Faster and Smaller Solutions of Obliging Games

Daniel Hausmann ✉ 🄳
University of Gothenburg, Gothenburg, Sweden
Chalmers University of Technology, Gothenburg, Sweden
University of Liverpool, Liverpool, United Kingdom

Nir Piterman ✉ 🄳
University of Gothenburg, Gothenburg, Sweden
Chalmers University of Technology, Gothenburg, Sweden

―――― **Abstract** ――――

Obliging games have been introduced in the context of the game perspective on reactive synthesis in order to enforce a degree of cooperation between the to-be-synthesized system and the environment. Previous approaches to the analysis of obliging games have been small-step in the sense that they have been based on a reduction to standard (non-obliging) games in which single moves correspond to single moves in the original (obliging) game. Here, we propose a novel, large-step view on obliging games, reducing them to standard games in which single moves encode long-term behaviors in the original game. This not only allows us to give a meaningful definition of the environment winning in obliging games, but also leads to significantly improved bounds on both strategy sizes and the solution runtime for obliging games.

1 Intro

Infinite duration games [11] and their analysis are central to various logical problems in computer science; problems with existing game reductions subsume model checking [5, 16] and satisfiability checking [9, 10] for temporal logics (such as CTL or the μ-calculus), or reactive synthesis for LTL specifications [8]. Game arenas that are used in such reductions typically incorporate two antagonistic players (called player $\exists$ and player $\forall$ in the current work) that have dual objectives. Then the reductions construct game arenas and objectives in such a way that the instance of the original problem has a positive answer if and only if player $\exists$ has a strategy that ensures that the player's objective is satisfied (that is, player $\exists$ *wins* the game). *Solving* games then amounts to determining their winner. Games with Borel objectives are known to be *determined* [21], that is, for each node in them, exactly one of the players i has a winning strategy of type $V^*V_i \to V$, where V is the set of game nodes, and V_i the set of nodes controlled by player i.

Reactive synthesis [22] considers a setting in which a system works within an antagonistic environment, and the system-enviroment interaction is modelled by means of input variables from a set I (controlled by the environment), and output variables from a set O (controlled by the system). The synthesis problem then takes a logical specification φ over the variables $I \cup O$ as input and asks for the automated construction of a system in which all interactions between the system and the environment satisfy the specification; if such a system exists, then φ is said to be *realizable*. The reactive synthesis problem therefore goes beyond checking realizability by also asking for a witnessing system in the case that the input specification is realizable. While the problem has been shown to be 2ExpTime-complete for specifications that are

formulated in LTL, a landscape of algorithms and implementations has been developed that shows good performance in (some) practical use cases (e.g. [18, 23]). The feasability of these algorithms is largely owed to an underlying reduction to infinite-duration two-player games, most commonly with parity objectives. Answering the realizability problem then corresponds to deciding the winner of the reduced game, while the construction of a witnessing system for a realizable specification corresponds to the extraction of a winning strategy from that game. This motivates the interest not only in game solving algorithms, but also in the analysis and extraction of winning strategies. In particular, the amount of memory required by winning strategies in the types of games at hand determines the minimum size of witnessing systems.

Building on the described game perspective on reactive synthesis, *obliging games* have been proposed to deal with the situation that a system might trivially realize a specification by disallowing most or all interactions with the environment (cf. [2, 6]). Obliging games address this problem by requiring the system player to have a strategy that not only always guarantuees that the system's strong objective (say α_S) is achieved, but to also always keep an interaction possible in which intuitively both players cooperate to achieve a second, weak, objective (say α_W). Such a strategy then is called *graciously winning* for the system player. For example, the generalized reactivity (GR[1]) setting (cf. [4]) incorporates k different *requests* and k corresponding *grants* (R_i, G_i for $1 \leq i \leq k$). Then the objective α_S states that "if all R_i (requests) hold infinitely often, then all G_i (grants) hold infinitely often". In this setting, player $\exists$ may satisfy the objective α_S trivially by simply ensuring that, for each interaction, there is some R_i that is eventually avoided forever. The obliging game setting allows to address this situation by taking α_W to require that all R_i hold infinitely often. Then a gracious strategy for player $\exists$ has to ensure α_S (possibly by avoiding, on most interactions, some R_i), but it also has to enable player $\forall$ to additionally realize α_W, thereby always enabling at least one "interesting" interaction on which all R_i and also all G_i hold infinitely often.

Previous approaches to the analysis of obliging games [6, 20] have been largely based on a reduction to equivalent non-obliging games in which the players still take single steps on the original game graph, but in addition to that, game nodes are annotated with auxiliary memory that is used to deal with the more involved obliging game's objectives. Obliging GR[1] games, as in the example above, have been considered under the names of "cooperative" [3] and "environmentally-friendly" [19]. Independent bespoke symbolic algorithms of slightly better complexity than the general solution have been suggested.

In this context, the contributions of the current work are threefold:

- We propose a novel perspective on obliging games that is based on considering multi-step strategies of players in the original game, rather than emulating single-step moves. In more detail, we provide an alternative reduction that takes obliging games to equivalent non-obliging games in which the system player commits to certain long-term behaviors, encoded by so-called witnesses, which are just plays of the original game (we therefore call the resulting games *witness games*). The environment player in turn can either check whether a given witness is valid, that is, satisfies both α_S and α_W, or accept the witness and exit it at any game node that occurs in the witness, thereby intuitively challenging the system player to still win when a play only traverses the witness up to the exit node and then continues on outside of the proposed witness; in the latter case, the system player has to provide a new witness for the challenged game node, and so on. We use the reduction to witness games to show determinacy of obliging games with Borel objectives (however, with respect to strategies of type $V^* \to V^\omega$ for the system player, and strategies of type $V^\omega \to V \cup V^\omega$ for the environment player).

- We show that witnesses for obliging games with ω-regular objectives α_S and α_W have finite representations, as they correspond to (accepting) runs of ω-automata with acceptance condition $\alpha_S \wedge \alpha_W$; we call such representations *certificates*. Using certificates in place of infinite witnesses, we modify witness games to obtain an alternative reduction that takes ω-regular obliging games to *finite ω-regular non-obliging games*, called *certificate games*. Technically, we present the reduction for obliging games with Emerson-Lei objectives. During the correctness proof for this reduction, we show that the memory requirements of graciously winning strategies for Emerson-Lei obliging games depend only linearly on the size of α_W, improving significantly upon existing upper bounds [6, 20] that are, in general, exponential in the size of α_W.
- The certificate games that we propose contain an explicit game node for any possible certificate within an obliging game. Hence they are prohibitively large and it is not viable to solve them naively. However, we show how a technique of fixpoint acceleration can be used to speed up the solving process of certificate games, replacing the exploration of all certificate nodes with emptiness checking for suitable ω-automata; this technique solves certificate games by computing nested fixpoints of a function that checks for the existence of suitable certificates. Thereby we are able to improve previous upper runtime bounds for the solution of obliging games; in many cases, our algorithm outperforms existing algorithms by an exponential factor.

Summing up, we propose a novel approach to the analysis of obliging games that provides more insight in their determinacy, and for obliging games with Emerson-Lei objectives, we significantly improve existing upper bounds both on strategy sizes and solution time.

Structure. We introduce obliging games and related notions in Section 2. In Section 3, we reduce obliging games to witness games and use the reduction to show that obliging games are determined (for strategies with additional information). Subsequently, we restrict our attention to ω-regular obliging games with Emerson-Lei objectives. In Section 4 we reduce witness games with such objectives to certificate games and use the latter to obtain improved upper bounds on strategy sizes in obliging games. In Section 5 we show how certificate games can be solved efficiently, in consequence improving previous upper bounds on the runtime complexity of solving obliging games. Full proofs and additional details can be found in an extended version of this paper.

2 Preliminaries

We start by recalling obliging games and extend the setup from previous work to use general Borel objectives in place of Muller objectives; we also introduce the special case of obliging games with Emerson-Lei objectives, and recall the definition of Emerson-Lei automata.

Arenas, plays, strategies. An *arena* is a graph $A = (V, V_\exists, E)$, consisting of a set V of *nodes* and a set $E \subseteq V \times V$ of *moves*; furthermore, we assume that the set of nodes is partitioned into the sets $V_\exists$ and $V_\forall := V \setminus V_\exists$ of nodes *owned* by player $\exists$ and by player $\forall$, respectively. We write $E(v) = \{w \in V \mid (v, w) \in E\}$ for the set of nodes reachable from node $v \in V$ by a single move. We assume without loss of generality that every node has at least one outgoing edge, that is, that $E(v) \neq \emptyset$ for all $v \in V$. A *play* $\pi = v_0 v_1 \ldots$ on A is a (finite or infinite) sequence of nodes such that $v_{i+1} \in E(v_i)$ for all $i \geq 0$. The length $|\pi| = n + 1$ of a finite play $\pi = v_0 v_1 \ldots v_n$ is the number of vertices it contains; throughout, we denote the set $\{0, \ldots, n\}$ for $n \in \mathbb{N}$ by $[n]$. By abuse of notation, we denote by A^ω the set of infinite plays on A and by A^* and A^+ the set of finite (nonempty) plays on A. A *strategy*

for player $i \in \{\exists, \forall\}$ is a function $\sigma : A^* \cdot V_i \to V$ that assigns a node $\sigma(\pi v) \in V$ to every finite play πv that ends in a node $v \in V_i$. A strategy σ for player i is said to be *positional* if the moves that it prescribes do not depend on previously visited game nodes. Formally this is the case if we have $\sigma(\pi v) = \sigma(\pi' v)$ for all $v \in V_i$ and all $\pi, \pi' \in A^*$. A play $\pi = v_0 v_1 \ldots$ is *compatible* with a strategy σ for player $i \in \{\exists, \forall\}$ if for all $j \geq 0$ such that $v_j \in V_i$, we have $v_{j+1} = \sigma(v_0 v_1 \ldots v_j)$.

