properties [which were not of type (11)] with the exception that instead of G^0 being an identity game we have $G^0 = \mathcal{G}$.

ANZU is a synthesis tool for GR(1) specifications. For each model, the numbers of persistence and non-persistence properties are given for both system (S) and environment (E) in Columns 2–5. All the persistent properties in the experiments were of the safety type. Column 6 reports the number of properties required by ANZU. Columns 7–9 report the number of latches in the final solution for the three methods. The number of latches represents a crude measure of the magnitude of the synthesized model. Columns 10–12 report the time spent to compute the final solution for the three methods. Columns 13 and 14 report the number of constant and equivalent bits removed by the optimizations applied in between games by SF and Pre-SF. The last column reports the time required to compute the solution when optimizations are turned off for SF. Times were measured on a 2.4-GHz Quad Core Pentium Duo with 4 GB of RAM. The table shows that the advantages of incremental game playing interleaved with optimizations are quite significant for the larger examples.

A monolithic specification of an 8-client AMBA Arbiter, where all the properties were combined together, was also considered, but the game play did not complete in three hours; playing the game incrementally, however, finished in just over two minutes.

Table 2 compares our approach with ANZU on the improved AMBA specification of [13]. Our implementation of Pre-SF does not currently handle specifications in which both the environment assumptions and system guarantees contain properties of the type of (11). Even though the new specification is meant to produce improved synthesis results the limitation of the pre-synthesis technique results in poor performance of our algorithm. Without pre-synthesis some outputs of the reactive system are defined by multiple latches instead of a single latch. This is evident when comparing the solutions of SF and Pre-SF.

The comparison with ANZU reveals several interesting differences. The number of properties required to specify the same system is much smaller in our approach. This reflects the fact that our two-stage approach accepts a more general form of specification. The input properties to our tool and ANZU are identical except for the fact that if a safety property requires memory to be realizable then ANZU requires that the property be broken down into sub-properties and partially synthesized by hand. Table 3 compares two properties from the AMBA Arbiter specification; one that is identical for both tools; the other that needs decomposition. Thanks to incremental synthesis and the optimizations that it allows, our synthesis times are quite competitive. Our approach does, may however, pay a price if pre-synthesis is not applicable. More optimizations, and in particular, avoiding the blowup of state space due to the properties described by (11) are required to produce more efficient implementations.

7 Conclusions

We have shown that in the game-theoretic approach to the synthesis of reactive systems it is possible to separate the safety and persistence component of the specification. This allows one to solve the generalized parity game into which the specification is translated into two stages. In the presence of persistence and safety properties—a common occurrence—one therefore achieves an improved bound on the runtime of the game solving procedure. The separation in two stages also opens the door to optimizations of the game graph in between the two stages only the most straightforward of which are currently exploited by our implementation. Though these optimizations may not affect the worst-case complexity, they are practically significant and allow our algorithm to outperform in speed the best known synthesis algorithms in spite of accepting a more general and less detailed specification. Analysis of our initial results has suggested numerous

[4] Let $\Phi = \{\phi_1, \phi_2, \ldots, \phi_k\}$ be the set of properties of type (11), then the incremental approach produces the game G^k. The translation procedure guarantees that $L(\mathcal{G}) = L(G^k)$.

Table 3 Partial LTL specification of 2-Client AMBA Bus Arbiter

Property type	SF: safety-first	ANZU
1. In case there are no requests, the bus is granted to *Master 0*	$\mathsf{G}((\text{DECIDE} \wedge \forall i : \neg\text{HBUSREQ}i)) \rightarrow \mathsf{X}(\text{HGRANT0}))$	$\mathsf{G}((\text{DECIDE} \wedge \forall i : \neg\text{HBUSREQ}i) \rightarrow X(\text{HGRANT0}))$
2. No spurious grants except to *Master 0*	$\forall i \neq 0 . \mathsf{G}(\neg\text{HGRANT}i \rightarrow (\text{HBUSREQ}i \,\mathsf{R}\, \neg\text{HGRANT}i)$	$\forall i \neq 0 . \mathsf{G}(((Q = \text{INIT}) \wedge (\text{HGRANT}i \vee \text{HBUSREQ}i)) \rightarrow \mathsf{X}(Q = \text{INIT}))$ $\forall i \neq 0 . \mathsf{G}(((Q = \text{INIT}) \wedge (\neg\text{HGRANT}i \wedge \neg\text{HBUSREQ}i)) \rightarrow \mathsf{X}(Q = \text{NG}))$ $\forall i \neq 0 . \mathsf{G}(((Q = \text{NG}) \wedge (\neg\text{HGRANT}i \wedge \neg\text{HBUSREQ}i)) \rightarrow \mathsf{X}(Q = \text{NG}))$ $\forall i \neq 0 . \mathsf{G}(((Q = \text{NG}) \wedge \text{HBUSREQ}i) \rightarrow \mathsf{X}(Q = \text{INIT}))$ $\forall i \neq 0 . \mathsf{G}(((Q = \text{NG}) \wedge \text{HGRANT}i) \rightarrow \bot)$

optimizations that should further increase the speed of synthesis and the quality of the resulting reactive system.

