# Symbolic Solution of Emerson-Lei Games for Reactive Synthesis[⋆]

Daniel Hausmann, Mathieu Lehaut and Nir Piterman

University of Gothenburg, Gothenburg, Sweden

**Abstract.** Emerson-Lei conditions have recently attracted attention due to both their succinctness and their favorable closure properties. In the current work, we show how infinite-duration games with Emerson-Lei objectives can be analyzed in two different ways. First, we show that the Zielonka tree of the Emerson-Lei condition naturally gives rise to a new reduction to parity games. This reduction, however, does not result in optimal analysis. Second, we show based on the first reduction (and the Zielonka tree) how to provide a direct fixpoint-based characterization of the winning region. The fixpoint-based characterization allows for symbolic analysis. It generalizes the solutions of games with known winning conditions such as Büchi, GR[1], parity, Streett, Rabin and Muller objectives, and in the case of these conditions reproduces previously known symbolic algorithms and complexity results.
We also show how the capabilities of the proposed algorithm can be exploited in reactive synthesis, suggesting a new expressive fragment of LTL that can be handled symbolically. Our fragment combines a safety specification and a liveness part. The safety part is unrestricted and the liveness part allows to define Emerson-Lei conditions on occurrences of letters. The symbolic treatment is enabled due to the simplicity of determinization in the case of safety languages and by using our new algorithm for game solving. This approach maximizes the number of steps solved symbolically in order to maximize the potential for efficient symbolic implementations.

## 1 Introduction

Infinite-duration two-player games are a strong tool that has been used, notably, for reactive synthesis from temporal specifications [38]. Many different winning conditions are considered in the literature.

Emerson-Lei (EL) conditions [21] were initially suggested in the context of automata but are among the most general (regular) winning conditions considered for such games. They succinctly express general liveness properties by encoding Boolean combinations of events that should occur infinitely or finitely often. Automata and games in which acceptance or winning is defined by Emerson-Lei conditions have garnered attention in recent years [35,40,27,25], in particular

⋆ This work is supported by the ERC Consolidator grant D-SynMA (No. 772459).

because of their succinctness and good compositionality properties (Emerson-Lei objectives are closed under conjunction, disjunction, and negation). In this work, we show how infinite-duration two-player games with Emerson-Lei winning conditions can be solved symbolically.

It has been established that solving Emerson-Lei games is PSpace-complete and that an exponential amount of memory may be required by winning strategies [25]. Zielonka trees are succinct tree-representations of Muller objectives [47]. They have been used to obtain tight bounds on the amount of memory needed for winning in Muller games [18], and can also be applied to analyze Emerson-Lei objectives and games. One indirect way to solve Emerson-Lei games is by transformation to equivalent parity games using later-appearance-records [25], and then solving the resulting parity games. Another, more recent, indirect approach goes through Rabin games by first extracting history-deterministic Rabin automata from Zielonka trees and then solving the resulting Rabin games [12]. Both these indirect solution methods are enumerative by nature. Here, we give a direct symbolic algorithmic solution for Emerson-Lei games. We show how the Zielonka tree allows to directly encode the game as a parity game. Furthermore, building on this reduction, we show how to construct a fixpoint equation system that captures winning in the game. As usual, fixpoint equation systems are recipes for game solving algorithms that manipulate sets of states symbolically. To the best of our knowledge, we thereby give the first description of a fully symbolic algorithm for the solution of Emerson-Lei games.

The algorithm that we obtain in this way is adaptive in the sense that the nesting structure of recursive calls is obtained directly from the Zielonka tree of the given winning objective. As the Zielonka tree is specific to the objective, this means that the algorithm performs just the fixpoint computations that are required for that specific objective. In particular, our algorithm instantiates to previously known fixpoint iteration algorithms in the case that the objective is a (generalized) Büchi, GR[1], parity, Streett, Rabin or Muller condition, reproducing previously known algorithms and complexity results. As we use fixpoint iteration, the instantiation of our algorithm to parity game solving is not directly a quasipolynomial algorithm. In the general setting, the algorithm solves unrestricted Emerson-Lei games with $k$ colors, $m$ edges and $n$ nodes in time $\mathcal{O}(k! \cdot m \cdot n^k)$ and yields winning strategies with memory $\mathcal{O}(k!)$.

We apply our symbolic solution of Emerson-Lei games to the automated construction of safe systems. The ideas of synthesis of reactive systems from temporal specifications go back to the early days of computer science [14]. These concepts were modernized and connected to linear temporal logic (LTL) and finite-state automata by Pnueli and Rosner [38]. In recent years, practical applications in robotics are using this form of synthesis as part of a framework producing correct-by-design controllers [28,6,44,32,34].

A prominent way to extend the capacity of reasoning about state spaces is by reasoning *symbolically* about sets of states/paths. In order to apply this approach to reactive synthesis, different fragments of LTL that allow symbolic game analysis have been considered. Notably, the GR[1] fragment has been widely used for the applications in robotics mentioned above [37,7]. But also larger fragments are being considered and experimented with [20,19,41]. Recently, De Giacomo and Vardi suggested that similar advantages can be had by changing the usual semantics of LTL from considering infinite models to finite models ($\mathrm{LTL}_f$) [22]. The complexity of the problem remains doubly-exponential, however, symbolic techniques can be applied. As models are finite, it is possible to use the classical subset construction (in contrast to Büchi determinization), which can be reasoned about symbolically. Furthermore, the resulting games have simple reachability objectives. This approach with finite models is used for applications in planning [11,10] and robotics [6].

Here, we harness our symbolic solution to Emerson-Lei games to suggest a large fragment of LTL that can be reasoned about symbolically. We introduce the *Safety and Emerson-Lei* fragment whose formulas are conjunctions $\varphi_{\text{safety}} \wedge \varphi_{\text{EL}}$ between an (unrestricted) safety condition and an (unrestricted) Emerson-Lei condition defined in terms of game states. This fragment generalizes GR[1] and the previously mentioned works in [20,19,41]. We approach safety and Emerson-Lei LTL synthesis in two steps: first, consider only the safety part and convert it to a symbolic safety automaton; second, reason symbolically on this automaton by solving Emerson-Lei games using our novel symbolic algorithm.

$$\begin{tikzcd}
\varphi_{\text{safety}} \wedge \varphi_{\text{EL}} \arrow[r] \arrow[dr, bend right] & \underset{\text{(Symbolic Safety)}}{\mathcal{D}_{\varphi_{\text{safety}}}} \arrow[r] & \text{synthesis game } G_{\varphi_{\text{safety}} \wedge \varphi_{\text{EL}}} \\
& \underset{\text{(Emerson-Lei objective)}}{\varphi_{\text{EL}}} \arrow[ur] &
\end{tikzcd}$$

We show that realizability of a safety and Emerson-Lei formula $\varphi_{\text{safety}} \wedge \varphi_{\text{EL}}$ can be checked in time $2^{\mathcal{O}(m \cdot \log m \cdot 2^n)}$, where $n = |\varphi_{\text{safety}}|$ and $m = |\varphi_{\text{EL}}|$. The overall procedure therefore is doubly-exponential in the size of the safety part but only single-exponential in the size of the liveness part; notably, both the automaton determinization and game solving parts can be implemented symbolically.

We begin by recalling Emerson-Lei games and Zielonka trees in Section 2, and also prove an upper bound on the size of Zielonka trees. Next we show how to solve Emerson-Lei games by fixpoint computation in Section 3. In Section 4 we formally introduce the safety and Emerson-Lei fragment of LTL and show how to construct symbolic games with Emerson-Lei objectives that characterize realizability and that can be solved using the algorithm proposed in Section 3. Omitted proofs and further details can be found in the full version of this paper [23].

## 2 Emerson-Lei Games and Zielonka Trees

We recall the basics of Emerson-Lei games [25] and Zielonka trees [47], and also show an apparently novel bound on the size of Zielonka trees; previously, the main interest was on the size of winning strategies induced by Zielonka trees, which is smaller [18].

*Emerson-Lei games.* We consider two-player games played between the *existential player* $\exists$ and its opponent, the *universal player* $\forall$. A *game arena* $A = (V, V_\exists, V_\forall, E)$ consists of a set $V = V_\exists \uplus V_\forall$ of nodes, partitioned into sets of *existential nodes* $V_\exists$ and *universal nodes* $V_\forall$, and a set $E \subseteq V \times V$ of *moves*; we put $E(v) = \{v' \in V \mid (v, v') \in E\}$ for $v \in V$. A *play* $\pi = v_0 v_1 \dots$ then is a sequence of nodes such that for all $i \geq 0$, $(v_i, v_{i+1}) \in E$; we denote the set of plays in $A$ by $\mathsf{plays}(A)$. A *game* $G = (A, \alpha)$ consists of a game arena $A$ together with an objective $\alpha \subseteq \mathsf{plays}(A)$.

A *strategy* for the existential player is a function $\sigma : V^* \cdot V_\exists \to V$ such that for all $\pi \in V^*$ and $v \in V_\exists$ we have $(v, \sigma(\pi v)) \in E$. A play $v_0 v_1 \dots$ is said to be *compliant* with strategy $f$ if for all $i \geq 0$ such that $v_i \in V_\exists$ we have $v_{i+1} = \sigma(v_0 \dots v_i)$. Strategy $\sigma$ is *winning* for the existential player from node $v \in V$ if all plays starting in $v$ that are compliant with $\sigma$ are contained in $\alpha$; then we say that the existential player *wins* $v$. We denote by $W_\exists$ the *winning region* for the existential player (that is, the set of nodes that the existential player wins).

