Software and Systems Modeling
https://doi.org/10.1007/s10270-023-01122-4

SPECIAL SECTION PAPER

Fairness, assumptions, and guarantees for extended bounded response LTL+P synthesis

Alessandro Cimatti[1] · Luca Geatti[2] · Nicola Gigante[3] · Angelo Montanari[2] · Stefano Tonetta[1]

Received: 3 June 2022 / Revised: 19 July 2023 / Accepted: 25 July 2023

Abstract
Realizability and reactive synthesis from temporal logics are fundamental problems in formal verification. The complexity of these problems for linear temporal logic with past (LTL+P) led to the identification of fragments with lower complexities and simpler algorithms. Recently, the logic of extended bounded response LTL+P (LTL_{EBR}+P for short) has been introduced. It allows one to express safety languages definable in LTL+P and it is provided with an efficient, fully symbolic algorithm for reactive synthesis. This paper features four related contributions. First, we introduce GR-EBR, an extension of LTL_{EBR}+P with fairness conditions, assumptions, and guarantees that, on the one hand, allows one to express properties beyond the safety fragment and, on the other, it retains the efficiency of LTL_{EBR}+P in practice. Second, we the expressiveness of GR-EBR starting from the expressiveness of its fragments. In particular, we prove that: (1) LTL_{EBR}+P is expressively complete with respect to the safety fragment of LTL+P, (2) the removal of past operators from LTL_{EBR}+P results into a loss of expressive power, and (3) GR-EBR is expressively equivalent to the logic GR(1) of Bloem et al. Third, we provide a fully symbolic algorithm for the realizability problem from GR-EBR specifications, that reduces it to a number of safety subproblems. Fourth, to ensure soundness and completeness of the algorithm, we propose and exploit a general framework for safety reductions in the context of realizability of (fragments of) LTL+P. The experimental evaluation shows promising results.

Keywords Reactive synthesis · Temporal logics · Safety reductions · Expressiveness

1 Introduction

One of the most important problems in formal methods and requirement analysis is establishing whether a specification over a set of controllable and uncontrollable actions is *implementable* (or *realizable*), that is, whether there exists a controller that chooses the value of the controllable actions and satisfies the specification, no matter what the values of uncontrollable actions are [5, 6]. This problem has been formalized in the literature under the name of *realizability* [3]. The very close problem of *reactive synthesis* aims at synthesizing such a controller, whenever the specification is realizable. Usually, these problems are modelled as two-player games between Environment, who tries to violate the specification, and Controller, who tries to fulfill it. Realizability is known to have a very high worst-case complexity. In particular, it has a non-elementary lower bound for S1S specifications [4], and it is 2EXPTIME-complete for Linear Temporal Logic (LTL) specifications [5, 6].

In order to apply realizability and reactive synthesis in real-world scenarios, research has focused on the identifica-

Communicated by Antonio Cerone and Frank de Boer.

This work is an extension of the conference papers [1, 2].

✉ Luca Geatti
luca.geatti@uniud.it

Alessandro Cimatti
cimatti@fbk.eu

Nicola Gigante
gigante@inf.unibz.it

Angelo Montanari
angelo.montanari@uniud.it

Stefano Tonetta
tonettas@fbk.eu

[1] Embedded System Unit, Fondazione Bruno Kessler, Via Sommarive, 18, Povo, 38123 Trento, Italy

[2] Department of Computer Science, University of Udine, Via delle Scienze 206, 33100 Udine, Italy

[3] Department of Computer Science, Free University of Bozen-Bolzano, Piazza Università, 1, 39100 Bolzano, Italy

Published online: 11 August 2023

tion of fragments of S1S and LTL, with a limited expressive power, for which realizability can be solved efficiently.

A well-known example is *Generalized Reactivity(1)* (GR(1), for short) [7]. In this fragment, a specification is syntactically partitioned into *assumptions* about the environment and *guarantees* for the controller. Both of them are either Boolean formulas (say, α) or safety formulas ($\mathsf{G}\alpha$) or conjunctions of recurrence formulas ($\bigwedge_{i=1}^{n} \mathsf{GF}\alpha_i$). The dichotomy between *assumptions* and *guarantees* reflects the way a system engineer usually formalizes system requirements, which is summarized by the following sentence: *"the controller has to behave in conformance to the guarantees, under the given assumptions on the environment"*.

On a different direction, other approaches focused on *safety fragments* of Linear Temporal Logic with Past (LTL+P) [8, 9]. In particular, *Extended Bounded Response* LTL+P ($\mathsf{LTL}_{\mathsf{EBR}}$+P, for short) is a safety fragment of LTL+P with the following features [9]: (i) its realizability problem is EXPTIME-complete; (ii) there exists a fully symbolic compilation of its formulas into deterministic automata. This last feature, in particular, contributes to a great improvement in solving time for the synthesis problem.