Objectives and games.   In this work we consider two types of objectives: *Borel objectives* and *Emerson-Lei objectives*. Borel objectives are explicit sets of infinite sequences of vertices. A set is *Borel definable* if it is in the σ-algebra over the open subsets of infinite sequences of vertices V^ω. That is, sets that can be obtained by countable unions, countable intersections, and complementations from the open sets (sets of the form wV^ω for $w \in V^*$). A play π on A *satisfies* a Borel objective B if $\pi \in B$.

Emerson-Lei objectives are specified relative to a coloring function $\gamma_C : E \to 2^C$ (for some set C of colors) that assigns a set $\gamma_C(v, w) \subseteq C$ of colors to every move $(v, w) \in E$; we note that our setup is more succinct than the one from [6] where each edge has (at most) one color. A play $\pi = v_0 v_1 \ldots$ then induces a sequence $\gamma_C(\pi) = \gamma_C(v_0, v_1)\gamma_C(v_1, v_2) \ldots$ of sets of colors. Emerson-Lei objectives are given as positive Boolean formulas $\varphi_C \in \mathbb{B}_+(\{\mathsf{Inf}\ c, \mathsf{Fin}\ c \mid c \in C\})$ over atoms of the shape $\mathsf{Inf}\ c$ and $\mathsf{Fin}\ c$. Such formulas are interpreted over infinite sequences $\gamma_0 \gamma_1 \ldots$ of sets of colors. We put $\gamma_0 \gamma_1 \ldots \models \mathsf{Inf}\ c$ if and only if there are infinitely many positions i such that $c \in \gamma_i$, and $\gamma_0 \gamma_1 \ldots \models \mathsf{Fin}\ c$ if there are only finitely many such positions; satisfaction of Boolean operators is defined in the usual way. Then an infinite play π on A *satisfies* the formula φ_C if and only if $\gamma_C(\pi) \models \varphi_C$ and we define the Emerson-Lei objective induced by γ_C and φ_C by putting

$$\alpha_{\gamma_C, \varphi_C} = \{\pi \in A^\omega \mid \gamma_C(\pi) \models \varphi_C\}.$$

Parity objectives are a special case of Emerson-Lei objectives with set $C = \{p_0, \ldots, p_k\}$ of colors, where each edge has exactly one color (also called *priority*), and where $\varphi = \bigvee_{i\ \text{even}} \mathsf{Inf}\ p_i \wedge \bigwedge_{i < j \leq k} \mathsf{Fin}\ p_j$, stating that the maximal priority that is visited infinitely often has an even index. We can denote such objectives by just a single function $\Omega : E \to [k]$. Further standardly used conditions include *Büchi, generalized Büchi, generalized reactivity (GR[1]), Rabin* and *Streett* objectives, all of which are special cases of Emerson-Lei objectives (see e.g. [15]); the memory required for winning in such games has been investigated in [7].

We note that neither Borel nor Emerson-Lei objectives enable a simple distinction between finite plays ending in existential and universal nodes; hence we will avoid deadlocks in our game reductions, ensuring that all games in this work allow only infinite plays.

A *standard game* is a tuple (A, α), where $A = (V, V_\exists, E)$ is an arena and α is an objective. A strategy σ is *winning* for player $\exists$ at some node $v \in V$ if all plays that start at v and are compatible with σ satisfy the objective α. A strategy τ for player $\forall$ is defined winning dually.

An *obliging game* is a tuple (A, α_S, α_W), consisting of an arena $A = (V, V_\exists, E)$ together with two objectives α_S and α_W, called the *strong* and *weak* objective, respectively; we also refer to such games as α_S / α_W obliging games. A strategy σ for player $\exists$ is *graciously winning* for $v \in V$ if all plays that start at v and are compatible with σ satisfy the strong objective φ_S and furthermore every finite prefix $\pi \in A^*$ of a play that is compatible with σ can be extended to an infinite play $\pi\tau$ that is compatible with σ and satisfies the weak objective α_W. We sometimes refer to the infinite plays $\pi\tau$ witnessing the satisfaction of the weak objective as *obliging* plays. It follows immediately that player $\exists$ can only win graciously at nodes at which at least one obliging play (satisfying $\alpha_S \wedge \alpha_W$) starts, so we assume without loss of generality that this is the case for all nodes in V.

▶ **Example 1.** We consider the Emerson-Lei obliging game depicted below with the set $\{a, b, c, d\}$ of colors, a Streett condition (with two pairs (a, b) and (c, d)) as strong objective α_S, and generalized Büchi condition enforcing visiting both a and c as weak objective α_W. In all examples in this work, nodes belonging to player $\exists$ are depicted with rounded corners, while $\forall$-nodes are depicted by rectangles; edges may have several colors in Emerson-Lei games, but in this example, each edge has at most one color. We use the dashed edge to illustrate both winning and losing in an obliging game.

$$\alpha_S = (\mathsf{Inf}\ a \to \mathsf{Inf}\ b) \wedge (\mathsf{Inf}\ c \to \mathsf{Inf}\ d)$$
$$\alpha_W = \mathsf{Inf}\ a \wedge \mathsf{Inf}\ c$$
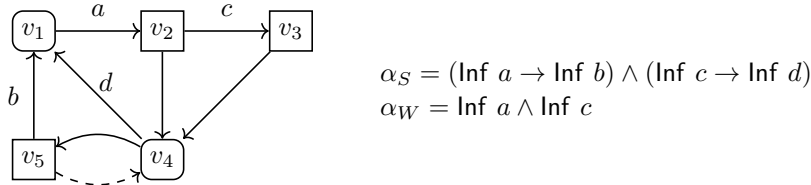
Consider the strategy σ with which player $\exists$ alternatingly moves to v_5 and to v_1 when node v_4 is reached, depending on where they moved from v_4 the last time it has been visited. Without the dashed edge, σ is graciously winning: every play compatible with σ visits the colors a, b and d infinitely often and hence satisfies α_S; also, σ allows player $\forall$ to visit color c arbitrarily often by moving from v_2 to v_3, so every finite prefix of a σ-play can be continued to an obliging σ-play that infinitely often visits v_3 and therefore satisfies $\alpha_S \wedge \alpha_W$. If the dashed edge is added to the arena, then σ is no longer graciously winning. Indeed, when playing against σ, player $\forall$ can prevent b from ever being visited by always moving from v_5 to v_4. However, the modified strategy σ' that moves from v_4 to v_1 only if the last visited node is *not* v_5 and also b has been visited more recently than d (and otherwise moves back to v_5) is graciously winning. Every σ'-play that ends in $(v_4 v_5)^\omega$ satisfies α_S but also can be made into an obliging play by having $\forall$ move to v_1 whenever v_5 is reached.

Emerson-Lei automata. Given an Emerson-Lei objective $\alpha_{\gamma_C, \varphi_C}$, an *Emerson-Lei automaton* is a tuple $A = (\Sigma, Q, \delta, q_0, \alpha_{\gamma_C, \varphi_C})$, where Σ is the alphabet, Q is a set of *states*, $\delta \subseteq Q \times \Sigma \times Q$ is the transition relation, and $q_0 \in Q$ is the initial state; in this context, we assume that $\gamma_C : \delta \to 2^C$ assigns sets of colors to transitions in A. A *run* of A on some infinite word $w = a_0 a_1 \ldots \in \Sigma^\omega$ is a sequence $\pi = q_0 q_1 \ldots \in Q^\omega$ such that $(q_i, a_i, q_{i+1}) \in \delta$ for all $i \geq 0$. A run π is *accepting* if and only if $\gamma_C(\pi) \models \varphi_C$, and A *recognizes* the language

$$L(A) = \{w \in \Sigma^\omega \mid \text{there is an accepting run of } A \text{ on } w\}.$$

An Emerson-Lei automaton A is *non-empty* if and only if $L(A) \neq \emptyset$. All automata that we consider in this work will have a single-letter alphabet $\Sigma = \{*\}$ so that they can read just the single infinite word $*^\omega$.

3 Determinacy of Obliging Games

Chatterjee et al. [6], when formalizing obliging games, stated that the standard shape of strategies (that is, $A^* \cdot V_i \to V$) does not allow player $\forall$ to counteract player $\exists$'s strategy with a single strategy. We show that, for a more general form of strategy (of type $A^\omega \to V \cup A^\omega$), this is possible. Furthermore, with these more general strategies, the games are determined. That is, players are not disadvantaged by revealing their strategy first and one of the players always has a winning strategy. This insight allows offering alternative (more efficient) solutions to obliging games in the next sections.