In *Emerson-Lei games*, each node is colored by a set of colors, and the objective $\alpha$ is induced by a formula that specifies combinations of colors that have to be visited infinitely often, or are allowed to be visited only finitely often. Formally, we fix a set $C$ of colors and use *Emerson-Lei formulas*, that is, finite positive Boolean formulas $\varphi \in \mathbb{B}_+(\{\mathsf{Inf}\, c, \mathsf{Fin}\, c\}_{c \in C})$ over atoms of the shape $\mathsf{Inf}\, c$ or $\mathsf{Fin}\, c$, to define sets of plays. The satisfaction relation $\models$ for a set $D \subseteq C$ of colors and an Emerson-Lei formula $\varphi$ (written $D \models \varphi$) is defined in the usual inductive way; $D$ will represent the set of colors that are visited infinitely often by plays. E.g. the clauses for atoms $\mathsf{Inf}\, c$ and $\mathsf{Fin}\, c$ are

$$D \models \mathsf{Inf}\, c \Leftrightarrow c \in D \qquad\qquad D \models \mathsf{Fin}\, c \Leftrightarrow c \notin D$$

Consider a game arena $A = (V, V_\exists, V_\forall, E)$. An *Emerson-Lei condition* is given by an Emerson-Lei formula $\varphi$ together with a coloring function $\gamma : V \to 2^C$ that assigns a (possibly empty) set $\gamma(v)$ of colors to each node $v \in V$. The formula $\varphi$ and the coloring function $\gamma$ together specify the objective

$$\alpha_{\gamma,\varphi} = \Big\{ v_0 v_1 \dots \in \mathsf{plays}(A) \Big| \{c \in C \mid \forall i.\ \exists j \geq i.\ c \in \gamma(v_j)\} \models \varphi \Big\}$$

Thus a play $\pi = v_0 v_1 \dots$ is winning for the existential player (formally: $\pi \in \alpha_{\gamma,\varphi}$) if and only if the set of colors that are visited infinitely often by $\pi$ satisfies $\varphi$. Below, we will also make use of *parity games*, denoted by $(V, V_\exists, V_\forall, E, \Omega)$ where $\Omega : V \to \{1, \dots, 2k\}$ (for $k \in \mathbb{N}$) is a priority function, assigning priorities to game nodes. The objective of the existential player then is that the maximal priority that is visited infinitely often is an even number. Parity games are an instance of Emerson-Lei games, obtained with set $C = \{p_1, \dots, p_{2k}\}$ of colors, a coloring function that assigns exactly one color to each node and with objective

$$\mathsf{Parity}(p_1, \dots, p_{2k}) = \bigvee\nolimits_{i \text{ even}} \Big( \mathsf{Inf}\, p_i \wedge \bigwedge\nolimits_{i < j \leq 2k} \mathsf{Fin}\, p_j \Big).$$

Similarly, Emerson-Lei objectives directly encode (combinations of) other standard objectives, such as Büchi, Rabin, Streett or Muller conditions:

| | |
|---|---|
| — $\mathsf{Inf}\, f$ | $\mathsf{Büchi}(f)$ |
| — $\bigvee_{1\leq i\leq k}(\mathsf{Inf}\, e_i \wedge \mathsf{Fin}\, f_i)$ | $\mathsf{Rabin}(e_1, f_1, \ldots, e_k, f_k)$ |
| — $\bigwedge_{1\leq i\leq k}(\mathsf{Fin}\, r_i \vee \mathsf{Inf}\, g_i)$ | $\mathsf{Streett}(r_1, g_1, \ldots, r_k, g_k)$ |
| — $\bigvee_{D\in\mathcal{U}}(\bigwedge_{c\in D} \mathsf{Inf}\, c \wedge \bigwedge_{d\in C\setminus D} \mathsf{Fin}\, d)$ | $\mathsf{Muller}(\mathcal{U} \subseteq 2^C)$ |

*Zielonka Trees.* We introduce a succinct encoding of the algorithmic essence of Emerson-Lei objectives in the form of so-called Zielonka trees [47,18].

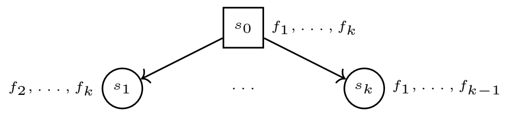
**Definition 1.** *The* Zielonka tree *for an Emerson-Lei formula $\varphi$ over set $C$ of colors is a tuple $\mathcal{Z}_\varphi = (T, R, l)$ where $(T, R \subseteq T \times T)$ is a tree and $l : T \to 2^C$ is a labeling function that assigns sets $l(t)$ of colors to vertices $t \in T$. We denote the root of $(T, R)$ by $r$. Then $\mathcal{Z}_\varphi$ is defined to be the unique tree (up to reordering of child vertices) that satisfies the following constraints.*

- *The root vertex is labeled with $C$, that is, $l(r) = C$.*
- *Each vertex $t$ has exactly one child vertex $t_D$ (labeled with $l(t_D) = D$) for each set $D$ of colors that is maximal in $\{D' \subsetneq l(t) \mid D' \models \varphi \Leftrightarrow l(t) \not\models \varphi\}$.*

*For $s, t \in T$ such that $s$ is an ancestor of $t$, we write $s \leq t$. Given a vertex $s \in T$, we denote its set of direct successors by $R(s) = \{t \in T \mid (s, t) \in R\}$ and the set of leafs below it by $L(s) = \{t \in T \mid s \leq t$ and $R(t) = \emptyset\}$; we write $L$ for the set of all leafs. We assume some fixed total order $\preceq$ on $T$ that respects $\leq$; this order induces a numbering of $T$. A vertex $t$ in the Zielonka tree is said to be* winning *if $l(t) \models \varphi$, and* losing *otherwise. We let $T_\Box$ and $T_\bigcirc$ denote the sets of winning and losing vertices in $\mathcal{Z}_\varphi$, respectively. Finally, we assign a* level $\mathsf{lev}(t)$ *to each vertex $t \in T$ so that $\mathsf{lev}(r) = |C|$, and $\mathsf{lev}(s') = \mathsf{lev}(s) - 1$ for all $(s, s') \in R$.*

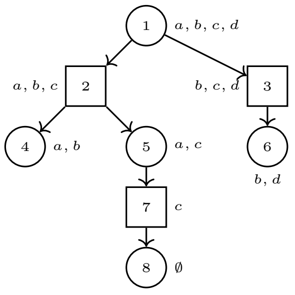
*Example 2.* As mentioned above, Emerson-Lei games and Zielonka trees instantiate naturally to games with, e.g., Büchi, generalized Büchi, GR[1], parity, Rabin, Streett and Muller objectives; for brevity, we illustrate this for selected examples here (more instances can be found in [23]).

1. *Generalized Büchi condition*: Given $k$ colors $f_1, \ldots, f_k$, the winning objective $\varphi = \bigwedge_{1\leq i\leq k} \mathsf{Inf}\ f_i$ expresses that all colors are visited infinitely often (not necessarily simultaneously); the induced Zielonka tree is depicted below with boxes and circles representing winning and losing vertices, respectively.



2. *Streett condition*: The vertices in the Zielonka tree for Streett condition given by $\varphi = \bigwedge_{1\leq i\leq k} (\mathsf{Fin}\ r_i \vee \mathsf{Inf}\ g_i)$ are identified by duplicate-free lists $\mathsf{L}$ of colors (each entry being $r_i$ or $g_i$ for some $1 \leq i \leq k$) that encode the vertex position in the tree. Vertex $\mathsf{L}$ has label $l(\mathsf{L}) = C \setminus \mathsf{L}$ and is winning if and only if $|\mathsf{L}|$ is even. Winning vertices $\mathsf{L}$ have one child vertex $\mathsf{L} : g_j$ for each $g_j \in C \setminus \mathsf{L}$ resulting in $|C \setminus \mathsf{L}|/2$ many child vertices. Losing vertices $\mathsf{L}$ have the single child vertex $\mathsf{L} : r_j$ where the last entry $\mathsf{last}(\mathsf{L})$ in $\mathsf{L}$ is $g_j$. All leafs are winning and are labeled with $\emptyset$. The tree has height $2k$ and $2(k!)$ vertices.

3. To obtain a Zielonka tree that has branching at both winning and losing vertices, we consider the objective $\varphi_{EL} = (\mathsf{Fin}\ a \vee \mathsf{Inf}\ b) \wedge ((\mathsf{Fin}\ a \vee \mathsf{Fin}\ d) \wedge \mathsf{Inf}\ c)$. This property can be seen as the conjunction of a Streett pair $(a, b)$ with two disjunctive Rabin pairs $(c, a)$ and $(c, d)$, altogether stating that $c$ occurs infinitely often and $a$ occurs finitely often or $b$ occurs infinitely often and $d$ occurs finitely often. Below we depict the induced Zielonka tree.



**Lemma 3.** *The height and the branching width of $\mathcal{Z}_\varphi$ are bounded by $|C|$ and $2^{|C|}$ respectively; the number of vertices in $\mathcal{Z}_\varphi$ is bounded by $e|C|!$ (where $e$ is Euler's number).*

## 3 Solving Emerson-Lei Games

We now show how to extract from the Zielonka tree of an Emerson-Lei objective a fixpoint characterization of the winning regions of an Emerson-Lei game. Solving the game then reduces to computing the fixpoint, yielding a game solving algorithm that works by fixpoint iteration and hence is directly open to symbolic implementation. The algorithm is adaptive in the sense that the structure of its recursive calls is extracted from the Zielonka tree and hence tailored to the objective. As a stepping stone towards obtaining our fixpoint characterization, we first show how Zielonka trees can be used to reduce Emerson-Lei games to parity games that are structured into tree-like subgames.

Recall that $G = (V, V_\exists, V_\forall, E, \alpha_{\gamma,\varphi})$ is an Emerson-Lei game and that the associated Zielonka tree is $\mathcal{Z}_\varphi = (T, R, l)$ with set $L$ of leaves, sets $T_\bigcirc$ and $T_\Box$ of winning and losing vertices, respectively, and with root $r$. Following [18], we define the *anchor vertex* of $v \in V$ and $t \in T$ by

$$\mathsf{anchor}(v, t) = \max_{\leq}\{s \in T \mid s \leq t \wedge \gamma(v) \subseteq l(s)\};$$

it is the lower-most ancestor of $t$ that contains $\gamma(v)$ in its label.

*A novel reduction to parity games.* Intuitively, our reduction annotates nodes in $G$ with leaves of $\mathcal{Z}_\varphi$ that act as a memory, holding information about the order in which colors have been visited. In the reduced game, the memory value $t \in L$ is updated according to a move from $v$ to $w$ in $G$ by playing a subgame along the Zielonka tree. This subgame starts at the anchor vertex of $v$ and $t$ and the players in turn pick child vertices, with the existential player choosing the branch that is taken at vertices from $T_{\bigcirc}$ and the universal player choosing at vertices from $T_{\Box}$.[1] Once this subgame reaches a leaf $t' \in L$, the memory value is updated to $t'$ and another step of $G$ is played. Due to the tree structure of $\mathcal{Z}_\varphi$ every play in the reduced game (walking through the Zielonka tree in the described way, repeatedly jumping from a leaf to an anchor vertex and then descending to a leaf again) has a unique topmost vertex from $T$ that it visits infinitely often; by the definition of anchor vertices, the label of this vertex corresponds to the set of colors that is visited infinitely often by the according play of $G$. A parity condition can be used to decide whether this vertex is winning or losing.