A second research direction on reactive synthesis focuses on finding good algorithms for the average case. Among these, an important class of algorithms makes use of the so-called *safety reductions*, that reduce the realizability problem for the original formula to a sequence of subproblems for *safety* formulas by *bounding* some behaviors of the former, e.g., the visits to the rejecting states of the corresponding automaton, by some integer k. The rationale behind these techniques is that a safety problem is usually much simpler to solve, since it amounts to a *strong reachability* problem [10]. This is in turn inspired by safety reductions for the model checking problem, where the validity of an LTL formula over a Kripke structure is reduced to checking the reachability of a given error state [11, 12]. Usually, safety reductions (for both realizability and model checking) are: (i) *sound*, meaning that a positive answer to any of the subproblems implies a positive result of the original formula, and (ii) *complete*, ensuring that there exists an upper bound μ such that, if the k-th subproblem has a negative answer for all $k \leq \mu$, then also the result for the original formula is negative. Meaningful examples of safety reductions for realizability are the contributions on *bounded synthesis* [13], which are based on the pioneering work on *Safraless algorithms* [14], and all the different encodings proposed for solving it [15].

1.1 Contributions

The main contributions of the paper are the following ones.

First, we introduce the logic of *Generalized Reactivity(1)* $\mathsf{LTL}_{\mathsf{EBR}}$+P (GR-EBR, for short), an extention of $\mathsf{LTL}_{\mathsf{EBR}}$+P that admits:

1. fairness conditions, in particular conjunctions of *recurrence formulas*, that is, $\bigwedge_i \mathsf{GF}\alpha_i$, forcing each formula α_i to be true infinitely often;
2. assumptions/guarantees in the form of an $\mathsf{LTL}_{\mathsf{EBR}}$+P formula augmented with fairness conditions.

In addition to make it possible to express any $\mathsf{LTL}_{\mathsf{EBR}}$+P formula, GR-EBR also allows one to define properties that are not captured by the safety fragment, like, for instance, $\mathsf{G}p \rightarrow \mathsf{G}q$.

Second, we investigate the expressiveness of GR-EBR. We begin with the analysis of the expressive power of $\mathsf{LTL}_{\mathsf{EBR}}$+P, which, by definition, is a fragment of GR-EBR, and we prove that it can express all and only the safety properties that are definable in LTL+P. A crucial role in the proof of such an expressive equivalence is played by past temporal modalities of $\mathsf{LTL}_{\mathsf{EBR}}$+P. Then, we prove that they are really necessary by showing that the logic $\mathsf{LTL}_{\mathsf{EBR}}$, that is, the fragment of $\mathsf{LTL}_{\mathsf{EBR}}$+P where past modalities have been removed, is strictly less expressive than $\mathsf{LTL}_{\mathsf{EBR}}$+P. Finally, by exploiting the previous results, we demonstrate that GR-EBR is expressively equivalent to GR(1) [7].

Third, we study the reactive synthesis problem for GR-EBR. In particular, we give an algorithm that, at each iteration, builds a safety subproblem and checks its realizability. If it returns a positive result, then the initial formula is realizable as well, and a controller implementing the specification can be effectively built; otherwise, it continues with the next iteration. If the upper bound given by the reduction has been reached, the algorithm outputs the unrealizability of the initial formula. As a matter of fact, the upper bound is doubly exponential in the size of the formula and thus prohibitively large. For this reason, in practice, it is useful to use the algorithm in *parallel* with another one checking for the unrealizability of the formula. The first that terminates stops the other and, thus, the entire procedure. A crucial property of the algorithm is that the realizability check of each safety subproblem is performed in a *fully symbolic* way, thus retaining the distinctive feature of $\mathsf{LTL}_{\mathsf{EBR}}$+P.

Fourth, we prove the correctness of the proposed algorithm for realizability. To this end, we devise a novel framework for deriving *complete* safety reductions in the context of realizability of (fragments of) LTL+P. A notable feature of the framework is that it provides a link to safety reductions for model checking, by showing that if a reduction is complete for model checking, then it is complete for realizability as well. On the one hand, this allows one to reason on Kripke structures instead of strategies, which is simpler; on the other hand, it enables the use of some reductions already exploited in model checking for realizability, provided that they conform to the framework. We use the framework to derive a *complete* safety reduction for the realizability problem of GR-EBR. Moreover, we show how to apply it to prove

the completeness of Bounded Synthesis [13] using the K-Liveness reduction to safety [11].