Fix an obliging game $G = (A, \alpha_S, \alpha_W)$. Let $A = (V, V_\exists, E)$ and let α_S and α_W be Borel sets. A *witness* for vertex $v \in V$ is an infinite play $\pi = v_0 v_1 \cdots \in A^\omega$ such that $v_0 = v$; we write $\mathsf{witness}(v)$ to denote the set of witnesses for $v \in V$. The *witness game* $W(G) = (A', \alpha')$ captures the obliging game by allowing player $\exists$ to choose an explicit witness for a given vertex. Player $\forall$ can then either check whether the witness satisfies both α_S and α_W, or stop at some point verifying the witness and ask for a new witness. If player $\forall$ changes witnesses infinitely often, they still check that the resulting play satisfies α_S.

Formally, we have $A' = (V', V'_\exists, E')$, where $V' = V \cup A^\omega$, $V'_\exists = V$ and E' is as follows.

$$
\begin{aligned}
E' \quad = \quad & \{(v, \pi) \mid v \in V, \pi \in \mathsf{witness}(v)\} \quad \cup \quad \{(v\pi, \pi) \mid v\pi \in A^\omega\} \quad \cup \\
& \{(v\pi, v'') \mid v\pi \in A^\omega, v \in V_\forall, v'' \in E(v)\}.
\end{aligned}
$$

Given $v\pi \in V'_\forall$ put $\mathsf{head}(v\pi) = v$. We extend head to sequences over V' in the natural way. Consider a play $\pi' \in (A')^\omega$. Let $\pi' \Downarrow_{V'_\forall}$ denote the projection of π' to the elements of $V'_\forall$, that is, $\pi' \Downarrow_{V'_\forall}$ is obtained from π' by removing all elements in $V'_\exists = V$. Then put $\mathsf{seq}_A(\pi') = \mathsf{head}(\pi' \Downarrow_{V'_\forall})$. Clearly, $\mathsf{seq}_A(\pi') \in A^\omega$, that is, $\mathsf{seq}_A(\pi')$ extracts from π' the infinite play in A that is followed in π'. The winning condition α' consists of plays π' that remain eventually in $V'_\forall$ forever such that $\mathsf{seq}(\pi')$ satisfies both α_S and α_W, or plays π' that visit $V'_\exists$ infinitely often such that $\mathsf{seq}(\pi')$ satisfies α_S. Formally, we have the following:

$$
\begin{aligned}
\alpha' \quad = \quad & \{\pi' \in V'^* \cdot (V'_\forall)^\omega \mid \mathsf{seq}_A(\pi') \in \alpha_S \cap \alpha_W\} \quad \cup \\
& \{\pi' \in V'^* \cdot (V'_\exists \cdot (V'_\forall)^*)^\omega \mid \mathsf{seq}_A(\pi') \in \alpha_S\}
\end{aligned}
$$

▶ **Example 2.** For brevity, we refrain from showing the complete witness game associated to the obliging game from Example 1 and instead consider just two witnesses for the node v_1, namely $(v_1 v_2 v_4 v_5)^\omega$ and $(v_1 v_2 v_3 v_4 v_1 v_2 v_4 v_5)^\omega$. The former satisfies the Streett objective α_S as it visits the colors a and b. However, it does not satisfy the generalized Büchi objective α_W as color c is not visited infinitely often. The latter witness, contrarily, visits all colors infinitely often and hence satisfies $\alpha_S \wedge \alpha_W$. In the witness game, player $\exists$ can move from v_1 to these two witnesses (and to many more). Player $\forall$ in turn can win the first witness by exploring it indefinitely, thereby showing that it does not satisfy α_W; doing the same for the second witness, player $\forall$ loses. In both certificates, we have exit moves from every position such that the node at the position is owned by player $\forall$, that is, moves from positions with node v_2 to $E(v_2)$, and similarly for v_3 and v_5.

▶ **Lemma 3.** *Player $\exists$ is graciously winning in G at v iff player $\exists$ is winning in $W(G)$ at v.*

▶ **Corollary 4.** *Obliging games are determined.*

Proof. This follows immediately from Lemma 3 and the determinacy of games with Borel winning conditions [21]. Notice that Martin's determinacy result holds also for games with continuous vertex-spaces and continuous branching degrees, as in the witness game. ◀

4 Reducing ω-Regular Obliging Games to Finite Games

From this point on, we restrict our attention to obliging games in which both objectives are Emerson-Lei objectives; we note that every ω-regular objective can be transformed to an Emerson-Lei objective. It turns out that due to the ω-regularity of Emerson-Lei objectives, obliging plays in such games have finite witnesses that take on the form of lassos that are built over the game arena at hand. Formally, we show that for every such game we can create a game that is smaller than the witness game by restricting attention to witnesses of this specific form that satisfy the acceptance conditions. We call such witnesses *certificates*, which we define next.

4.1   Certificates from Witnesses

We fix an Emerson-Lei obliging game $G = (A, \alpha_S, \alpha_W)$ with arena $A = (V, V_\exists, E)$, objectives $\alpha_S = (\gamma_S, \varphi_S)$ and $\alpha_W = (\gamma_W, \varphi_W)$, and put $n := |V|$, $d := |S|$ and $k := |W|$.

▶ **Definition 5** (Certificate). *Given a node $v \in V$, a* certificate *for v (in G) is a witness for v that is of the form $c = wu^\omega$; if c satisfies $\varphi_S \wedge \varphi_W$, then we say that c is a* valid certificate.

We equally represent $c = wu^\omega$ by the pair $c = (w, u)$. Let $w = w_0 w_1 \ldots w_m$ and $u = u_0 u_1 \ldots u_r$. We refer to w as the *stem* and to u as the *loop* of c, and to m and r as the length of the stem and the loop, respectively. When the partition of c is not important we sometimes just write $c = v_0 \ldots v_{m+r+1}$. Clearly, as satisfaction of Emerson-Lei objectives depends only on the infinite suffixes of a play, it follows that in a valid certificate, u^ω also satisfies φ_S and φ_W.

Given a coloring function γ_C over some set C of colors, the *C-fingerprint* of a finite play $\pi = v_0 v_1 \ldots v_j \in A^*$ is the set $\bigcup_{0 \le i < j} \gamma_C(v_i, v_{i+1})$ of colors visited by π, according to γ_C.

Next we show that given a witness in G that satisfies $\alpha_S \wedge \alpha_W$, we can alway construct a valid certificate of size at most $\mathsf{certLen} := n \cdot d + (d + k + 1) \cdot (n + 1) \in \mathcal{O}(n \cdot (\max(d, k)))$.

▶ **Lemma 6** (Certificate existence). *Let $v \in V$ and let $\pi = \pi_0 \pi_1 \ldots$ be a witness for $\pi_0 = v$ that satisfies $\varphi_S \wedge \varphi_W$. Then there is a valid certificate $c = (w, u)$ for v with stem length at most $n \cdot d$ and loop length at most $(d + k + 1) \cdot (n + 1)$, such that for all positions i in c there is a position j in π such that $v_i = \pi_j$ and the S-fingerprints of $v_0 \ldots v_i$ and $\pi_0 \ldots \pi_j$ coincide.*

We let $\mathsf{Cert}(v)$ and Cert denote the sets of all valid certificates for some $v \in V$ in G and all valid certificates for all $v \in V$ in G, respectively, subject to the size bounds obtained in Lemma 6. Then we have $|\mathsf{Cert}(v)| \le |\mathsf{Cert}| \le n^{\mathsf{certLen}} \in 2^{\mathcal{O}(n \cdot (\max(d, k)) \cdot \log n)}$.

4.2   Certificate Games and Smaller Winning Strategies

Next we adapt the witness games from Section 3 to use finite certificates in place of witnesses, which leads to *certificate games* that have finite game arenas. In a certificate game, player $\exists$ has to pick a single valid certificate at each node $v \in V$, thereby commiting to a long-term future behavior that satisfies $\alpha_S \wedge \alpha_W$. Player $\forall$ in turn can either accept the certificate (and thereby lose the play), or pick some position i in the certificate and challenge whether player $\exists$ still has a strategy to graciously win when player $\forall$ exits the certificate at position i. All plays then either end with player $\forall$ eventually losing by accepting some certificate, or player $\forall$ infinitely often exits certificates, in which case the winner of the play is determined using just the strong objective α_S.

Formally, the certificate game associated to G is a (non-obliging) Emerson-Lei game $C(G) = (B, \beta)$ with arena $B = (N, N_\exists, R)$. The game is played over the sets $N_\exists = V$ and $N_\forall = \mathsf{Cert} \cup \mathsf{Cert} \times [\mathsf{certLen}] \times V$ of nodes. The moves in $C(G)$ are defined by

$$R(v) = \mathsf{Cert}(v) \qquad R(c) = \{(c, i, v) \in N_\forall \mid v_i \in V_\forall, v \in E(v_i)\} \cup \{c\} \qquad R(c, i, v) = \{v\}$$

for $v \in V$, $c = v_0 v_1 \ldots v_m \in \mathsf{Cert}$ and $i \in [\mathsf{certLen}]$. Thus player $\exists$ has to provide a valid certificate for v whenever a play reaches a node $v \in V$. Player $\forall$ in turn can challenge the way certificates are combined by exiting them from universal nodes in their stem or loop, and continuing the game at the exit node; if a certificate c does not contain a universal node, then player $\forall$ has to take the loop at c, intuitively accepting the certificate c. Intermediate nodes (c, i, v) are used to make explicit the S-fingerprint of the path through a certificate c

that is taken before exiting it by moving from v_i to v; this is necessary since a certificate may contain universal nodes that allow moving from different positions in the certificate to a single exit node v, potentially giving player $\forall$ a choice on the path that is taken (and on the S-fingerprint that is accumulated) through the certificate before exiting to v.