Formally, we define the parity game $P_G = (V', V'_\exists, V'_\forall, E', \Omega)$, played over $V' = V \times T$, as follows. Nodes $(v,t) \in V'$ are owned by the existential player if either $t$ is not a leaf, and it is not a winning vertex ($t \notin L$ and $t \in T_{\bigcirc}$), or if $t$ is a leaf and, in $G$, $v$ is owned by the existential player ($t \in L$ and $v \in V_\exists$); all other nodes are owned by the universal player. Moves and priorities are defined by

$$E'(v,t) = \begin{cases} \{v\} \times R(t) & t \notin L \\ E(v) \times \{\mathsf{anchor}(v,t)\} & t \in L \end{cases} \qquad \Omega(v,t) = \begin{cases} 2 \cdot \mathsf{lev}(t) & t \in T_{\Box} \\ 2 \cdot \mathsf{lev}(t) + 1 & t \in T_{\bigcirc} \end{cases}$$

for $(v,t) \in V'$. Thus from $(v,t)$ such that $t$ is a leaf ($t \in L$), the owner of $v$ picks a move $(v,w) \in E$ and the game continues with $(w, \mathsf{anchor}(v,t))$. From $(v,t)$ such that $t$ is not a leaf ($t \notin L$), the owner of $t$ picks a child $t' \in R(t)$ of $t$ in the Zielonka tree and the game continues with $(v,t')$, leaving the game node component $v$ unchanged. Therefore, plays in $P_G$ correspond to plays from $G$ that are annotated with memory values $t \in T$ that are updated according to the colors that are visited (by moving to the anchor vertex); in addition to that, the owners of vertices in the Zielonka Tree are allowed to decide (by selecting one of the child vertices) with which colors they intend to satisfy the sub-objectives that are encoded by vertex labels. The priority function $\Omega$ then is used to identify the top-most anchor vertex $s$ that is visited infinitely often in a play of $P_G$, deciding a play to be winning if and only if $s$ is a winning vertex ($t \in T_{\Box}$). We note that $|V'| = |V| \cdot |T| \leq |V| \cdot e|C|!$ by Lemma 3.

**Theorem 4.** *For all $v \in V$, the existential player wins $v$ in the Emerson-Lei game $G$ if and only if the existential player wins $(v,r)$ in the parity game $P_G$.*

This reduction yields a novel indirect method to solve Emerson-Lei games with $n$ nodes and $k$ colors by solving parity games with $n \cdot ek!$ nodes and $2k$ priorities; by itself, this reduction does not improve upon using later appearance records [25]. However, the game $P_G$ consists of subgames of particular tree-like shapes. The remainder of this section is dedicated to showing how the special structure of $P_G$ allows for direct symbolic solution by solving equivalent systems of fixpoint equations over $V$ (rather than over the exponential-sized set $V'$).

[1] Players choose from vertices where they lose, which explains the notation $T_{\Box}$ and $T_{\bigcirc}$.

*Fixpoint equation systems.* Recall (from e.g. [4]) that a hierarchical system of fixpoint equations is given by equations

$$X_i =_{\eta_i} f_i(X_1, \dots, X_k)$$

for $1 \leq i \leq k$, where $\eta_i \in \{\mathsf{GFP}, \mathsf{LFP}\}$ and the $f_i : \mathcal{P}(V)^k \to \mathcal{P}(V)$ are *monotone* functions, that is, $f_i(A_1, \dots, A_k) \subseteq f_i(B_1, \dots, B_k)$ whenever $A_j \subseteq B_j$ for all $1 \leq j \leq k$. As we aim to use fixpoint equation systems to characterize winning regions of games, it is convenient to define the semantics of equation systems also in terms of games, as proposed in [4]. For a system $S$ of $k$ fixpoint equations, the *fixpoint game* $G_S = (V, V_\exists, V_\forall, E, \Omega)$ is a parity game with sets of nodes $V_\exists = V \times \{1, \dots, k\}$ and $V_\forall = \mathcal{P}(V)^k$. The set of edges $E$ and the priority function $\Omega : V \to \{0, \dots, 2k-1\}$ are defined, for $(v, i) \in V_\exists$ and $\bar{A} = (A_1, \dots, A_k) \in V_\forall$, by

$$E(v, i) = \{\bar{A} \in V_\forall \mid v \in f_i(\bar{A})\} \qquad E(\bar{A}) = \{(v, i) \in V_\exists \mid v \in A_i\}$$

and by $\Omega(v, i) = 2(k-i) + \iota_i$ and $\Omega(\bar{A}) = 0$, where $\iota_i = 1$ if $\eta_i = \mathsf{LFP}$ and $\iota_i = 0$ if $\eta_i = \mathsf{GFP}$. We say that $v$ is contained in the *solution* of variable $X_i$ (denoted by $v \in [\![X_i]\!]$) if and only if the existential player wins the node $(v, i)$ in $G_S$. In order to show containment of a node $v$ in the solution of $X_i$, the existential player thus has to provide a solution $(A_1, \dots, A_k) \in V_\forall$ for all variables such that $v \in f_i(A_1, \dots, A_k)$; the universal player in turn can challenge a claimed solution $(A_1, \dots, A_k)$ by picking some $1 \leq i \leq k$ and $v \in A_i$ and moving to $(v, i)$. The game objective checks whether the dominating equation in a play (that is, the equation with minimal index among the equations that are evaluated infinitely often in the play) is a least or a greatest fixpoint equation.

Baldan et al. have shown in [4] that this game characterization is equivalent to the more traditional Knaster-Tarski-style definition of the semantics of fixpoint equation systems in terms of nested fixpoints of the involved functions $f_i$.

To give a flavor of the close connection between fixpoint equation systems and winning regions in games, we recall that for a given set $V$ of nodes, the *controllable predecessor function* $\mathsf{CPre} : 2^V \to 2^V$ is defined, for $X \subseteq V$, by

$$\mathsf{CPre}(X) = \{v \in V_\exists \mid E(v) \cap X \neq \emptyset\} \cup \{v \in V_\forall \mid E(v) \subseteq X\}.$$

*Example 5.* Given a Büchi game $(V, V_\exists, V_\forall, E, \mathsf{Inf}\ f)$ with coloring function $\gamma : V \to 2^{\{f\}}$, the winning region of the existential player is the solution of the equation system

$$X_1 =_{\mathsf{GFP}} X_2 \qquad\qquad X_2 =_{\mathsf{LFP}} (f \cap \mathsf{CPre}(X_1)) \cup (\overline{f} \cap \mathsf{CPre}(X_2))$$

where $f = \{v \in V \mid \gamma(v) = \{f\}\}$ and $\overline{f} = V \setminus f$.

Our upcoming fixpoint characterization of winning regions in Emerson-Lei games uses the following notation that relates game nodes with anchor vertices in the Zielonka tree.

**Definition 6.** *For a set $D \subseteq C$ of colors, and $\bowtie \in \{\subseteq, \not\subseteq\}$ we put $\gamma^{-1}_{\bowtie D} = \{v \in V \mid \gamma(v) \bowtie D\}$. For $s, t \in T$ such that $s < t$ (that is, $s$ is an ancestor of $t$ in $\mathcal{Z}_\varphi$), we define*

$$\mathsf{anc}^s_t = \gamma^{-1}_{\subseteq l(s)} \cap \gamma^{-1}_{\not\subseteq l(s_t)}$$

*where $s_t$ is* the *child vertex of $s$ that leads to $t$; we also put $\mathsf{anc}^t_t = \gamma^{-1}_{\subseteq l(t)}$.*

Note that for fixed $t \in T$ and $v \in V$, there is a unique $s \in T$ such that $s \leq t$ and $v \in \mathsf{anc}^s_t$ (possibly, $s = t$); this $s$ is the anchor vertex of $t$ at $v$.

Next, we present our fixpoint characterization of winning in Emerson-Lei games, noting that it closely follows the definition of $P_G$.

**Definition 7 (Emerson-Lei equation system).** *We define the system $S_\varphi$ of fixpoint equations for the objective $\varphi$ by putting*

$$X_s =_{\eta_s} \begin{cases} \bigcup_{t \in R(s)} X_t & R(s) \neq \emptyset, s \in T_{\bigcirc} \\ \bigcap_{t \in R(s)} X_t & R(s) \neq \emptyset, s \in T_{\Box} \\ \bigcup_{s' \leq s} \left(\mathsf{anc}^{s'}_s \cap \mathsf{CPre}(X_{s'})\right) & R(s) = \emptyset \end{cases}$$

*for $s \in T$. For every $t \in T$, we use $X_t$ to refer to the variable $X_i$ where $i$ is the index of $t$ according to $\preceq$ and similarly for $\eta_t$. Furthermore, $\eta_s = \mathsf{GFP}$ if $s \in T_\Box$ and $\eta_s = \mathsf{LFP}$ if $s \in T_{\bigcirc}$.*

*Example 8.* Instantiating Definition 7 to the Büchi objective $\varphi = \mathsf{Inf}\ f$ yields exactly the equation system given in Example 5. Revisiting the objectives from Example 2, we obtain the following fixpoint characterizations (further examples can be found in [23]).