Last but not least, we provide an implementation of the algorithm as a prototype tool called GRACE (*GR-ebr reAlizability ChEcker*). The experimental evaluation shows good performance against tools for full LTL+P synthesis.

This work considerably extends [1, 2] by:

1. adding a characterization of the exact expressive power of GR-EBR;
2. providing the complete proofs of all lemmata and theorems;
3. describing in detail the formalization of Bounded Synthesis in terms of the proposed general framework;
4. improving the analysis of related work.

1.2 Relevance for SoSyM

The relevance of the present work for a journal like the *International Journal on Software and Systems Modeling* (SoSyM) is manyfold. It proposes a novel formalism to express meaningful temporal properties, and pairs it with some effective techniques to check their *realizability*. The latter feature is fundamental in the context of correct-by-construction design; moreover, it turns out to be quite useful in the process of specification debugging: while satisfiability checks fail to establish whether a specification is *implementable*, realizability can be successfully employed to solve this task.

1.3 Organization

The rest of the paper is organized as follows. In Sect. 2, we introduce basic notation and definitions. In Sect. 3, we define the logic GR-EBR and we give an example of GR-EBR specification. In Sect. 4, we determine the expressive power of $\mathsf{LTL}_{\mathsf{EBR}}$+P, $\mathsf{LTL}_{\mathsf{EBR}}$, and GR-EBR. The symbolic algorithm that solves GR-EBR realizability is given in Sect. 5. In Sect. 6, we describe the framework for deriving complete reductions and we apply it to the case of GR-EBR, proving completeness of the algorithm described in the previous section. In Sect. 7, we analyse related work. The outcomes of the experimental evaluation are reported in Sect. 8. Finally, in Sect. 9, we point out some interesting future research directions.

2 Preliminaries

2.1 Temporal logics

Linear Temporal Logic with Past (LTL+P) is a modal logic interpreted over infinite state sequences. Let Σ be a set of proposition letters. LTL+P formulas are inductively defined as follows:

$$\phi := p \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \mathsf{X}\,\phi \mid \phi_1 \;\mathsf{U}\; \phi_2 \mid \mathsf{Y}\,\phi \mid \phi_1 \;\mathsf{S}\; \phi_2$$

where $p \in \Sigma$. Temporal operators can be partitioned into *future operators* (*next* (X) and *until* (U)) and *past operators* (*yesterday* (Y) and *since* (S)). We define the following common abbreviations, where $\top$ stands for any tautology, e.g., $p \vee \neg p$: (i) *release*: $\phi_1\,\mathsf{R}\,\phi_2 \equiv \neg(\neg\phi_1\,\mathsf{U}\,\neg\phi_2)$; (ii) *eventually*: $\mathsf{F}\phi_1 \equiv \top\,\mathsf{U}\,\phi_1$; (iii) *globally*: $\mathsf{G}\phi_1 \equiv \neg\,\mathsf{F}\,\neg\phi_1$; (iv) *once*: $\mathsf{O}\phi_1 \equiv \top\,\mathsf{S}\,\phi_1$; (v) *historically*: $\mathsf{H}\phi_1 \equiv \neg\,\mathsf{O}\,\neg\phi_1$; (vi) *weak yesterday*: $\mathsf{Z}\phi_1 \equiv \neg\,\mathsf{Y}\,\neg\phi_1$. We say that an LTL+P formula is *pure past* if (and only if) all its temporal operators are past operators. We call *pure past* LTL+P, written as $\mathsf{LTL+P}_{\mathsf{P}}$, the fragment of LTL+P containing only pure past formulas. Moreover, we denote by $\mathsf{LTL+P}_{\mathsf{BF}}$ the *Bounded Future* fragment of LTL+P, that is, the fragment of LTL+P where the only admitted future temporal modality is X.

Formulas of LTL+P are interpreted over *state sequences*. A *state sequence* $\sigma = \langle \sigma_0, \sigma_1, \ldots \rangle$ is an *infinite*, linearly ordered sequence of *states*, where each state σ_i is a set of proposition letters, that is, $\sigma_i \in 2^{\Sigma}$, for $i \in \mathbb{N}$. We will interchangeably use the term *ω-word* over the alphabet 2^{Σ} to refer to a state sequence. A set of ω-words is called *ω-language*. Given two indices $i, j \in \mathbb{N}$, with $i \leq j$, we denote by $\sigma_{[i,j]}$ the interval of σ from index i to index j, that is, $\langle \sigma_i, \ldots, \sigma_j \rangle$ if $i \geq 0$, or $\langle \sigma_0, \ldots, \sigma_j \rangle$ otherwise. Finally, we denote by $\sigma_{[i,\infty]}$ the (infinite) suffix of σ starting from i.