As our notion of games does not support deadlocks, we annotate trivial loops with an additional color $c_\top$ to encode the situation that player $\forall$ accepts a certificate and thereby loses. The coloring function $\gamma' : R \to 2^{S \cup \{c_\top\}}$ also keeps track of the S-fingerprints when passing through certificates and is defined by $\gamma'(c, (c, i, v)) = \gamma_S(v_0 \ldots v_i v)$, $\gamma'(v, c) = \emptyset$ and $\gamma'((c, i, v), v) = \emptyset$ for the non-looping moves and by $\gamma'(c, c) = c_\top$ for looping moves at certificates $c \in \mathsf{Cert}$. We then define $\varphi' = \mathsf{Inf}(c_\top) \vee \varphi_S$, intuitively expressing that player $\exists$ can win either according to φ_S or by forcing player $\forall$ to get stuck in a trivial loop. Then we define β to be the objective induced by φ' and γ'.

▶ **Example 7.** Below we depict the construction for a single node $v \in V$, showing, as examples, just two valid certificates $c_1 = v_0 v_1 v_2 \in \mathsf{Cert}(v)$ and $c_2 = w_0 w_1 w_2 w_3 \in \mathsf{Cert}(v)$ (in particular, we assume $v_0 = w_0 = v$ and that the certificates c_1 and c_2 satisfy both φ_S and φ_W). We further assume that $v_2 \in V_\forall$, $E(v_2) = \{v_1, x\}$, and that $w_1 = w_3 \in V_\forall$ and $E(w_1) = E(w_3) = \{w_2\}$; all other nodes in c_1 and c_2 are assumed to be contained in $V_\exists$.
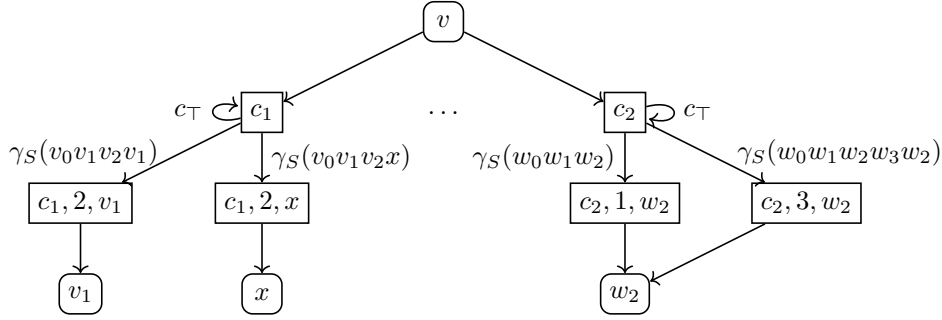
At node v, player $\exists$ has to provide some valid certificate for v; assuming that player $\exists$ picks the certificate $c_1 = v_0 v_1 v_2$, player $\forall$ in turn can exit the certificate at position 2 (as $v_2 \in V_\forall$) by moving to either $(c_1, 2, v_1)$ or $(c_1, 2, x)$ (as $E(v_2) = \{v_1, x\}$) thereby triggering S-fingerprint $\gamma_S(v_0 v_1 v_2 v_1)$ or $\gamma_S(v_0 v_1 v_2 x)$; player $\forall$ intuitively cooperates during the play that leads from v_0 to v_2, but may stop cooperating at node v_2 by moving to either v_1 or x. Similarly, player $\forall$ can exit the certificate c_2 at positions 1 or 3, both leading to the node w_2, but with different S-fingerprints, that is, with different sets of visited colors.

As the above example shows, certificate games have a three-stepped structure. Plays progress from nodes v of the original game on to certificate nodes c, and then onwards to nodes (c, i, v') that encode the part of c that is visited before exiting c, and then proceed back to some node v' of the original game. We refer to these three-step subgames a *gadgets*; each gadget has a starting node v and a (possibly empty) set of exiting nodes. We point out that, crucially, gadgets do not contain (non-trivial) loops.

We state the correctness of the reduction to certificate games as follows.

▶ **Theorem 8.** *Let G be an Emerson-Lei obliging game and let v be a node in G, and recall that $W(G)$ and $C(G)$ respectively are the witness game and the certificate game associated with G. The following are equivalent:*

1. *player $\exists$ graciously wins v in G;*
2. *player $\exists$ wins v in $W(G)$;*
3. *player $\exists$ wins v in $C(G)$.*

Proof. The equivalence of the first two items is stated by Lemma 3 and the proof of the implication from item two to item three is technical but straight-forward. Therefore we show just the implication from item three to item one, which also shows how to construct a graciously winning strategy in G from a winning strategy in $C(G)$. For this proof, we use *strategies with memory*, introduced next. A strategy for player $i \in \{\exists, \forall\}$ *with memory* M is a tuple $\sigma = (M, m_0, \mathsf{update} : M \times E \to M, \mathsf{move} : V_i \times M \to V)$, where M is some set of *memory values*, $m_0 \in M$ is the initial memory value, the *update function* update assigns the outcome $\mathsf{update}(m, e) \in M$ of updating the memory value $m \in M$ according to the effects of taking the move $e \in E$, and the *moving function* move prescribes a single move $(v, \mathsf{move}(v, m)) \in E$ to every game node $v \in V_i$ that is owned by player i, depending on the memory value m. We extend update to finite plays π by putting $\mathsf{update}(m, \pi) = m$ in the base case that π consists of a single node, and by putting $\mathsf{update}(m, \pi) = \mathsf{update}(\mathsf{update}(m, \tau), (v, w))$ if π is of the shape τvw, that is, contains at least two nodes.

Let $\sigma = (M, m_0, \mathsf{update}, \mathsf{move})$ be a winning strategy with memory M for player $\exists$ in the game $C(G)$. We construct a strategy $\tau = (M', (m_0, v, 1), \mathsf{update}', \mathsf{move}')$ with memory $M' = V \times M \times \{1, \ldots, 2|S| + |W|\}$ for player $\exists$ in the game G such that τ graciously wins v. To this end, we note that σ provides, for each node $w \in V$ and memory value $m \in M$, a certificate $c(w, m) := \mathsf{move}(w_\exists, m) \in \mathsf{Cert}(w)$ for w in G.

The strategy τ uses memory values (w, m, i), where w and m identify a current certificate $c(w, m)$ and i is a counter used for the construction of this certificate. We define τ such that player $\exists$ starts by building the certificate $c(v, m_0)$ for v and m_0. This process continues as long as player $\forall$ obliges, that is, does not move outside of this certificate. Assuming that player $\forall$ obliges, the memory required to construct the certificate is bounded by $2|S| + W$, walking, in the prescribed order, through the certificate. In the case that player $\forall$ eventually stops obliging and takes a move to some node w that is not the next node on the path prescribed by the certificate, the memory for the strong objective is updated according to the play from v to w, resulting in a new memory value m, and the memory for certificate construction is reset. Then the certificate construction starts again, this time for the certificate $c(w, m)$ prescribed by σ for the new starting values w and m.

In more detail, every node from V is visited at most $2|S| + |W|$ times within a certificate $c(v, m) = (v_0 v_1 \ldots v_n)$ before the end of the loop v_n is reached. To each position i within $c(v, m)$, we associate the number $\sharp i$ such that v_i is the $\sharp i$-th occurence of the node v_i in $c(v, m)$; we note that for all $1 \le i \le n$, we have $1 \le \sharp i \le 2|S| + |W|$. Conversely, given a node w and a number j between 1 and $2|S| + |W|$, we let $\mathsf{pos}(w, j)$ denote the position of the j-th occurence of w in $c(v, m)$. In the case w occurs less than j times in $c(v, m)$, we leave $\mathsf{pos}(w, j)$ undefined; below we make sure that this value is always defined when it is used.

Let j be the starting position of the loop of $c(v, m)$. Given a position i in $c(v, m)$, we abuse notation to let $i + 1$ denote just $i + 1$ if $i < n$, and to denote j if $i = n$; in this way, we encode taking single steps within a certificate, wrapping back to the start of the certificate loop, once the end of the loop is reached.

We define the strategy τ to always move to the next node in the current certificate, using the memory value i together with the current node w to find the current position in the certificate $c(v, m)$, that is, we put

$$\mathsf{move}'(w, (v, m, i)) = v_{\mathsf{pos}(w, i) + 1}$$

for $w \in V_\exists$ and $(v, m, i) \in M'$. The memory update in τ incorporates the memory update function from σ, but additionally also keeps track of the memory for certificate construction. For moves (w, w') that stay within the current certificate (that is, $w' = v_{\mathsf{pos}(w, i) + 1}$), this

memory is updated according to proceeding one step within the certificate; for moves that leave the current certificate, the memory for certificate construction is reset while the memory for σ is updated according to the path taken through the certificate before exiting it. Formally, we put

$$\mathsf{update}'((v, m, i), (w, w')) = \begin{cases} (v, m, \sharp(\mathsf{pos}(w, i) + 1)) & w' = v_{\mathsf{pos}(w,i)+1} \\ (w', \mathsf{update}(m, (v_0 v_1 \dots w w')), 1) & w' \neq v_{\mathsf{pos}(w,i)+1} \end{cases}$$

for $(v, m, i) \in M'$ such that $c(v, m) = v_0 v_1 \dots v_n$ and $(w, w') \in E$; notice that in plays that adhere to τ, the latter case can, by definition of move', only happen for $w \in V_\forall$.