1. *Generalized Büchi condition:*

   $$X_{s_0} =_{\mathsf{GFP}} \bigcap\nolimits_{1 \leq i \leq k} X_{s_i} \quad X_{s_i} =_{\mathsf{LFP}} (\mathsf{anc}^{s_0}_{s_i} \cap \mathsf{CPre}(X_{s_0})) \cup (\mathsf{anc}^{s_i}_{s_i} \cap \mathsf{CPre}(X_{s_i}))$$

   where $\mathsf{anc}^{s_0}_{s_i} = \gamma^{-1}_{\subseteq C} \cap \gamma^{-1}_{\not\subseteq C \setminus \{f_i\}} = \{v \in V \mid f_i \in \gamma(v)\}$ and $\mathsf{anc}^{s_i}_{s_i} = \gamma^{-1}_{\subseteq C \setminus \{f_i\}}$.
2. *Streett condition*:

   $$X_{\mathsf{L}} =_{\eta_{\mathsf{L}}} \begin{cases} \bigcap_{g_j \notin \mathsf{L}} X_{\mathsf{L}:g_j} & |\mathsf{L}| \text{ even}, |\mathsf{L}| < 2k \\ X_{\mathsf{L}:r_j} & |\mathsf{L}| \text{ odd}, \mathsf{last}(\mathsf{L}) = g_j \\ (\mathsf{anc}^{[]}_{\mathsf{L}} \cap \mathsf{CPre}(X_{[]})) \cup \ldots \cup (\mathsf{anc}^{\mathsf{L}}_{\mathsf{L}} \cap \mathsf{CPre}(X_{\mathsf{L}})) & |\mathsf{L}| = 2k \end{cases}$$

   where $\eta_{\mathsf{L}} = \mathsf{GFP}$ if $|\mathsf{L}|$ is even and $\eta_{\mathsf{L}} = \mathsf{LFP}$ if $|\mathsf{L}|$ is odd. Here, $\mathsf{anc}^{\mathsf{K}}_{\mathsf{L}} = \gamma^{-1}_{\subseteq C \setminus \mathsf{K}} \cap \gamma^{-1}_{\not\subseteq C \setminus I}$ for $\mathsf{K} \neq \mathsf{L}$ and $I = \mathsf{K}_{\mathsf{L}}$, and $\mathsf{anc}^{\mathsf{L}}_{\mathsf{L}} = \gamma^{-1}_{\subseteq \emptyset}$, both for $\mathsf{L}$ such that $|\mathsf{L}| = 2k$.
3. The equation system associated to the Zielonka tree for the complex objective $\varphi_{EL}$ from Example 2.3 is as follows, where we use a formula over the colors to denote the set of vertices whose label satisfies the formula. For example,

$b \wedge \neg d$ corresponds to vertices whose set of colors contains $b$ but does not contain $d$.

$$X_1 =_{\mathsf{LFP}} X_2 \cup X_3 \qquad X_2 =_{\mathsf{GFP}} X_4 \cap X_5 \qquad X_3 =_{\mathsf{GFP}} X_6 \qquad X_5 =_{\mathsf{LFP}} X_7 \qquad X_7 =_{\mathsf{GFP}} X_8$$
$$X_4 =_{\mathsf{LFP}} (\neg c \wedge \neg d \cap \mathsf{Cpre}(X_4)) \cup (c \wedge \neg d \cap \mathsf{Cpre}(X_2)) \cup (d \cap \mathsf{Cpre}(X_1))$$
$$X_6 =_{\mathsf{LFP}} (\neg a \wedge \neg c \cap \mathsf{Cpre}(X_6)) \cup (\neg a \wedge c \cap \mathsf{Cpre}(X_3)) \cup (a \cap \mathsf{Cpre}(X_1))$$
$$X_8 =_{\mathsf{LFP}} (\neg a \wedge \neg b \wedge \neg c \wedge \neg d \cap \mathsf{Cpre}(X_8)) \cup (\neg a \wedge \neg b \wedge c \wedge \neg d \cap \mathsf{Cpre}(X_7)) \cup$$
$$(a \wedge \neg b \wedge \neg d \cap \mathsf{Cpre}(X_5)) \cup (b \wedge \neg d \cap \mathsf{Cpre}(X_2)) \cup (d \cap \mathsf{Cpre}(X_1)),$$

**Theorem 9.** *Referring to the equation system from Definition 7 and recalling that $r$ is the root of the Zielonka tree $\mathcal{Z}_\varphi$, the solution of the variable $X_r$ is the winning region of the existential player in the Emerson-Lei game $G$.*

By Theorem 4, it suffices to mutually transform winning strategies in $P_G$ and the fixpoint game $G_{S_\varphi}$ for the equation system $S_\varphi$ from Definition 7.

Given the fixpoint characterization of winning regions in Emerson-Lei games with objective $\varphi$ in Definition 7, we obtain a fixpoint iteration algorithm that computes the solution of Emerson-Lei games. The algorithm is by nature open to symbolic implementation. The main function is recursive, taking as input one vertex $s \in T$ of the Zielonka tree $\mathcal{Z}_\varphi$ and a list $l$ of subsets of the set $V$ of nodes, and returns a subset of $V$ as result. For calls SOLVE$(s, ls)$, we require that the argument list $ls$ contains exactly one subset $X_{s'}$ of $V$ for each ancestor $s'$ of $s$ in the Zielonka tree (with $s' < s$).

**Algorithm 1** SOLVE$(s, ls)$

```
if $s \in T_\bigcirc$ then $X_s \leftarrow \emptyset$ else $X_s \leftarrow V$        ▷ Initialize variable $X_s$ for lfp/gfp
$W \leftarrow V \setminus X_s$
while $X_s \neq W$ do                                        ▷ Compute fixpoint
    $W \leftarrow X_s$
    if $R(s) \neq \emptyset$ then                             ▷ Case: $s$ is not a leaf in $\mathcal{Z}_\varphi$
        for $t \in R(s)$ do
            $U \leftarrow$ SOLVE$(t, ls : W)$                  ▷ Recursively solve for $t$
            if $s \in T_\bigcirc$ then $X_s \leftarrow X_s \cup U$
                            else $X_s \leftarrow X_s \cap U$
        end for
    else                                                      ▷ Case: $s$ is a leaf in $\mathcal{Z}_\varphi$
        $Y \leftarrow \emptyset$
        for $t \leq s$ do
            $U \leftarrow \mathsf{anc}_s^t \cap \mathsf{CPre}((ls : W)(t))$    ▷ Compute one-step attraction w.r.t. $s$
            $Y \leftarrow Y \cup U$
        end for
        $X_s \leftarrow Y$
    end if
end while
return $X_s$                                  ▷ Return stabilized set $X_s$ as result
```

**Lemma 10.** *For all $v \in V$, we have $v \in [\![X_r]\!]$ if and only if $v \in \textsc{Solve}(r, [])$.*

*Proof (Sketch).* The algorithm computes the solution of the equation system by standard Kleene-approximation for nested least and greatest fixpoints.

**Lemma 11.** *Given an Emerson-Lei game $(V, V_\exists, V_\forall, E, \alpha_{\gamma,\varphi})$ with set of colors $C$ and induced Zielonka tree $\mathcal{Z}_\varphi$, the solution $[\![X_r]\!]$ of the equation system $S_\varphi$ from Definition 7 can be computed in time $\mathcal{O}(|\mathcal{Z}_\varphi| \cdot |E| \cdot |V|^k)$, where $k \leq |C|$ denotes the height of $\mathcal{Z}_\varphi$.*

Combining Theorem 9 with Lemmas 3, 10 and 11 we obtain

**Corollary 12.** *Solving Emerson-Lei games with $n$ nodes, $m$ edges and $k$ colors can be implemented symbolically to run in time $\mathcal{O}(k! \cdot m \cdot n^k)$; the resulting strategies require memory at most $e \cdot k!$.*

*Remark 13.* Strategy extraction works as follows. The algorithm computes a set $[\![X_t]\!]$ for each Zielonka tree vertex $t \in \mathcal{Z}_\varphi$. Furthermore it yields, for each non-leaf vertex $s \in T_\bigcirc$ and each $v \in [\![X_s]\!]$, a single child vertex $\mathsf{choice}(v, s) \in R(s)$ of $s$ such that $v \in [\![X_{\mathsf{choice}(v,s)}]\!]$. The algorithm also yields, for each leaf vertex $t$ and each $v \in V_\exists \cap [\![X_t]\!]$, a single game move $\mathsf{move}(v, t)$. All these choices together constitute a winning strategy for existential player in the parity game $P_G$. We define a strategy for the Emerson-Lei game that uses leaves of the Zielonka tree as memory values, following the ideas used in the construction of $P_G$; the strategy moves, from a node $v \in V_\exists$ and having memory content $m$, to the node $\mathsf{move}(v, m)$. As initial memory value we pick some leaf of $\mathcal{Z}_\varphi$ that $\mathsf{choice}$ associates with the initial node in $G$. To update memory value $m$ according to visiting game node $v$, we first take the anchor vertex $s$ of $m$ and $v$. Then we pick the next memory value $m$ to be some leaf below $s$ that can be reached by talking the choices $\mathsf{choice}(v, s')$ for every vertex $s' \in T_\bigcirc$ passed along the way from $s$ to the leaf; if $s \in T_\Box$, then we additionally require the following: let $q = |R(s)|$, let $o$ be the number such that $m$ is a leaf below the $o$-th child of $s$, and put $j = o + 1 \mod q$; then we require that $m'$ is a leaf below the $j$-th child of $s$. By the correctness of the algorithm, the constructed strategy is a winning strategy.

Dziembowski et al. have shown that winning strategies can be extracted by using a walk through the Zielonka tree that requires memory only for the branching at winning vertices [18]. This yields, for instance, memoryless strategies for games with Rabin objectives, for which branching in the associated Zielonka trees takes place at losing vertices. Adapting the strategy extraction in our setting to this more economic method is straight-forward but notation-heavy, so we omit a more precise analysis of strategy size here.

Our algorithm hence can be implemented to run in time $2^{\mathcal{O}(k \log n)}$ for games with $n$ nodes and $k \leq n$ colors, improving upon the bound $2^{\mathcal{O}(n^2)}$ stated in [25], where the authors only consider the case that every game node has a distinct color, implying $n = k$. We note that the later appearance record construction used in [25] is known to be hard to represent symbolically. Our fixpoint characterization generalizes previously known algorithms for e.g. parity games [8], and

Streett and Rabin games [36], recovering previously known bounds on worst-case running time of fixpoint iteration algorithms for these types of games.

While it has recently been shown that parity games can be solved in quasipolynomial time [9], we note that in the case of parity objectives, our algorithm is not immediately quasipolynomial. However, there are quasipolynomial methods for solving nested fixpoints [24,2] (with the latter being open to symbolic implementation); in the case of parity objectives, these more involved algorithms can be used in place of fixpoint iteration to solve our equation system and recover the quasipolynomial bound. The precise complexity of using quasipolynomial methods for solving fixpoint equation systems beyond parity conditions is subject to ongoing research.

## 4 Synthesis for Safety and Emerson-Lei LTL

In this section we present an application of the results from Section 3. We introduce the safety and Emerson-Lei fragment of LTL and show that synthesis for this fragment can be reasoned about symbolically. The idea for safety and Emerson-Lei LTL synthesis is twofold: first, consider only the safety part and create a symbolic arena capturing its satisfaction. Second, play a game on this arena by adding the Emerson-Lei part as a winning condition. Finally we use the results from the previous sections to solve the game symbolically.