Acknowledgments Thanks to Yashdeep Godhal, Krishnendu Chatterjee and Barbara Jobstmann for their help with ANZU and the AMBA bus models.

References

1. Alpern, B., Schneider, F.B.: Defining liveness. Inf. Process. Lett. **21**, 181–185 (1985)
2. Alpern, B., Schneider, F.B.: Recognizing safety and liveness. Distrib. Comput. **2**, 117–126 (1987)
3. Brayton, R.K., et al.: VIS: a system for verification and synthesis. In: Henzinger, T., Alur, R. (eds.) Eighth Conference on Computer Aided Verification (CAV'96). LNCS, vol. 1102, pp. 428–432. Springer, Rutgers University (1996)
4. Bloem, R., Galler, S., Jobstmann, B., Piterman, N., Pnueli, A., Weiglhofer, M.: Automatic hardware synthesis from specifications: a case study. In: Proceedings of the Design, Automation and Test in Europe, pp. 1188–1193 (2007)
5. Bloem, R., Galler, S., Jobstmann, B., Piterman, N., Pnueli, A., Weiglhofer, M.: Specify, compile, run: hardware form PSL. In: 6th International Workshop on Compiler Optimization Meets Compiler Verification 2007. Electronic Notes in Theoretical Computer Science. http://www.entcs.org/
6. Bernet, J., Janin, D., Walukiewicz, I.: Permissive strategies: from parity games to safety games. RAIRO Theor. Inf. Appl. **36**(3), 261–275 (2002)
7. Büchi, J.R.: On a decision method in restricted second order arithmetic. In: Proceedings of the 1960 International Congress on Logic, Methodology, and Philosophy of Science, pp. 1–11. Stanford University Press (1962)
8. Chatterjee, K., Henzinger, T.A., Piterman, N.: Generalized parity games. In: 10th International Conference on Foundations of Software Science and Computation Structures. LNCS, vol. 4423, pp. 153–167. Springer, Berlin (2007)
9. Carton, O., Maceiras, R.: Computing the Rabin index of a parity automaton. Theor. Inf. Appl. **33**, 495–505 (1999)
10. Emerson, E.A., Jutla, C.S.: Tree automata, mu-calculus and determinacy. In Proceedings of the 32nd IEEE Symposium on Foundations of Computer Science, pp. 368–377 (1991)
11. Emmanuel, F., Jin, N., Raskin, J.-F.: An antichain algorithm for LTL realizability. In: Proceedings of the 21st International Conference on Computer Aided Verification, pp. 263–277 (2009)
12. Filiot, Emmanuel, Jin, Nayiong, Raskin, Jean-François: Compositional algorithms for LTL synthesis. In Proceedings of the 8th international conference on Automated technology for verification and analysis. LNCS, vol. 6252, pp. 112–127, 2010
13. Godhal, Y., Chatterjee, K., Henzinger, T.: Synthesis of amba ahb from formal specification: a case study. Int. J. Softw. Tools Technol. Transf. (STTT), 1–17, (2011)
14. Gerth, R., Peled, D., Vardi, M.Y., Wolper, P.: Simple on-the-fly automatic verification of linear temporal logic. In: Protocol Specification, Testing, and Verification, pp. 3–18. Chapman & Hall, London (1995)
15. Henzinger, T., Kupferman, O., Rajamani, S.: Fair simulation. In: Proceedings of the 9th International Conference on Concurrency Theory (CONCUR'97). LNCS, vol. 1243, pp. 273–287. Springer, Berlin (1997)
16. Horn, F.: Streett games on finite graphs. In: Workshop on Games in Design and Verification Edimburgh, UK, July 2005
17. Henzinger, T.A., Piterman, N.: Solving games without determinization. In: 15th Conference on Computer Science Logic, Szeged, Hungary. LNCS, vol. 4207, pp. 394–409 (2006)
18. Harding, A., Ryan, M., Schobbens, P.Y.: A new algorithm for strategy synthesis in LTL games. In: Tools and Algorithms for the Construction and Analysis of Systems, Edinburgh, UK. LNCS, vol. 3440, pp. 477–492 (2005)
19. Jurdziński, M., Paterson, M., Zwick, U.: A deterministic subexponential algorithm for solving parity games. In: Proceedings of ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, FL, pp. 117–123 (2006)
20. Jurdziński, M.: Small progress measures for solving parity games. In: STACS 2000, 17th Annual Symposium on Theoretical Aspects of Computer Science, Lille, France. LNCS, vol. 1770, pp. 290–301. Springer, Berlin (2000)
21. Klein, U., Pnueli, A.: Revisiting synthesis of GR(1) specifications. In: Barner, S., Kroening, D., Raz, O. (eds) Proceeding of the 6th International Haifa Verification Conference (HVC '10). LNCS, vol. 6504, pp. 161–181 (2011)
22. Kupferman, O., Piterman, N., Vardi, M.Y.: Safraless compositional synthesis. In: Eighteenth Conference on Computer Aided Verification. LNCS, vol. 4144, pp. 31–44 (2006)
23. Kupferman, O., Vardi, M.Y.: Model checking of safety properties. In: Halbwachs, N., Peled, D. (eds.) Eleventh Conference on