To see that player $\exists$ graciously wins v using the constructed strategy τ, let π be a play that adheres to τ. Then π either eventually stays within one certificate $c(w, m)$ forever, or π induces an infinite play ρ of $C(G)$ that adheres to σ. In the latter case, π changes the certificate infinitely often, and the S-fingerprints of π and ρ coincide by construction, showing that π satisfies φ_S; if π eventually stays within one certificate $c(w, m)$ forever, we note that the loop of $c(w, m)$ satisfies φ_S so that π satisfies φ_S as well. Thus every play that adheres to τ satisfies the strong objective φ_S. Next, let π_f be a finite prefix of some play that adheres to τ, and let π_f end in some node w. We have to show that there is an infinite play π that adheres to τ, extends π_f and satisfies φ_W. Let m be the value of the memory for the strong objective at the end of π_f. We consider the play of G in which player $\forall$ obliges from the end of π_f on, forever. By construction, this play is π_f extended with the certificate $c(w, m)$ which adheres to τ and satisfies φ_W. ◀

The strategy construction given in the proof of Theorem 8 yields:

▶ **Corollary 9.** *Given an obliging game with Emerson-Lei objectives α_S and α_W and n nodes that contains a node v at which $\exists$ is graciously winning, there is a graciously winning strategy for v that uses memory at most $n \cdot (2|S| + |W|) \cdot m$, where m is the amount of memory required by winning strategies for player $\exists$ in standard games with objective α_S.*

▶ **Remark 10** (Canonical certificates). With the proposed strategy extraction, certificate strategies memorize the starting point of the certificate that player $\exists$ currently attempts to construct, leading to an additional linear factor n in strategy size. We conjecture that winning strategies in certificate games can be transformed to make them *reuse* certificates (so that the choice of the certificate does not depend on the starting point). Strategies with such canonical certificate choices would allow for removing the additional factor n in Corollary 9.

In particular, our result shows the existence of graciously winning strategies with quadratic sized strategies for all obliging games that have a half-positional strong objective (i.e. one for which standard games have positional winning strategies for player $\exists$); this covers, e.g., obliging games in which α_S is a parity or Rabin objective.

In the table below we compare the solution via certificate games to a previous solution method [6, 20] reduces α_S / α_W obliging games to standard games with an objective of type $\alpha_S \wedge$ Büchi. In this approach, the weak objective is encoded by a non-deterministic Büchi automaton and graciously winning strategies obtained in this way incorporate the state space of the automaton. The encoding of α_W by a Büchi automaton leads to linear blow-up if α_W is a Rabin objective, but is exponential if α_W is a Streett or general Emerson-Lei objective.

In contrast to this, the certificate games that we propose here always have (essentially) just α_S as objective, and the dependence of strategy size on the weak objective α_W is in all cases linear in $|W|$. In comparison to [6], our approach hence leads to significantly smaller

▪ **Table 1** Comparison of upper bounds on strategy sizes for various types of obliging games.

type of α_S	type of α_W	objective red. [6]	strategy size [6]	strategy size
parity(d)	Rabin(k)	parity(d)$\wedge$ Büchi	$d(4k+2)$	$(2d+2k)n$
	Streett(k)		$2^{k+1}dk$	$(2d+2k)n$
	EL(k)		$2^{k+1}dk$	$(2d+k)n$
Rabin(d)	Rabin(k)	Rabin(d)$\wedge$ Büchi	$d(4k+2)$	$(4d+2k)n$
	Streett(k)		$2^{k+1}dk$	$(4d+2k)n$
	EL(k)		$2^{k+1}dk$	$(4d+k)n$
Streett(d)	Rabin(k)	Streett($d+1$)	$(d+1)!(4k+2)$	$(4d+2k)d!n$
	Streett(k)		$(d+1)!2^{k+2}k$	$(4d+2k)d!n$
	EL(k)		$(d+1)!2^{k+2}k$	$(4d+k)d!n$
EL(d)	EL(k)	EL($d+1$)	$(d+1)!2^{k+2}k$	$(2d+k)d!n$

strategies for obliging games in which α_W is at least a Streett objective. In the cases where α_W is a Rabin objective, strategies obtained from certificate games are slightly larger than in [6]; we conjecture that this can be improved by sharing certificates (cf. Remark 10).

For instance, for obliging games with $\alpha_S = \text{Rabin}(d)$ and $\alpha_W = \text{Streett}(k)$, the approach from [6] uses nondeterministic Büchi automata with $2^k k$ states to encode the weak (Streett) objective. The reduction leads to a game with objective Rabin(d)$\wedge$ Büchi; winning strategies in such games require $2d$ additional memory values for each automaton state (cf. [7]), resulting in an overall memory requirement of $2^{k+1}dk$. In contrast, the reduced certificate game in this case is a Rabin(d) game, and the extracted gracious strategies require memory only to identify the current certificate and a position in it; the overall memory requirement for strategies obtained through our approach thus is $(4d+2k)n$.

5 Solving Certificate Games Efficiently

In the previous section we have shown how obliging games with Emerson-Lei objectives α_S and α_W can be reduced to certificate games. This reduction not only yields games with a simpler objective (essentially just α_S) than in the previously known alternative reduction from [6], but it also shows that the memory required for graciously winning strategies in obliging games always depends only linearly on $|W|$. However, the proposed reduction to certificate games makes explicit all possible certificates that exist in the original obliging game and therefore incurs exponential blowup. In more detail, given an Emerson-Lei obliging game G with n nodes and sets S and W of colors, the certificate game $C(G)$ from the previous section is of size $2^{\mathcal{O}(n \cdot \max(|S|,|W|) \cdot \log n)}$ and uses $|S|$ many colors; solving it naively does not improve upon previously known solution algorithms for obliging games.

In this section, we show that the gadget constructions in certificate games essentially encode non-emptiness checking for non-deterministic ω-automata; intuitively, a certificate for a game node is a witness for the non-emptiness of an Emerson-Lei automaton with acceptance condition $\alpha_S \wedge \alpha_W$ that lives over (parts of) the original game arena. We show that it suffices to check for the existence of a single suitable certificate, rather than exploring all possible certificates that occur as explicit nodes in the reduced game.

As pointed out above, the gadget parts in a certificate game do not have non-trivial cycles (that is, we can regard them as directed acyclic graphs, DAGs). Consequently, it is possible to simplify the solution process of these (potentially) exponential-sized parts of the game by instead using non-emptiness checking for suitable ω-automata. If for instance both α_S and α_W are Streett conditions (so that $\alpha_S \wedge \alpha_W$ again is a Streett condition), then the

solution of every gadget part in the game can be reduced to non-emptiness checking of Streett automata. As non-emptiness checking for Streett automata can be done in polynomial time, this trick in effect removes the exponential runtime factor that originates from the large number of certificates when solving such games (and in general, whenever non-emptiness of $\alpha_S \wedge \alpha_W$-automata can be checked efficiently).

Our program is as follows: We first show how Emerson-Lei certificate games can be reduced to parity games using a tailored later-appearance-record (LAR) construction, incurring blow-up $|S|!$ in game size, but retaining the DAG structure of gadget subgames. Importantly, this is required to retain the dependence of non-DAG nodes on a small number of (post DAG) successors. Then we show the relation between attractor computation in gadget subgames within the obtained parity games and the non-emptiness problem for specific Emerson-Lei automata. Finally, we apply a fixpoint acceleration method from [14] to show that during the solution of parity games, the solution of DAG substructures can be replaced with a procedure that decides attraction to (subsets of) the exit nodes of the DAG. Overall, we therefore show that the winning regions of certificate games can be computed as nested fixpoints of a function that checks certificate existence by nonemptiness checking suitable Emerson-Lei automata.

5.1   Lazy parity transform

We intend to transform $C(G)$ to an equivalent parity game, using a lazy variant of the later-appearance-record (LAR) construction (cf. [12, 17]). To this end, we fix a set C of colors and introduce notation for permutations over C. We let $\Pi(C)$ denote the set of permutations over C, and for a permutation $\pi \in \Pi(C)$ and a position $1 \leq i \leq |C|$, we let $\pi(i) \in C$ denote the element at the i-th position of π. For $D \subseteq C$ and $\pi \in \Pi(C)$, we let $\pi@D$ denote the permutation that is obtained from π by moving *the* element of D that occurs at the right-most position in π to the front of π; for instance, for $C = \{a, b, c, d\}$ and $\pi = (a, d, c, b) \in \Pi(C)$, we have $\pi@\{a, d\} = \pi@\{d\} = (d, a, c, b)$ and $(d, a, c, b)@\{a, d\} = \pi$. Crucially, restricting the reordering to single colors, rather than sets of colors, ensures that for each $\pi \in \Pi(C)$ and all $D \subseteq C$, there are only $|C|$ many π' such that $\pi@D = \pi'$. Given a permutation $\pi \in \Pi(C)$ and an index $1 \leq i \leq |C|$, we furthermore let $\pi[i]$ denote the set of colors that occur in one of the first i positions in π.

Next we show how permutations over C can be used to transform Emerson-Lei games with set C of colors to parity games; the reduction annotates nodes from the original game with permutations that serve as a memory, encoding the order in which colors have recently been visited. The transformation is lazy as it just moves the most significant color that is visited by a set of colors C rather than the entire set.