### 4.1 Safety LTL and Symbolic Safety Automata

We start by defining safety LTL, symbolic safety automata, and recalling known results about those.

**Definition 14 (LTL and Safety LTL [45]).** *Given a non-empty set* $\mathsf{AP}$ *of atomic propositions, the general syntax for LTL formulas is as follows:*

$$\varphi := \top \mid \bot \mid p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid X\varphi \mid \varphi_1 U \varphi_2 \qquad p \in \mathsf{AP}.$$

*Standard abbreviations are defined as follows:* $\varphi_1 R \varphi_2 := \neg(\neg\varphi_1 U \neg\varphi_2)$*,* $F\varphi := \top U \varphi$*, and* $G\varphi := \neg F \neg \varphi$*. We define the satisfaction relation* $\models$ *for a formula* $\varphi$ *and its language* $\mathcal{L}(\varphi)$ *as usual.*

*An LTL formula is said to be a* safety formula *if it is in negative normal form (i.e. all negations are pushed to atomic propositions) and only uses* $X, R, G$ *as temporal operators (i.e. no* $U$ *or* $F$ *are allowed).*

It is a safety formula in the sense that every word that does not satisfy the formula has a finite prefix that already falsifies the formula. In other words, such a formula is satisfied as long as "bad states" are avoided forever.

**Definition 15 (Symbolic Safety Automata).** *A symbolic safety automaton is a tuple* $\mathcal{A} = (2^{\mathsf{AP}}, V, T, \theta_0)$ *where* $V$ *is a set of variables,* $T(V, V', \mathsf{AP})$ *is the transition assertion, and* $\theta_0(V)$ *is the initialization assertion. A run of* $\mathcal{A}$ *on* *the word $w \in (2^{\mathsf{AP}})^\omega$ is a sequence $\rho = s_0 s_1 \dots$ where the $s_i \in 2^V$ are variable assignments such that 1. $s_0 \models \theta_0$, and 2. for all $i \geq 0$, $(s_i, s_{i+1}, w(i)) \models T$. A word $w$ is in $\mathcal{L}(\mathcal{A})$ if and only if there is an infinite run of $\mathcal{A}$ on $w$. $\mathcal{A}$ is deterministic if for all words $w \in (2^{\mathsf{AP}})^\omega$ there is at most one run of $\mathcal{A}$ on $w$.*

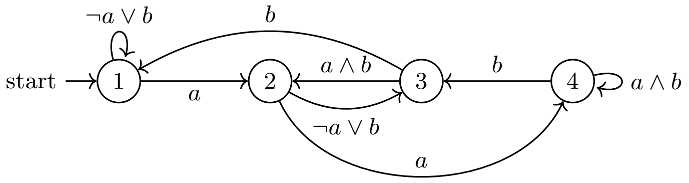
Kupferman and Vardi show how to convert a safety LTL formula into an equivalent deterministic symbolic safety automaton [30].

**Lemma 16.** *A safety LTL formula $\varphi$ can be translated to a deterministic symbolic safety automaton $\mathcal{D}_{\mathsf{symb}}$ accepting the same language, with $|\mathcal{D}_{\mathsf{symb}}| = 2^{|\varphi|}$.*

The idea is to first convert $\varphi$ to a (non-symbolic) non-deterministic safety automaton $\mathcal{N}_\varphi$, which is of size exponential of the size of the formula, and then symbolically determinize $\mathcal{N}_\varphi$ by a standard subset construction to obtain $\mathcal{D}_{\mathsf{symb}}$. Note that while the size of $\mathcal{D}_{\mathsf{symb}}$ is only exponential in the size of the formula, its state space would be double exponential when fully expanded.

*Example 17.* Let $\varphi = G(b \vee c) \wedge G(a \to b \vee XXb)$ be a safety LTL formula over $\mathsf{AP} = \{a, b, c\}$. An execution satisfying $\varphi$ must have at least one of $b$ or $c$ at every step, moreover every $a$ sees a $b$ present at the same step or two steps afterwards.

As an intermediate step towards building the equivalent $\mathcal{D}_{\mathsf{symb}}$, we first present below a corresponding non-deterministic safety automaton $\mathcal{N}_\varphi$.



For the sake of presentation, we use Boolean combinations of $\mathsf{AP}$ in transitions instead of labeling them with elements of $2^{\mathsf{AP}}$, with the intended meaning that $s \xrightarrow{\psi} s' = \{s \xrightarrow{C} s' \mid C \in 2^{\mathsf{AP}},\ C \models \psi\}$. We also omit the $G(b \vee c)$ part of the formula in the construction. One can simply append $\dots \wedge (b \vee c)$ to every transition of $\mathcal{N}_\varphi$ to get back the original formula. Intuitively state 1 correspond to not seeing an $a$, state 2 means that a $b$ must be seen at the next step, state 3 means that there must be a $b$ now, and state 4 that $b$ is needed now and next as well.

Then the symbolic safety automaton is $\mathcal{D}_{\mathsf{symb}} = (2^{\mathsf{AP}}, V, T, \theta_0)$ with:

- $V = \{v_1, v_2, v_3, v_4\}$ are the variables corresponding to the four states of $\mathcal{N}_\varphi$,
- $\theta_0 = v_1 \wedge \neg v_2 \wedge \neg v_3 \wedge \neg v_4$ asserts that only the state $v_1$ is initial,
- The transition assertion is $T = (v_1' \leftrightarrow (v_1 \wedge (\neg a \vee b)) \vee (v_3 \wedge b)) \wedge (v_2' \leftrightarrow (v_1 \wedge a) \vee (v_3 \wedge (a \wedge b))) \wedge (v_3' \leftrightarrow (v_2 \wedge (\neg a \vee b)) \vee (v_4 \wedge b)) \wedge (v_4' \leftrightarrow (v_2 \wedge a) \vee (v_4 \wedge (a \wedge b))) \wedge (v_1 \vee v_2 \vee v_3 \vee v_4)$.

Determinizing $\mathcal{N}_\varphi$ enumeratively would give an automaton with 9 states (see Example 23).

*Remark 18.* Restricting attention to safety LTL enables the two advantages mentioned above with respect to determinization. First, subset construction suffices (as observed also in [46]), avoiding the more complex Büchi determinization. Second, this construction, due to its simplicity, can be implemented symbolically. Interestingly, recent implementations of the synthesis from LTL$_f$ [46] or from safety LTL [45] have used indirect approaches for obtaining deterministic automata. For example, by translating LTL to first order logic and applying the tool MONA to the results [45,46], or by concentrating on minimization of deterministic automata [42]. The direct construction is similar to approaches used for checking universality of nondeterministic finite automata [42] or SAT-based bounded model checking [1]. We are not aware of uses of this direct implementation of the subset construction in reactive synthesis. The worst case complexity of this part is doubly-exponential, which, just like for LTL and LTL$_f$, cannot be avoided [43,3].

## 4.2 Symbolic Games

We use *symbolic game structures* to represent a certain class of games. Formally, a *symbolic game structure* $\mathcal{G} = \langle \mathcal{V}, \mathcal{X}, \mathcal{Y}, \theta_\exists, \rho_\exists, \varphi \rangle$ consists of:

- $\mathcal{V} = \{v_1, \ldots, v_n\}$ : A finite set of typed *variables* over finite domains. Without loss of generality, we assume they are all Boolean. A node $s$ is an valuation of $\mathcal{V}$, assigning to each variable $v_i \in \mathcal{V}$ a value $s[v_i] \in \{0, 1\}$. Let $\Sigma$ be the set of nodes.
  We extend the evaluation function $s[\cdot]$ to Boolean expressions over $\mathcal{V}$ in the usual way. An *assertion* is a Boolean formula over $\mathcal{V}$. A node $s$ satisfies an assertion $\varphi$ denoted $s \models \varphi$, if $s[\varphi] = \mathbf{true}$. We say that $s$ is a $\varphi$-node if $s \models \varphi$.
- $\mathcal{X} \subseteq \mathcal{V}$ is a set of *input variables*. These are variables controlled by the universal player. Let $\Sigma_\mathcal{X}$ denote the possible valuations to variables in $\mathcal{X}$.
- $\mathcal{Y} = \mathcal{V} \setminus \mathcal{X}$ is a set of *output variables*. These are variables controlled by the existential player. Let $\Sigma_\mathcal{Y}$ denote the possible valuations to variables in $\mathcal{Y}$.
- $\theta_\exists(\mathcal{X}, \mathcal{Y})$ is an assertion characterizing the initial condition.
- $\rho_\exists(\mathcal{V}, \mathcal{X}', \mathcal{Y}')$ is the transition relation. This is an assertion relating a node $s \in \Sigma$ and an input value $s_\mathcal{X} \in \Sigma_\mathcal{X}$ to an output value $s_\mathcal{Y} \in \Sigma_\mathcal{Y}$ by referring to primed and unprimed copies of $\mathcal{V}$. The transition relation $\rho_\exists$ identifies a valuation $s_\mathcal{Y} \in \Sigma_\mathcal{Y}$ as a *possible output* in node $s$ reading input $s_\mathcal{X}$ if $(s, (s_\mathcal{X}, s_\mathcal{Y})) \models \rho_\exists$, where $s$ is the assignment to variables in $\mathcal{V}$ and $s_\mathcal{X}$ and $s_\mathcal{Y}$ are the assignment to variables in $\mathcal{V}'$ induced by $(s_\mathcal{X}, s_\mathcal{Y}) \in \Sigma$.
- $\varphi$ is the winning condition, given by an LTL formula.

For two nodes $s$ and $s'$ of $\mathcal{G}$, $s'$ is a *successor* of $s$ if $(s, s') \models \rho_\exists$.

A symbolic game structure $\mathcal{G}$ defines an arena $A_\mathcal{G}$, where $V_\forall = \Sigma$, $V_\exists = \Sigma \times \Sigma_\mathcal{X}$, and $E$ is defined as follows:

$$E = \{(s, (s, s_\mathcal{X})) \mid s \in \Sigma \text{ and } s_\mathcal{X} \in \Sigma_\mathcal{X}\} \cup \{((s, s_\mathcal{X}), (s_\mathcal{X}, s_\mathcal{Y})) \mid (s, (s_\mathcal{X}, s_\mathcal{Y})) \models \rho_\exists\}.$$

When reasoning about symbolic game structures we ignore the intermediate visits to $V_\exists$. Indeed, they add no information as they can be deduced from the nodes in $V_\forall$ preceding and following them. Thus, a play $\pi = s_0 s_1 \dots$ is *winning for the existential player* if $\sigma$ is infinite and satisfies $\varphi$. Otherwise, $\sigma$ is *winning for the universal player*.