Computer Aided Verification (CAV'99). LNCS, vol. 1633, pp. 172–183. Springer, Berlin (1999)
24. Kupferman, O., Vardi, M.Y.: Safraless decision procedures. In: Foundations of Computer Science, Pittsburgh, PA, pp. 531–542 (2005)
25. Löding, C.: Optimal bounds for transformations of ω-automata. In: Proceedings of the 19th Conference on Foundations of Software Technology and Theoretical Computer Science, 1999. LNCS, vol. 1738 (1999)
26. Landweber, L.H.: Decision problems for ω-automata. Math. Syst. Theory **3**(4), 376–384 (1969)
27. Lindner, T.: Case Study "Production Cell": A Comparative Study in Formal Software Development, chap. 2, pp. 9–21. FZI (1994)
28. Lichtenstein, O., Pnueli, A.: Checking that finite state concurrent programs satisfy their linear specification. In: Proceedings of the Twelfth Annual ACM Symposium on Principles of Programming Languages, New Orleans, pp. 97–107 (1985)
29. Löding, C., Thomas, W.: Alternating automata and logics over infinite words. In: Theoretical Computer Science (TCS 2000). LNCS, vol. 1872, pp. 521–535. Springer, Berlin (2000)
30. Martin, D.A.: Borel determinacy. Ann. Math. Second Ser. **102**, 363–371 (1975)
31. Mostowski, A.W.: Regular expressions for infinite trees and a standard form of automata. In: Skowron, A. (ed.) Computation Theory. LNCS, vol. 208, pp. 157–168. Springer, Berlin (1984)
32. Manna, Z., Pnueli, A.: A hierarchy of temporal properties. In: Annual ACM Symposium on Principles of Distributed Computing, Quebec City, Quebec, Canada, pp. 377–410 (1990)
33. Morgenstern, A., Schneider, K.: Exploiting the temporal logic hierarchy and the non-confluence property for efficient LTL synthesis. In: Games, Automata, Logics, and Formal Verification (GandALF). Electronic Proceedings in Theoretical Computer Science (EPTCS), Minori, Italy, vol. 25, pp. 89–102 (2010)
34. Muller, D.E., Saoudi, A., Schupp, P.: Alternating automata, the weak monadic theory of trees and its complexity. Theor. Comput. Sci. **97**, 233–244 (1992)
35. Piterman, N.: From nondeterministic Büchi and Streett automata to deterministic parity automata. In: 21st Symposium on Logic in Computer Science, Seattle, WA, pp. 255–264 (2006)
36. Piterman, N., Pnueli, A., Saár, Y.: Synthesis of reactive(1) designs. In: 7th International Conference on Verification, Model Checking and Abstract Interpretation. LNCS, vol. 3855, pp. 364–380. Springer, Berlin (2006)
37. Pnueli, A., Rosner, R.: On the synthesis of a reactive module. In: Proceedings of Symposium on Principles of Programming Languages (POPL '89), pp. 179–190 (1989)
38. Safra, S.: Complexity of Automata on Infinite Objects. PhD thesis, The Weizmann Institute of Science, March 1989
39. Somenzi, F., Bloem, R.: Efficient Büchi automata from LTL formulae. In: Emerson, E.A., Sistla, A.P. (eds.) Twelfth Conference on Computer Aided Verification (CAV'00). LNCS, vol. 1855, pp. 248–263. Springer, Berlin (2000)
40. Scott, D.: Finite automata and their decision problems. IBM J. Res. Dev. **3**, 114–125 (1959)
41. Sistla, A.P.: Safety, liveness and fairness in temporal logic. Formal Aspects Comput. **6**, 495–511 (1994)
42. Sohail, S., Somenzi, F., Ravi, K.: A hybrid algorithm for LTL games. In: Verification, Model Checking and Abstract Interpretation, San Francisco, CA. LNCS, vol. 4905, pp. 309–323 (2008)
43. Thomas, W.: On the synthesis of strategies in infinite games. In: Proceedings of the 12th Annual Symposium on Theoretical Aspects of Computer Science. LNCS, vol. 900, pp. 1–13. Springer, Berlin (1995)
44. Wolper, P., Vardi, M.Y., Sistla, A.P.: Reasoning about infinite computation paths. In: Proceedings of the 24th IEEE Symposium on Foundations of Computer Science, pp. 185–194 (1983)
45. Zielonka, W.: Infinite games on finitely coloured graphs with applications to automata on infinite trees. Theor. Comput. Sci. **200**(1–2), 135–183 (1998)