▶ **Definition 11.** *Let $G = (A, \alpha_C)$ be an Emerson-Lei game with arena $A = (V, V_\exists, E)$, set C of colors and objective α_C induced by γ_C and φ_C. We define the parity game*

$$P(G) = (V \times \Pi(C), V_\exists \times \Pi(C), E', \Omega : E' \to \{1, \dots, 2|C| + 1\})$$

by putting $E'(v, \pi) = \{(w, \pi@\gamma_C(v, w)) \mid (v, w) \in E\}$ for $(v, \pi) \in V \times \Pi(C)$, as well as $\Omega((v, \pi), (w, \pi')) = 2p$ if $\pi[p] \models \varphi_C$ and $\Omega((v, \pi), (w, \pi')) = 2p + 1$ if $\pi[p] \not\models \varphi_C$, for $((v, \pi), (w, \pi')) \in E'$. In the definition of $\Omega((v, \pi), (w, \pi'))$, we write p to denote the right-most position in π that contains some color from $\gamma_C(v, w)$. For a finite play $\tau = (v_0, \pi_0)(v_1, \pi_1) \dots (v_n, \pi_n)$ of $P(G)$, we let $p(\tau)$ denote the right-most position in π_0 such that $\pi_0(p(\tau)) \in \gamma_C(v_0 v_1 \dots v_n)$. The game $P(G)$ has $|V| \cdot |C|!$ nodes and $2|C| + 1$ priorities.

▶ **Lemma 12.** *For all $v \in V$ and $\pi \in \Pi(C)$, player $\exists$ wins from v in G iff player $\exists$ wins from (v, π) in $P(G)$.*

Now we consider the parity game $P(C(G))$ that is obtained by applying the above LAR construction to the certificate game $C(G)$ from Section 4. It is a parity game with $2^{\mathcal{O}(n \cdot \max(|S|, |W|) \cdot \log n)} \cdot |S|!$ many nodes and $2|S| + 1$ priorities. We recall that all nodes in $C(G)$ that are of the shape c or (c, i, v) such that $c, (c, i, v) \in N_\forall$ are inner nodes of gadget subgames that consist of three layers. Each such subgame has exactly one entry node and at most n exit nodes, all contained in $N_\exists = V$. The LAR construction preserves this general structure as it simply annotates game nodes with permutations of colors. Specifically, $P(G)$ has $|S|! \cdot n$ entry and exit nodes for all such subgames together. Furthermore, for all entry nodes (v, π) of a subgame in $P(G)$, we have that every exit node (that can be reached from (v, π) with excactly three moves, not accounting for trivial loops) is of the shape $(w, \pi@i)$ where $w \in V$ and $0 \leq i \leq |C|$. While an entry node for an individual subgame in $C(G)$ has at most n exit nodes, each entry node for a subgame in $P(C(G))$ has at most $n(|S|+1)$ exit nodes. Using the classical LAR would result in a potentially exponential number of exit nodes. Indeed, for every possible subset $C' \subseteq C$ there could be a different exit node.

▶ **Example 13.** Below, we depict (part of) the parity game that is obtained from the certificate game fom Example 7 by using the proposed LAR construction.
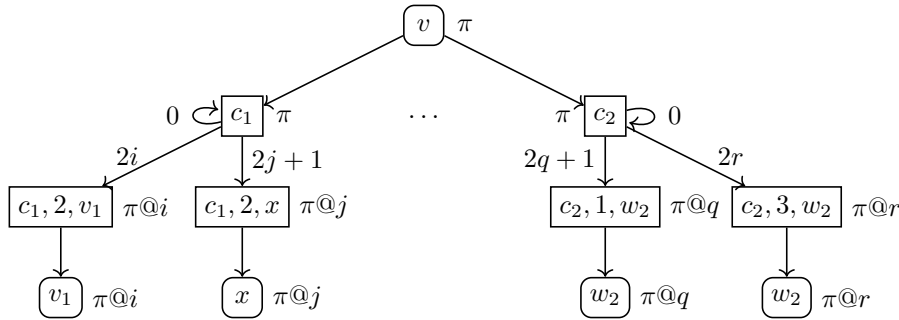
Here $i = p(v_0 v_1 v_2 v_1)$, $j = p(v_0 v_1 v_2 x)$, $q = p(w_0 w_1 w_2)$, $r = p(w_0 w_1 w_2 w_3 w_2)$ and we assume that the sets $\pi[i]$ and $\pi[r]$ satisfy φ_S and that the sets $\pi[j]$ and $\pi[q]$ do not satisfy φ_S, leading to the respective even and odd priorities. In particular, we have $2q + 1 \neq 2r$ so that $(w_2, \pi@q)$ and $(w_2, \pi@r)$ are two distinct exit nodes from the certificate c_2. Intuitively they correspond to two different paths through c_2 with different S-fingerprints; while in the Emerson-Lei game $C(G)$, the different fingerprints are dealt with by signalling different sets of colors, in the parity game $P(C(G))$, the two paths have different effects on the later-appearence memory π and thus lead to different outcome nodes. The self-loops at nodes c_1 and c_2 correspond to player $\forall$ giving up, so we assign priority 0 to these moves.

5.2 Solving parity games using DAG attraction

A standard way of solving parity games is by computing a nested fixpoint of a function that encodes one-step attraction in the game; the domain of this fixpoint computation then is the set of all game nodes. For parity games that contain cycle-free parts (DAGs), this process can be improved by instead computing a nested fixpoint of a function that encodes multi-step attraction along the DAG parts of the game. The domain of the latter fixpoint computation then does not contain the internal nodes of the DAG parts, which leads to accelerated fixpoint stabilization. We formalize this idea as follows.

▶ **Definition 14** (DAGs in games). *Let $G = (A, \Omega)$ be a parity game with $A = (V, V_\exists, E)$ and $k+1$ priorities $0, \dots, k$. We refer to a set $W \subseteq V$ of nodes as a DAG (directed acyclic graph) if it does not contain an E-cycle; then there is no play of G that eventually stays within W forever. A DAG need not be connected, that is, it may consist of several cycle-free subgames of G. Given a DAG $W \subseteq V$, we write $V' = V \setminus W$ and refer to the set V' as real nodes (with respect to W). A DAG is positional if for each existential node $w \in W \cap V_\exists$ in it, there is exactly one real node $v \in V'$ from which w is reachable without visiting other real nodes.*

▶ **Definition 15** (DAG attraction). *Given a DAG W and $k+1$ sets $\overline{V} = (V_0, \dots, V_k)$ of real nodes and a real node $v \in V'$, we say that player $\exists$ can attract to $\overline{V}$ from v within W if they have a strategy σ such that for all plays π that start at v and adhere to σ, the first real node v' in π such that $v' \neq v$ is contained in V_p, where p is the maximal priority that is visited by the part of π that leads from v to v'. Given a dag W, we define the dag attractor function $\mathsf{DAttr}_\exists^W :$ $\mathcal{P}(V')^k \to \mathcal{P}(V')$ by $\mathsf{DAttr}_\exists^W(\overline{V}) = \{v \in V' \mid$ player $\exists$ can attract to $\overline{V}$ from v within $W\}$ for $\overline{V} = (V_0, \dots, V_k) \in \mathcal{P}(V')^k$. We denote by $t_{\mathsf{Attr}_\exists^W}$ the time required to compute, for every input $\overline{V} \in \mathcal{P}(V')^k$, the dag attractor of $\overline{V}$ through W.*

▶ **Remark 16.** The sets V_i in the above definition correspond to valuations of fixpoint variables in the nested fixpoint computation that is used by the fixpoint acceleration method in Lemma 17 below to solve games via DAG attraction. These sets monotonically increase or decrease during the solution process and at each point of the solution process, a set V_i intuitively holds game nodes for which it currently is assumed that player $\exists$ wins if they can force the game to reach a node from V_i via a partial play in which the maximal priority is i.

During the computation of the DAG attractor to a tuple $\overline{V} = (V_0, \dots, V_k)$, we therefore intuitively consider the argument nodes $\overline{V}$ to be *safe* in the sense that in order to win from a node v, is suffices that existential player has a strategy that ensures that every partial play through the DAG exits it to a node from V_i, where i the maximal priority visited by that play along the DAG. Thus if player $\exists$ can win from all nodes in $\overline{V}$, then they can win from all nodes in $\mathsf{DAttr}_\exists^W(\overline{V})$.

In our case, from a node v, a strategy σ corresponds to choosing one certificate. Then, player $\forall$ can attract to all successors of $\forall$-nodes on the certificate. The path through the certificate to this $\forall$-node and to its successor outside the certificate shows a certain priority j that implies the successor must be in the set V_j.

▶ **Lemma 17** ([13]). *Let G be a parity game with priorities 0 to k and set V of nodes, let W be a positional DAG in G, and let $n = |V|$ and $m = n - |W|$. Then G can be solved with $\mathcal{O}(m^{\log k + 1})$ computations of a DAG attractor; if $k + 1 < \log m$, then G can be solved with a number of DAG attractor computations that is polynomial in m (specifically: in $\mathcal{O}(m^5)$).*

We always have $t_{\mathsf{Attr}_\exists^W} \leq |E|$, using a least fixpoint computation to check for alternating reachability, thereby possibly exploring all DAG edges of the game. However, in the case that $m < \log n$ and $t_{\mathsf{Attr}_\exists^W} \in \mathcal{O}(\log n)$ (that is, when most of the game nodes are part of a DAG, and DAG attractability can be decided without exploring most of the DAG nodes), Lemma 17 enables exponentially faster game solving.