The notion of strategy and winning region is trivially generalized from games to symbolic game structures. When needed, we treat $W_\exists$ (the set of nodes winning for the existential player) as an assertion. We define winning in the *entire* game structure by incorporating the initial assertion: a game structure $\mathcal{G}$ is said to be *won* by the existential player, if for all $s_\mathcal{X} \in \Sigma_\mathcal{X}$ there exists $s_\mathcal{Y} \in \Sigma_\mathcal{Y}$ such that $(s_\mathcal{X}, s_\mathcal{Y}) \models \theta_\exists \wedge W_\exists$.

## 4.3 Realizability and Synthesis

Let $\varphi$ be an LTL formula over input and output variables $I$ and $O$, controlled by *the environment* and *the system*, respectively (the universal and the existential player, respectively).

The reactive synthesis problem asks whether there is a strategy for the system of the form $\sigma : (2^I)^+ \to 2^O$ such that for all sequences $x_0 x_1 \dots \in (2^I)^\omega$ we have:

$$(x_0 \cup \sigma(x_0))(x_1 \cup \sigma(x_0 x_1)) \dots \models \varphi$$

If there is such a strategy we say that $\varphi$ is *realizable* [38].

Equivalently, $\varphi$ is *realizable* if the system is winning in the symbolic game $\mathcal{G}_\varphi = \langle I \cup O, I, O, \top, \top, \varphi \rangle$ with $I$ for input variables $\mathcal{X}$ and $O$ for output $\mathcal{Y}$.

**Theorem 19.** *[38] Given an LTL formula $\varphi$, the realizability of $\varphi$ can be determined in doubly exponential time. The problem is 2EXPTIME-complete.*

The game $\mathcal{G}_\varphi$ above uses neither the initial condition nor the system transition. Conversely, consider a symbolic game $\mathcal{G} = \langle \mathcal{V}, \mathcal{X}, \mathcal{Y}, \theta_\exists, \rho_\exists, \varphi \rangle$:

**Theorem 20.** *[7] The system wins in $\mathcal{G}$ iff $\varphi_\mathcal{G} = \theta_\exists \wedge G\rho_\exists \wedge \varphi$ is realizable.*[2][3]

## 4.4 Safety and Emerson-Lei Synthesis

We now define the class of LTL formulas that are supported by our technique and show how to construct appropriate games capturing their realizability problem.

For $\psi \in \mathbb{B}(\mathsf{AP})$, let $\mathsf{Inf}\,\psi := GF\psi$ and $\mathsf{Fin}\,\psi := FG\neg\psi = \neg\mathsf{Inf}\,\psi$. The *Emerson-Lei fragment* of LTL consists of all formulas that are positive Boolean combinations of $\mathsf{Inf}\,\psi$ and $\mathsf{Fin}\,\psi$ for all Boolean formulas $\psi$ over atomic propositions. The satisfaction of such formulas depends only on the set of letters (truth assignments to propositions) appearing infinitely often in a word.

[2] Technically, $\rho_\exists$ contains primed variables and is not an LTL formula. This can be easily handled by using the next operator $X$. We thus ignore this issue.

[3] We note that Bloem et al. consider more general games, where the environment also has an initial assertion and a transition relation. Our games are obtained from theirs by setting the initial assertion and the transition relation of the environment to true.

*Remark 21.* The Emerson-Lei fragment easily accommodates various liveness properties that cannot be encoded in smaller fragments such as GR[1]. One prominent example for this is the property of *stability* (as encoded by LTL formulas of the shape $FG\ p$), which appears frequently as a guarantee in usage of synthesis for robotics and control (see, e.g., the work of Ehlers [19] and Ozay [32]), and commonly is approximated in GR[1] but, as a guarantee or as part of a specification, cannot be captured exactly in the game context. Another important example is *strong fairness* (as encoded by LTL formulas of the shape $\bigwedge_i (GF\ r_i \rightarrow GF\ g_i)$) which allows to capture the exact relation between cause and effect. Particularly, in GR[1] only if *all* "resources" are available infinitely often there is an obligation on the system to supply *all* its "guarantees". In contrast, strong fairness allows to connect particular resources to particular supplied guarantees. Ongoing studies on fairness assumptions that arise from the abstraction of continuous state spaces to discrete state spaces [32,33] provide further examples of fairness assumptions that can be expressed in EL but not in GR[1]. Emerson-Lei liveness allows free combination of all properties mentioned above and more.

**Definition 22.** *The* Safety and Emerson-Lei fragment *is the set of formulas of the form $\varphi = \varphi_{\text{safety}} \wedge \varphi_{\text{EL}}$, where $\varphi_{\text{safety}}$ is a safety formula and $\varphi_{\text{EL}}$ is in the Emerson-Lei fragment.*

We assume a partition $\mathsf{AP} = I \uplus O$ where $I$ is a set of *input propositions* and $O$ a set of *output propositions*, both non-empty. Let $\varphi = \varphi_{\text{safety}} \wedge \varphi_{\text{EL}}$ be a safety and Emerson-Lei formula over $\mathsf{AP}$, and let $\mathcal{D}_{\mathsf{symb}} = (2^{\mathsf{AP}}, V, T, \theta_0)$ be the symbolic deterministic safety automaton associated to $\varphi_{\text{safety}}$. We construct $G_\varphi = \langle V \uplus \mathsf{AP}, I, O \uplus V, \theta_0, T, \varphi_{\text{EL}} \rangle$, thus $\mathcal{X} = I$ and $\mathcal{Y} = O \uplus V$.

*Example 23.* Let $\varphi_{\text{safety}} = G(b \vee c) \wedge G(a \rightarrow b \vee XXb)$, our running safety example from Example 17 with its associated symbolic deterministic automaton. Partition $\mathsf{AP}$ into $I = \{a\}$ and $O = \{b, c\}$. We depict the arena of the game $G_\varphi$ (independent of the formula $\varphi_{\text{EL}}$ that is yet to be defined) in Figure 23.

To keep the illustration readable and keep it from getting too large, a few modifications to the formal arena definition have been made. First, $c$ labels on edges have been omitted: every transition labeled with $b$ represent two transitions with sets $\{b\}$ and $\{b, c\}$, while transitions labeled with $\neg b$ stand for a single transition with set $\{c\}$ (due to the $G(b \vee c)$ requirement forbidding $\emptyset$). Similarly, existential nodes have been omitted when all choices for the existential player lead to the same destination. Instead, the universal and existential moves have been combined in one transition: $a; *$ for an $a$ followed by some existential move, and $a; b$ for when an $a$ requires the existential player to play $b$ (with or without $c$, as above). Finally, states are only labeled with variables from $V$ and not $\mathsf{AP}$, the latter is used to label edges instead. For a fully state-based labeling arena, states would have to store the last move, leading to various duplicate states.

Note that this game arena is given only for illustration purposes, as we want to solve the symbolic game without explicitly enumerating all its states and transitions like here.

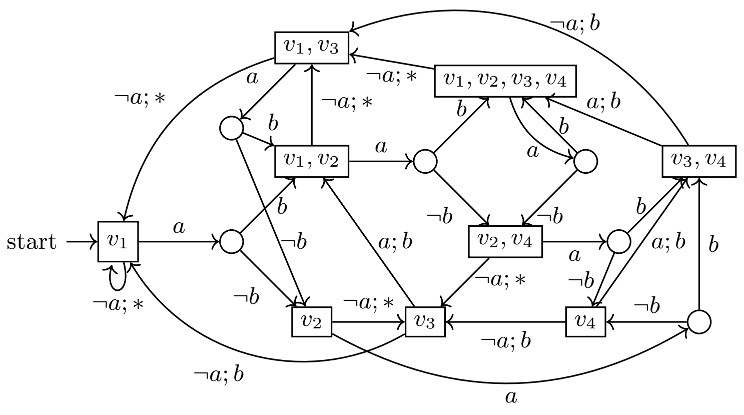


**Fig. 1.** Game arena for $G_\varphi$

**Lemma 24.** *The system wins $G_\varphi$ if and only if $\varphi$ is realizable.*

Next we detail how to solve the symbolic game $G_\varphi$ by using the results from Section 3.

**Lemma 25.** *Given a symbolic game $G = \langle \mathcal{V}, \mathcal{X}, \mathcal{Y}, \theta_\exists, \rho_\exists, \varphi \rangle$ such that $\varphi$ is an Emerson-Lei formula with set of colors*

$$C = \{\psi \in \mathbb{B}(\mathsf{AP}) \mid \psi \textit{ is a subformula of } \varphi\},$$

*the winning region $W_\exists$ of $G$ is characterized by the equation system from Definition 7, using the assertion*

$$\mathsf{CPre}(S) = \forall s_\mathcal{X} \in \Sigma_\mathcal{X}. \exists s_\mathcal{Y} \in \Sigma_\mathcal{Y}. S' \wedge (v, s_\mathcal{X}, s_\mathcal{Y}) \models \rho_\exists.$$

The proof of this lemma is by straightforward adaptation of the proof of Theorem 9 to the symbolic setting, following the relation between symbolic game structures and game arenas described above.

Finally, this gives us a procedure to solve the synthesis problem for safety and Emerson-Lei LTL.

**Theorem 26.** *The realizability of a formula $\varphi = \varphi_{\mathsf{safety}} \wedge \varphi_{EL}$ of the Safety and Emerson-Lei fragment of LTL can be checked in time $2^{\mathcal{O}(m \cdot \log m \cdot 2^n)}$, where $n = |\varphi_{\mathsf{safety}}|$ and $m = |\varphi_{EL}|$. Realizable formulas can be realized by systems of size at most $2^{2^n} \cdot e \cdot m!$.*

*Proof.* Using the construction described in this section,we obtain the symbolic game $G_\varphi$ of size $q = 2^{2^n}$ with winning condition $\varphi_{EL}$, using at most $m$ colors; by Theorem 24, this game characterizes realizibility of the formula. Using the results from the previous section, $G_\varphi$ can be solved in time $\mathcal{O}(m! \cdot q^2 \cdot q^m) \in \mathcal{O}(2^{m \log m} \cdot 2^{(m+2)2^n}) \in 2^{\mathcal{O}(m \cdot \log m \cdot 2^n)}$, resulting in winning strategies with memory at most $e \cdot m!$.