5.3   Checking certificate existence efficiently

The parity game $P(C(G))$ that is obtained by applying the LAR construction from Subsection 5.1 to the certificate game $C(G)$ has $2^{O(n \cdot \max(d,k) \cdot \log n)} \cdot |S|!$ nodes, but only $|S|!n$ of these are real nodes: all certificate nodes are internal nodes in a gadget that has a DAG structure. In this section, we show that DAG attractors in $P(C(G))$ can be computed efficiently relying on non-emptiness checking of suitable Emerson-Lei automata.

In $P(C(G))$, the DAG attractor to a tuple $\overline{V} = (V_1, \ldots, V_{2|C|+1})$ consists of the nodes (v, π) such that there is a valid certificate starting at v such that all exits points of the cerficate are safe in the sense of Remark 16, that is, exiting the certificate with fingerprint i is only possible to nodes $w \in V_i$; we refer to such certificates as *valid and safe.*

Next we show how the existence of valid and safe certificates can efficiently be checked by using non-emptiness checking for Emerson-Lei automata to find valid and safe certificate loops that reside over single sets V_i (as the fingerprint does not increase within certificate loops, by the construction of certificates as in the proof of Lemma 6), and then using a reachability analysis that keeps track of fingerprints to compute safe stems leading (with fingerprint i) to some loop over V_i. In more detail, we check for the existence of valid and safe certificates as follows, fixing a permutation π and a tuple $(V_1, \ldots, V_{2|C|+1})$, where each V_i is a set of real nodes in $P(C(G))$.

- *Make the fingerprints explicit.* Define the graph $M = (V \times [2|C| + 1], R)$ as follows. Vertices of the graph are pairs (v, i) consisting of a game node $v \in V$ and a priority $i \in [2|C| + 1]$, intuitively encoding the largest priority that has been visited since a DAG has been entered; edges in this graph correspond to game moves but also update the priority value according to visited priorities, that is, R contains exactly the edges $((v, i), (w, j))$ such that $w \in E(v)$ and $j = \max(i, \Omega_\pi(v, w))$, where $\Omega_\pi(v, w)$ denotes priority associated to seeing the set $\gamma_C(v, w)$ of colors on memory π (cf. Definition 11).

- *Remove unsafe vertices.* A vertex (u, i) such that $(u, \pi@i)$ is contained neither in V_{2i} nor in V_{2i+1} is *unsafe.* Remove from M all vertices (v, i) such that $v \in V_\forall$ and there is $w \in E(v)$ such that (w, i) is unsafe; these are vertices from where player $\forall$ can access an unsafe exit point of the DAG.

- *Find safe and valid certificate loops:* Define, for all $i \in [2|C| + 1]$, a nondeterministic Emerson-Lei automaton $A_i = (Q_i, \delta, \alpha_S \wedge \alpha_W)$ (with singleton alphabet $\{*\}$) by putting $Q_i = \{(v, i) \mid (v, i) \text{ still exists in } M\}$ and $\delta((v, i), *) = \{(w, i) \mid w \in E(v) \text{ and } \Omega_\pi(v, w) \leq i\}$ for $(v, i) \in Q_i$. Compute the non-emptiness region of A_i and call it N_i.

- *Find safe stems:* Remove from M all vertices $(v, 0)$ for which there is no j such that some vertex from N_j is reachable (in M) from $(v, 0)$.

For all vertices $(v, 0)$ that are contained in M after this procedure terminates, there is a safe stem w leading (with maximal priority j) to some safe and valid loop u over N_j; (w, u) is a valid certificate for v. We state the correctness of the described procedure as follows.

▶ **Lemma 18.** *Given subsets $\overline{V} = V_0, \ldots V_{2k}$ of the real nodes in $P(C(G))$ and a real node (v, π) in $P(C(G))$, we have that player $\exists$ can attract to $\overline{V}$ from (v, π) within $\mathsf{Cert} \times \Pi(S)$ if and only if the set M contains the pair $(v, 0)$ after execution the above procedure (for parameters $\overline{V}$ and (v, π)).*

▶ **Corollary 19.** *DAG attractors in $P(C(G))$ can be computed in time $\mathcal{O}(d!dt)$ where t denotes the time it takes to check $\alpha_S \wedge \alpha_W$ automata of size n for non-emptiness.*

Proof. In order to compute a DAG attractor in $P(C(G))$, it suffices to execute the above procedure once for each $\pi \in \Pi(S)$, that is, $d!$ many times; a single execution of the procedure can be implemented in time $\mathcal{O}(dt)$. ◀

5.4 Faster solution of obliging games

We are now ready to state the main result of this section.

▶ **Theorem 20.** *Certificate games for objectives α_S and α_W with n nodes and $d := |S|$ colors for the strong objective can be solved in time $\mathcal{O}((d!n)^5 d!dt)$, where t denotes the time it takes to check Emerson-Lei automata of size n and with acceptance condition $\alpha_S \wedge \alpha_W$ for non-emptiness. If α_S is a parity objective, then the runtime bound is $\mathcal{O}(n^{\log(2d+1)}dt)$.*

Proof. By Lemma 12, it suffices to solve the paritized version $P(C(G))$ of $C(G)$. By Lemma 18, computing DAG attractors in $P(C(G))$ can be done in time $\mathcal{O}(d!dt)$. As we have $d < \log(d! \cdot n)$, $P(C(G))$ can be solved with $(d!n)^5$ computations of a DAG attractor by Lemma 17. If α_S is a parity objective, then the LAR construction is not necessary as $C(G)$ already is a parity game; DAG attractors then can be computed in time $\mathcal{O}(dt)$ and $C(G)$ can be solved with $\mathcal{O}(n^{\log(2d+1)})$ computations of a DAG attractor.   ◀

We collect results on the complexity of non-emptiness checking of Emerson-Lei automata with acceptance condition $\alpha_S \wedge \alpha_W$ for specific α_S and α_W. It is known (cf. Table 2. in [1]) that while the problem is in P for most frequently used objectives (subsuming automata with generalized Büchi, Rabin or Streett conditions, with linear or quadratic dependence on the number of colors), it is NP-complete for Emerson-Lei conditions. For combinations of such objectives, the problem remains in P unless one of the objectives is of type Emerson-Lei:

▶ **Lemma 21.** *The Rabin(d)∧Streett(k) non-emptiness problem is in P (more precisely: in $\mathcal{O}(mdk^2)$ for automata with m edges).*

Proof. Let A be an Emerson-Lei automaton with n nodes, m edges and acceptance condition $\text{Streett}(d) \wedge \text{Rabin}(k)$. We check A for non-emptiness as follows. Let the $\text{Rabin}(k)$ condition be encoded by k Rabin pairs (E_i, F_i). For each $1 \leq i \leq k$, check the same automaton but with acceptance condition $\text{Streett}(d) \wedge \text{Inf}(F_i) \wedge \text{Fin}(E_i)$ for emptiness; call this automaton A_i. The acceptance condition of A_i can be treated as a $\text{Streett}(d+2)$ condition where the two additional Streett pairs are $(\top, F_i)$ and $(E_i, \bot)$. As a state in A is non-empty if and only if there is some i such the state is non-empty in A_i, it suffices to check the k many $\text{Streett}(d+2)$ automata for emptiness. The claim follows from the bound on emptiness checking for Streett automata given in [1].   ◀

Using the equivalence of certificate games and obliging games (Theorem 8), Theorem 20 together with the described complexities of emptiness checking yields improved upper bounds on the runtime complexity of solving obliging games, shown in the table below; we let n denote the number of nodes; m the number of edges; b the size of a nondeterministic Büchi automaton accepting α_W; and t the time required for emptiness checking an Emerson-Lei automaton of size n with acceptance condition $\alpha_S \wedge \alpha_W$; finally we let o abbreviate $\max(|W|, |S|)$. Recall that the approach from [6] reduces an obliging game to a standard game in which the objective α' is the conjunction of α_S with a Büchi objective, incurring blowup b on the arena size. This game then can be transformed to a parity game $\mathcal{G}$ (with parameters v, e and r) incurring additional blowup for the LAR transformation of α' to a parity objective. Notice the definition of e (an underapproximation of the number of edges in $\mathcal{G}$) in column 4 and its usage in column 5.