Both the automata determinization and the game solving can be implemented symbolically.

*Example 27.* To illustrate the overall synthesis method, we consider the game that is obtained by combining the game arena $G_{\varphi_{\mathsf{safety}}}$ from Example 23 with the winning objective $\varphi_{EL} = (\mathsf{Fin}\ a \vee \mathsf{Inf}\ b) \wedge (\mathsf{Fin}\ a \vee \mathsf{Fin} d) \wedge \mathsf{Inf}\ c$ from Example 2.3, where we instantiate the label $d$ to nodes satisfying $b \wedge c$ thus creating a game-specific dependency between the colors. Solving this game amounts to solving the equation system shown in Example 8.3. However, with the interpretation of $d = b \wedge c$, some of the conditions become simpler. For example, $\neg a \wedge \neg b \wedge \neg c \wedge \neg d$ becomes $\neg a \wedge \neg b \wedge \neg c$ and $b \wedge \neg d$ becomes $b \wedge \neg c$. It turns out that the system player wins the node $v_1$. Intuitively, the system can play $\{c\}$ whenever possible and thereby guarantee satisfaction of $\varphi_{EL}$. We extract this strategy from the computed solution of the equation system in Example 2.3 as described in Remark 13. E.g. for partial runs $\pi$ that end in $v_1$ and for which the last leaf vertex in the induced walk $\rho_\pi$ through $\mathcal{Z}_\varphi$ is the vertex 8, the system can react by playing $\{b\}$, $\{c\}$, or even $\{b, c\}$ whenever the environment plays $\emptyset$. The move $\{b\}$ continues the induced walk $\rho_\pi$ through vertex 2 to the leaf vertex 5; similarly, the move $\{b, c\}$ continues $\rho_\pi$ through the vertex 1 to the leaf vertex 6. The strategy construction gives precedence to the choice that leads through the lowest vertex in the Zielonka tree, which in this case means picking the move $\{c\}$ that continues $\rho_\pi$ through the vertex 7 to the leaf 8. Proceeding similarly for all other combinations of game nodes and vertices in the Zielonka tree, one obtains a strategy $\sigma$ for the system that always outputs singleton letters, giving precedence to $\{c\}$ whenever possible. To see that $\sigma$ is a winning strategy, let $\pi$ be a play that is compatible with $\sigma$. If $\pi$ eventually loops at $v_1$ forever, then $s_\pi$ is the existential vertex 7 and the existential player wins the play since it satisfies both $\mathsf{Fin}\ a$ and $\mathsf{Inf}\ c$. Any other play $\pi$ satisfies $\mathsf{Inf}\ a$, $\mathsf{Inf}\ b$ and $\mathsf{Inf}\ c$ since all cycles that are compatible with $\sigma$ (excluding the loop at $v_1$) contain at least one $a$-edge, at least one $b$-edge and also at least one $c$-edge that is prescribed by the strategy $\sigma$. For these plays, $\rho_\pi$ eventually reaches the vertex 2. Since the system always plays singleton letters (so that $\pi$ in particular satisfies $\mathsf{Fin}(b \wedge c)$), the vertex 1 is not visited again by $\rho_\pi$, once vertex 2 has been reached. Hence the dominating vertex for such plays is $s_\pi = 2$, an existential vertex.

## 4.5 Synthesis Extensions and Optimizations

We have chosen to use safety-LTL as the safety part of the Safety-EL fragment to showcase the options opened by having symbolic algorithms for the analysis of very expressive liveness conditions. The crucial feature of the safety fragment is the ability to convert that part of the specification to a symbolic deterministic automaton. It is important to note that *every* fragment of LTL (or $\omega$-regular in general) that can be easily converted to a symbolic deterministic automaton can be incorporated and handled with the same machinery. For example, it was suggested to extend the expressiveness of GR[1] by including deterministic automata in the safety part of the game and referring to their states in the liveness part [7]. Past LTL [31] can be handled in the same way in that it is incorporated for GR[1] [7]. An extreme example is GR-EBR, where safety parts are allowed to use bounded future and pure past, which still allows the symbolic treatment [15]. All of these alternatives can be incorporated in the safety part with no changes to our overall methodology. Unlike previous cases, if there is an easy translation to deterministic symbolic automata *with a non-trivial winning condition*, these can be incorporated as well with the EL part extended to handle their winning condition as well. We could consider also extensions to the liveness parts. For example, by using past LTL or reference to states of additional symbolic deterministic automata. The Boolean state formulas appearing as part of the EL condition can be replaced by formulas allowing one usage of the next operator, as in [39,19]. The generalization to handle transition-based EL games, which would be required in that case, rather than state-based EL games is straight-forward.

As the formulas we consider are conjunctions, optimizations can be applied to both conjuncts independently. This subsumes, for example, analyzing the winning region in a safety game prior to the full analysis [29,7,5], reductions in the size of nondeterministic automata [17], or symbolic minimization of deterministic automata [16].[4]

## 5 Conclusions and Future Work

We provide a symbolic algorithm to solve games with Emerson-Lei winning conditions. Our solution is based on an encoding of the Zielonka tree of the winning condition in a system of fixpoint equations. In case of known winning conditions, our algorithm recovers known algorithms and complexity results. As an application of this algorithm, we suggest an expressive fragment of LTL for which realizability can be reasoned about symbolically. Formulas in our fragment are conjunctions between an LTL safety formula and an Emerson-Lei liveness condition. This fragment is more general than, e.g., GR[1].

In the future, we believe that analysis of the Emerson-Lei part can reduce the size of Zielonka trees (and thus the symbolic algorithm). This can be done either through analysis and simplification of the LTL formula, e.g., [26], by means of alternating-cycle decomposition [12,13], or by analyzing the semantic meaning of colors. We would also like to implement the proposed overall synthesis method.

## References

1. Armoni, R., Egorov, S., Fraer, R., Korchemny, D., Vardi, M.Y.: Efficient LTL compilation for sat-based model checking. In: International Conference on Computer-Aided Design. pp. 877–884. IEEE Computer Society (2005). https://doi.org/10.1109/ICCAD.2005.1560185, `https://doi.org/10.1109/ICCAD.2005.1560185`

[4] Notice that explicit minimization as done, e.g., in [30] would require to explicitly construct the potentially doubly exponential deterministic automaton, nullifying the entire effort to keep all analysis symbolic.

2. Arnold, A., Niwinski, D., Parys, P.: A quasi-polynomial black-box algorithm for fixed point evaluation. In: Baier, C., Goubault-Larrecq, J. (eds.) 29th EACSL Annual Conference on Computer Science Logic, CSL 2021, January 25-28, 2021, Ljubljana, Slovenia (Virtual Conference). LIPIcs, vol. 183, pp. 9:1–9:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2021). https://doi.org/10.4230/LIPIcs.CSL.2021.9, `https://doi.org/10.4230/LIPIcs.CSL.2021.9`
3. Artale, A., Geatti, L., Gigante, N., Mazzullo, A., Montanari, A.: Complexity of safety and cosafety fragments of linear temporal logic. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 37, pp. 6236–6244 (2023)
4. Baldan, P., König, B., Mika-Michalski, C., Padoan, T.: Fixpoint games on continuous lattices. Proc. ACM Program. Lang. **3**(POPL), 26:1–26:29 (2019). https://doi.org/10.1145/3290339
5. Bansal, S., Giacomo, G.D., Stasio, A.D., Li, Y., Vardi, M.Y., Zhu, S.: Compositional safety LTL synthesis. In: 14th International Conference on Verified Software, Theories, Tools and Experiments. Lecture Notes in Computer Science, vol. 13800, pp. 1–19. Springer (2022). https://doi.org/10.1007/978-3-031-25803-9_1, `https://doi.org/10.1007/978-3-031-25803-9_1`
6. Bhatia, A., Maly, M.R., Kavraki, L.E., Vardi, M.Y.: Motion planning with complex goals. IEEE Robotics Autom. Mag. **18**(3), 55–64 (2011). https://doi.org/10.1109/MRA.2011.942115
7. Bloem, R., Jobstmann, B., Piterman, N., Pnueli, A., Sa'ar, Y.: Synthesis of reactive(1) designs. J. Comput. Syst. Sci. **78**(3), 911–938 (2012). https://doi.org/10.1016/j.jcss.2011.08.007
8. Bruse, F., Falk, M., Lange, M.: The fixpoint-iteration algorithm for parity games. In: International Symposium on Games, Automata, Logics and Formal Verification, GandALF 2014. EPTCS, vol. 161, pp. 116–130 (2014). https://doi.org/10.4204/EPTCS.161.12
9. Calude, C., Jain, S., Khoussainov, B., Li, W., Stephan, F.: Deciding parity games in quasipolynomial time. In: Theory of Computing, STOC 2017. pp. 252–263. ACM (2017)
10. Camacho, A., McIlraith, S.A.: Learning interpretable models expressed in linear temporal logic. In: Twenty-Ninth International Conference on Automated Planning and Scheduling. pp. 621–630. AAAI Press (2019). https://doi.org/10.1609/icaps.v29i1.3529
11. Camacho, A., Triantafillou, E., Muise, C.J., Baier, J.A., McIlraith, S.A.: Non-deterministic planning with temporally extended goals: LTL over finite and infinite traces. In: Thirty-First AAAI Conference on Artificial Intelligence. pp. 3716–3724. AAAI Press (2017). https://doi.org/10.1609/aaai.v31i1.11058
12. Casares, A., Colcombet, T., Lehtinen, K.: On the size of good-for-games rabin automata and its link with the memory in muller games. In: Bojanczyk, M., Merelli, E., Woodruff, D.P. (eds.) International Colloquium on Automata, Languages, and Programming, ICALP 2022. LIPIcs, vol. 229, pp. 117:1–117:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2022). https://doi.org/10.4230/LIPIcs.ICALP.2022.117
13. Casares, A., Duret-Lutz, A., Meyer, K.J., Renkin, F., Sickert, S.: Practical applications of the alternating cycle decomposition. In: Fisman, D., Rosu, G. (eds.) Tools and Algorithms for the Construction and Analysis of Systems - 28th International Conference, TACAS 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany,

April 2-7, 2022, Proceedings, Part II. Lecture Notes in Computer Science, vol. 13244, pp. 99–117. Springer (2022). https://doi.org/10.1007/978-3-030-99527-0_6, https://doi.org/10.1007/978-3-030-99527-0_6