For instance for an obliging game with objectives $\alpha_S = \text{Streett}(d)$ and $\alpha_W =$ generalized Büchi(d) consisting of the Streett requests, the approach from [6] reduces the game to a parity game $\mathcal{G}$ with $d!nd$ nodes, at least $d!md$ edges, and $2d$ priorities; such a game can be solved in time $\mathcal{O}(d!md^6(d!n)^5)$. In contrast, emptiness checking for $\alpha_S \wedge \alpha_W$-automata is just emptiness checking for generalized Büchi automata so that our new algorithm solves such games in time $\mathcal{O}(d!md^2(d!n)^5)$. For objectives $\alpha_S = \text{Rabin}(d)$ and $\alpha_W = \text{Streett}(k)$, the approach from [6] has time complexity $\mathcal{O}(m(d!k2^k)^6 n^5)$ while our algorithm has time

Table 2 Comparison of runtime complexities for solving obliging games of various types.

type of α_S	type of α_W	b	$|\mathcal{G}|(v,e,r)$ [6]	time [6]	t [1]	time here
parity(d)	Rabin(k)	$k+1$	dnb, dmb, d	$\mathcal{O}(e(dnb)^{\log d})$	$\mathcal{O}(mo^2)$	$\mathcal{O}(n^{\log d+1}dt)$
	Streett(k)	$2^k k$			$\mathcal{O}(mo^2)$	
	EL(k)	$2^k k$			$\mathcal{O}(mo^2 2^o)$	
Rabin(d) or Streett(d)	Rabin(k)	$k+1$	$d!nb, d!mb, 2d$	$\mathcal{O}(e(d!nb)^5)$	$\mathcal{O}(mo^3)$	$\mathcal{O}((d!n)^5 d!dt)$
	Streett(k)	$2^k k$			$\mathcal{O}(mo^3)$	
	EL(k)	$2^k k$			$\mathcal{O}(mo^2 2^o)$	
EL(d)	EL(k)	$2^k k$	$d!nb, d!mb, 2d$	$\mathcal{O}(e(d!nb)^5)$	$\mathcal{O}(mo^2 2^o)$	$\mathcal{O}((d!n)^5 d!dt)$
Streett(d)	g.Büchi(d)	d	$d!nb, d!mb, 2d$	$\mathcal{O}(e(d!nb)^5)$	$\mathcal{O}(md)$	$\mathcal{O}((d!n)^5 d!dt)$
GR[1](d,k)	g.Büchi(d)	d	$dknb, dkmb, 3$	$\mathcal{O}(e(dknb)^2)$	$\mathcal{O}(md)$	$\mathcal{O}((dk)^3 n^2 dt)$

complexity just $\mathcal{O}(m(d!)^6 n^5 do^3)$; this is due to the fact that Büchi automata that recognize Streett objectives are of exponential size (as they have to guess a set of Streett pairs and then verify their satisfaction), while emptiness checking for Streett(d)$\wedge$ Rabin(k)-automata can be done in time cubic in o.

6 Conclusion

We propose a new angle of looking at the solution of obliging games. In contrast to previous approaches that have been based on single-step game reasoning, our method requires players to make promises about their long-term future behavior, which we formalize using the concept of certificates (or, more generally, witnesses). This new approach to obliging games enables us to not only show their determinacy (with strategies that contain additional information), but to also significantly improve previously existing upper bounds both on the size of graciously winning strategies, and on the worst-case runtime complexity of the solution of such games.

Technically, we use our new approach to show that the strategy sizes for Emerson-Lei obliging games with strong objective α_S and weak objective α_W are linear in the number $|W|$ of colors used in the weak objective; we obtain a similar polynomial dependence on $|W|$ for the runtime of solving obliging games, however with the important exception of the case where α_W is a full Emerson-Lei objective that cannot be expressed by a simpler (e.g. Rabin or Streett) objective. In previous approaches, those dependencies on $|W|$ have been, in general, exponential. Thereby we show that the strategy complexity of α_S / α_W obliging games is not significantly higher than that of standard games with objective just α_S, and that in many cases, this holds for the runtime complexity as well.

We leave the existence of canonical certificates (cf. Remark 10) as an open question for future work; such canonical certificates would allow for the extraction of yet smaller graciously winning strategies for obliging games.

We solve certificate games by using the LAR reduction to obtain equivalent parity games and then solving these parity games by fixpoint acceleration, computing nested fixpoints of a function that checks for certificate existence. We conjecture that it is possible to directly compute the more involved nested fixpoint associated to Emerson-Lei objectives (as given in [15]) over the original game arena; this would avoid the LAR reduction step in the solution process.

References

1 Christel Baier, Frantisek Blahoudek, Alexandre Duret-Lutz, Joachim Klein, David Müller, and Jan Strejcek. Generic emptiness check for fun and profit. In *Automated Technology for Verification and Analysis, ATVA 2019*, volume 11781 of *LNCS*, pages 445–461. Springer, 2019. `doi:10.1007/978-3-030-31784-3_26`.

2 Roderick Bloem, Rüdiger Ehlers, Swen Jacobs, and Robert Könighofer. How to handle assumptions in synthesis. In *Workshop on Synthesis, SYNT 2014*, volume 157 of *EPTCS*, pages 34–50, 2014. `doi:10.4204/EPTCS.157.7`.

3 Roderick Bloem, Rüdiger Ehlers, and Robert Könighofer. Cooperative reactive synthesis. In *Automated Technology for Verification and Analysis, ATVA 2015*, volume 9364 of *Lecture Notes in Computer Science*, pages 394–410. Springer, 2015. `doi:10.1007/978-3-319-24953-7_29`.

4 Roderick Bloem, Barbara Jobstmann, Nir Piterman, Amir Pnueli, and Yaniv Sa'ar. Synthesis of reactive(1) designs. *J. Comput. Syst. Sci.*, 78(3):911–938, 2012. `doi:10.1016/J.JCSS.2011.08.007`.

5 Julian C. Bradfield and Igor Walukiewicz. The μ-calculus and model checking. In Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem, editors, *Handbook of Model Checking*, pages 871–919. Springer, 2018. `doi:10.1007/978-3-319-10575-8_26`.

6 Krishnendu Chatterjee, Florian Horn, and Christof Löding. Obliging games. In *Concurrency Theory, 21st International Conference, CONCUR 2010*, volume 6269 of *LNCS*, pages 284–296. Springer, 2010. `doi:10.1007/978-3-642-15375-4_20`.

7 Stefan Dziembowski, Marcin Jurdzinski, and Igor Walukiewicz. How much memory is needed to win infinite games? In *Logic in Computer Science, LICS 1997*, pages 99–110. IEEE Computer Society, 1997. `doi:10.1109/LICS.1997.614939`.

8 Javier Esparza, Jan Kretínský, Jean-François Raskin, and Salomon Sickert. From linear temporal logic and limit-deterministic Büchi automata to deterministic parity automata. *Int. J. Softw. Tools Technol. Transf.*, 24(4):635–659, 2022. `doi:10.1007/S10009-022-00663-1`.

9 Oliver Friedmann and Martin Lange. Deciding the unguarded modal μ-calculus. *J. Appl. Non Class. Logics*, 23(4):353–371, 2013. `doi:10.1080/11663081.2013.861181`.

10 Oliver Friedmann, Markus Latte, and Martin Lange. Satisfiability games for branching-time logics. *Log. Methods Comput. Sci.*, 9(4), 2013. `doi:10.2168/LMCS-9(4:5)2013`.

11 Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002. `doi:10.1007/3-540-36387-4`.

12 Yuri Gurevich and Leo Harrington. Trees, automata, and games. In *Symposium on Theory of Computing, STOC 1982*, pages 60–65. ACM, 1982. `doi:10.1145/800070.802177`.

13 Daniel Hausmann. Faster game solving by fixpoint acceleration. *CoRR*, abs/2404.13687, 2024. `arXiv:2404.13687`.

14 Daniel Hausmann. Faster game solving by fixpoint acceleration. In *Fixed Points in Computer Science, FICS 2024*, EPTCS, 2024, to appear.

15 Daniel Hausmann, Mathieu Lehaut, and Nir Piterman. Symbolic solution of Emerson-Lei games for reactive synthesis. In *Foundations of Software Science and Computation Structures, FoSSaCS 2024*, volume 14574 of *LNCS*, pages 55–78. Springer, 2024. `doi:10.1007/978-3-031-57228-9_4`.

16 Daniel Hausmann and Lutz Schröder. Game-based local model checking for the coalgebraic μ-calculus. In *Concurrency Theory, CONCUR 2019*, volume 140 of *LIPIcs*, pages 35:1–35:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. `doi:10.4230/LIPICS.CONCUR.2019.35`.

17 Paul Hunter and Anuj Dawar. Complexity bounds for regular games. In *Mathematical Foundations of Computer Science, MFCS 2005*, volume 3618 of *Lecture Notes in Computer Science*, pages 495–506. Springer, 2005. `doi:10.1007/11549345_43`.

18   Michael Luttenberger, Philipp J. Meyer, and Salomon Sickert. Practical synthesis of reactive
         systems from LTL specifications via parity games. *Acta Informatica*, 57(1-2):3–36, 2020.
         `doi:10.1007/s00236-019-00349-3`.

19   Rupak Majumdar, Nir Piterman, and Anne-Kathrin Schmuck. Environmentally-friendly GR(1)
         synthesis. In *Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2019*,
         volume 11428 of *LNCS*, pages 229–246. Springer, 2019. `doi:10.1007/978-3-030-17465-1_13`.

20   Rupak Majumdar and Anne-Kathrin Schmuck. Supervisory controller synthesis for nonter-
         minating processes is an obliging game. *IEEE Trans. Autom. Control.*, 68(1):385–392, 2023.
         `doi:10.1109/TAC.2022.3143108`.

21   Donald A. Martin. Borel determinacy. *Annals of Mathematics*, 102(2):363–371, 1975. `doi:`
         `10.2307/1971035`.

22   Amir Pnueli and Roni Rosner. On the synthesis of a reactive module. In *Principles of
         Programming Languages, POPL 1989*, pages 179–190. ACM Press, 1989. `doi:10.1145/75277.`
         `75293`.

23   Tom van Dijk, Feije van Abbema, and Naum Tomov. Knor: reactive synthesis using oink. In
         Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2024, volume
         14570 of *LNCS*, pages 103–122. Springer, 2024. `doi:10.1007/978-3-031-57246-3_7`.