14. Church, A.: Logic, arithmetic, and automata. In: International Congress of Mathematicians. Institut Mittag-Leffler, Sweden (1963)
15. Cimatti, A., Geatti, L., Gigante, N., Montanari, A., Tonetta, S.: Fairness, assumptions, and guarantees for extended bounded response ltl+p synthesis. Software and System Modeling (2023). https://doi.org/10.1007/s10270-023-01122-4
16. D'Antoni, L., Veanes, M.: Minimization of symbolic automata. In: Symposium on Principles of Programming Languages (POPL). pp. 541–554. ACM (2014). https://doi.org/10.1145/2535838.2535849, https://doi.org/10.1145/2535838.2535849
17. Duret-Lutz, A., Renault, E., Colange, M., Renkin, F., Aisse, A.G., Schlehuber-Caissier, P., Medioni, T., Martin, A., Dubois, J., Gillard, C., Lauko, H.: From spot 2.0 to spot 2.10: What's new? In: 34th International Conference on Computer Aided Verification. Lecture Notes in Computer Science, vol. 13372, pp. 174–187. Springer (2022). https://doi.org/10.1007/978-3-031-13188-2_9
18. Dziembowski, S., Jurdzinski, M., Walukiewicz, I.: How much memory is needed to win infinite games? In: 12th Annual IEEE Symposium on Logic in Computer Science. pp. 99–110. IEEE Computer Society (1997). https://doi.org/10.1109/LICS.1997.614939
19. Ehlers, R.: Generalized rabin(1) synthesis with applications to robust system synthesis. In: Third International Symposium on NASA Formal Methods. Lecture Notes in Computer Science, vol. 6617, pp. 101–115. Springer (2011). https://doi.org/10.1007/978-3-642-20398-5_9
20. Ehlers, R.: Unbeast: Symbolic bounded synthesis. In: 17th International Conference on Tools and Algorithms for the Construction and Analysis of Systems. Lecture Notes in Computer Science, vol. 6605, pp. 272–275. Springer (2011). https://doi.org/10.1007/978-3-642-19835-9_25
21. Emerson, E.A., Lei, C.: Modalities for model checking: Branching time logic strikes back. Sci. Comput. Program. **8**(3), 275–306 (1987). https://doi.org/10.1016/0167-6423(87)90036-0, https://doi.org/10.1016/0167-6423(87)90036-0
22. Giacomo, G.D., Vardi, M.Y.: Synthesis for LTL and LDL on finite traces. In: Yang, Q., Wooldridge, M.J. (eds.) Twenty-Fourth International Joint Conference on Artificial Intelligence. pp. 1558–1564. AAAI Press (2015)
23. Hausmann, D., Lehaut, M., Piterman, N.: Symbolic solution of Emerson-Lei games for reactive synthesis. CoRR **abs/2305.02793** (2023), https://arxiv.org/abs/2305.02793
24. Hausmann, D., Schröder, L.: Quasipolynomial computation of nested fixpoints. In: Groote, J.F., Larsen, K.G. (eds.) Tools and Algorithms for the Construction and Analysis of Systems - 27th International Conference, TACAS 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27 - April 1, 2021, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12651, pp. 38–56. Springer (2021). https://doi.org/10.1007/978-3-030-72016-2_3, https://doi.org/10.1007/978-3-030-72016-2_3
25. Hunter, P., Dawar, A.: Complexity bounds for regular games. In: 30th International Symposium on Mathematical Foundations of Computer Science. Lecture Notes in Computer Science, vol. 3618, pp. 495–506. Springer (2005). https://doi.org/10.1007/11549345_43

26. John, T., Jantsch, S., Baier, C., Klüppelholz, S.: Determinization and limit-determinization of Emerson-Lei automata. In: 19th International Symposium on Automated Technology for Verification and Analysis. Lecture Notes in Computer Science, vol. 12971, pp. 15–31. Springer (2021). https://doi.org/10.1007/978-3-030-88885-5_2
27. John, T., Jantsch, S., Baier, C., Klüppelholz, S.: From emerson-lei automata to deterministic, limit-deterministic or good-for-mdp automata. Innov. Syst. Softw. Eng. **18**(3), 385–403 (2022). https://doi.org/10.1007/s11334-022-00445-7, https://doi.org/10.1007/s11334-022-00445-7
28. Kress-Gazit, H., Fainekos, G.E., Pappas, G.J.: Temporal-logic-based reactive mission and motion planning. IEEE Trans. Robotics **25**(6), 1370–1381 (2009). https://doi.org/10.1109/TRO.2009.2030225
29. Kugler, H., Segall, I.: Compositional synthesis of reactive systems from live sequence chart specifications. In: 15th International Conference on Tools and Algorithms for the Construction and Analysis of Systems. Lecture Notes in Computer Science, vol. 5505, pp. 77–91. Springer (2009). https://doi.org/10.1007/978-3-642-00768-2_9, https://doi.org/10.1007/978-3-642-00768-2_9
30. Kupferman, O., Vardi, M.Y.: Model checking of safety properties. Formal methods in system design **19**(3), 291–314 (2001). https://doi.org/10.1023/A:1011254632723
31. Lichtenstein, O., Pnueli, A., Zuck, L.D.: The glory of the past. In: Conference on Logics of Programs. Lecture Notes in Computer Science, vol. 193, pp. 196–218. Springer (1985). https://doi.org/10.1007/3-540-15648-8_16, https://doi.org/10.1007/3-540-15648-8_16
32. Liu, J., Ozay, N., Topcu, U., Murray, R.M.: Synthesis of reactive switching protocols from temporal logic specifications. IEEE Trans. Autom. Control. **58**(7), 1771–1785 (2013). https://doi.org/10.1109/TAC.2013.2246095
33. Majumdar, R., Schmuck, A.: Supervisory controller synthesis for nonterminating processes is an obliging game. IEEE Trans. Autom. Control. **68**(1), 385–392 (2023). https://doi.org/10.1109/TAC.2022.3143108
34. Moarref, S., Kress-Gazit, H.: Automated synthesis of decentralized controllers for robot swarms from high-level temporal logic specifications. Auton. Robots **44**(3-4), 585–600 (2020). https://doi.org/10.1007/s10514-019-09861-4
35. Müller, D., Sickert, S.: LTL to deterministic emerson-lei automata. In: Bouyer, P., Orlandini, A., Pietro, P.S. (eds.) Proceedings Eighth International Symposium on Games, Automata, Logics and Formal Verification, GandALF 2017, Roma, Italy, 20-22 September 2017. EPTCS, vol. 256, pp. 180–194 (2017). https://doi.org/10.4204/EPTCS.256.13, https://doi.org/10.4204/EPTCS.256.13
36. Piterman, N., Pnueli, A.: Faster solutions of rabin and streett games. In: 21th IEEE Symposium on Logic in Computer Science (LICS 2006), 12-15 August 2006, Seattle, WA, USA, Proceedings. pp. 275–284. IEEE Computer Society (2006). https://doi.org/10.1109/LICS.2006.23
37. Piterman, N., Pnueli, A., Sa'ar, Y.: Synthesis of reactive(1) designs. In: 7th International Conference on Verification, Model Checking, and Abstract Interpretation. Lecture Notes in Computer Science, vol. 3855, pp. 364–380. Springer (2006). https://doi.org/10.1007/11609773_24
38. Pnueli, A., Rosner, R.: On the synthesis of a reactive module. In: Sixteenth ACM Symposium on Principles of Programming Languages. pp. 179–190. ACM Press (1989). https://doi.org/10.1145/75277.75293

39. Raman, V., Piterman, N., Finucane, C., Kress-Gazit, H.: Timing semantics for abstraction and execution of synthesized high-level robot control. IEEE Trans. Robotics **31**(3), 591–604 (2015). https://doi.org/10.1109/TRO.2015.2414134, `https://doi.org/10.1109/TRO.2015.2414134`
40. Renkin, F., Duret-Lutz, A., Pommellet, A.: Practical "paritizing" of emerson-lei automata. In: Hung, D.V., Sokolsky, O. (eds.) Automated Technology for Verification and Analysis - 18th International Symposium, ATVA 2020, Hanoi, Vietnam, October 19-23, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12302, pp. 127–143. Springer (2020). https://doi.org/10.1007/978-3-030-59152-6_7, `https://doi.org/10.1007/978-3-030-59152-6_7`
41. Sohail, S., Somenzi, F.: Safety first: a two-stage algorithm for the synthesis of reactive systems. Int. J. Softw. Tools Technol. Transf. **15**(5-6), 433–454 (2013). https://doi.org/10.1007/s10009-012-0224-3
42. Tabakov, D., Vardi, M.Y.: Experimental evaluation of classical automata constructions. In: 12th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning. Lecture Notes in Computer Science, vol. 3835, pp. 396–411. Springer (2005). https://doi.org/10.1007/11591191_28, `https://doi.org/10.1007/11591191_28`
43. Vardi, M.Y., Stockmeyer, L.J.: Improved upper and lower bounds for modal logics of programs: Preliminary report. In: Proceedings of the 17th Annual ACM Symposium on Theory of Computing. pp. 240–251. ACM (1985)
44. Wongpiromsarn, T., Topcu, U., Murray, R.M.: Receding horizon temporal logic planning. IEEE Trans. Autom. Control. **57**(11), 2817–2830 (2012). https://doi.org/10.1109/TAC.2012.2195811
45. Zhu, S., Tabajara, L.M., Li, J., Pu, G., Vardi, M.Y.: A symbolic approach to safety LTL synthesis. In: 13th International Haifa Verification Conference: Hardware and Software - Verification and Testing. Lecture Notes in Computer Science, vol. 10629, pp. 147–162. Springer (2017). https://doi.org/10.1007/978-3-319-70389-3_10, `https://doi.org/10.1007/978-3-319-70389-3_10`
46. Zhu, S., Tabajara, L.M., Pu, G., Vardi, M.Y.: On the power of automata minimization in temporal synthesis. In: Proceedings 12th International Symposium on Games, Automata, Logics, and Formal Verification. EPTCS, vol. 346, pp. 117–134 (2021). https://doi.org/10.4204/EPTCS.346.8, `https://doi.org/10.4204/EPTCS.346.8`
47. Zielonka, W.: Infinite games on finitely coloured graphs with applications to automata on infinite trees. Theor. Comput. Sci. **200**(1-2), 135–183 (1998). https://doi.org/10.1016/S0304-3975(98)00009-7, `https://doi.org/10.1016/S0304-3975(98)00009-7`