Given a state sequence σ, a position $i \geq 0$, and an LTL+P formula ϕ, we inductively define the *satisfaction* of ϕ by σ at position i, written as $\sigma, i \models \phi$, as follows:

1. $\sigma, i \models p$ iff $p \in \sigma_i$;
2. $\sigma, i \models \neg\phi$ iff $\sigma, i \not\models \phi$;
3. $\sigma, i \models \phi_1 \vee \phi_2$ iff $\sigma, i \models \phi_1$ or $\sigma, i \models \phi_2$;
4. $\sigma, i \models \mathsf{X}\phi$ iff $\sigma, i+1 \models \phi$;
5. $\sigma, i \models \mathsf{Y}\phi$ iff $i > 0$ and $\sigma, i-1 \models \phi$;
6. $\sigma, i \models \phi_1\,\mathsf{U}\,\phi_2$ iff there exists $j \geq i$ such that $\sigma, j \models \phi_2$, and $\sigma, k \models \phi_1$ for all k, with $i \leq k < j$;
7. $\sigma, i \models \phi_1\,\mathsf{S}\,\phi_2$ iff there exists $j \leq i$ such that $\sigma, j \models \phi_2$, and $\sigma, k \models \phi_1$ for all k, with $j < k \leq i$;

We say that σ *satisfies* ϕ, written as $\sigma \models \phi$, if it satisfies the formula at the first state, that is, if $\sigma, 0 \models \phi$: in this case, we say that σ is a *model* of ϕ. We say that two formulas ϕ and ψ are *equivalent* ($\phi \equiv \psi$) if and only if they are satisfied by the same set of state sequences.

2.2 Notation

Let ϕ be a LTL+P formula. We define the *language* of ϕ, denoted by $\mathcal{L}(\phi)$, as $\mathcal{L}(\phi) = \{\sigma \in (2^{\Sigma})^{\omega} \mid \sigma \models \phi\}$. If

ϕ contains only past operators, we change the definition of the *language* of ϕ as follows: for all $\phi \in \mathsf{LTL{+}P_P}$, the language of ϕ over *finite words* is $\mathcal{L}^{<\omega}(\phi) := \{\sigma \in (2^{\Sigma})^* \mid \sigma = \langle \sigma_0, \ldots, \sigma_n \rangle \text{ and } \sigma, n \models \phi\}$.

From now on, given a linear temporal logic $\mathbb{L}$, with some abuse of notation, we will denote by $\mathbb{L}$ also the set of formulas that *syntactically* belong to $\mathbb{L}$. Conversely, we denote by $[\![\mathbb{L}]\!]$ the set of all and only those languages $\mathcal{L}$ of infinite words for which there is a formula $\phi \in \mathbb{L}$ (i.e., ϕ syntactically belongs to $\mathbb{L}$) such that $\mathcal{L} = \mathcal{L}(\phi)$. For the logic $\mathsf{LTL{+}P_P}$, we denote by $[\![\mathsf{LTL{+}P_P}]\!]^{<\omega}$ the set of languages $\mathcal{L}$ over *finite* words such that $\mathcal{L} = \mathcal{L}^{<\omega}(\phi)$ for some $\phi \in \mathsf{LTL{+}P_P}$.

Notice that, since past modalities do *not* add expressive power to LTL [16–18], $[\![\mathsf{LTL}]\!]$ is the same as $[\![\mathsf{LTL{+}P}]\!]$.

2.3 ω-regular expressions and (co-)safety classes

Let REG be the set of *regular languages* of finite words [19]. An *ω-regular language* is a set of ω-words recognized by an ω-regular expression, that is, an expression of the form $\bigcup_{i=1}^{n} U_i \cdot (V_i)^{\omega}$, where $n \in \mathbb{N}$ and $U_i, V_i \in \mathsf{REG}$ for $i = 1, \ldots, n$. We denote the set of ω-regular languages by ω-REG. One of the seminal results in automata theory is the correspondence between ω-regular languages and Büchi automata [20, 21]. An important class of ω-regular languages is the class of those languages preventing something "bad", e.g., a deadlock or a simultaneous access to a critical section by different processes, from happening. Languages in this class are called *safety languages* or, equivalently, *safety properties*.

Definition 1 (*Safety language* [22])
Let $\mathcal{L} \subseteq \Sigma^{\omega}$ be an ω-regular language. We say that $\mathcal{L}$ is a *safety language* if and only if for all the words $\sigma \in \Sigma^{\omega}$ it holds that, if $\sigma \notin \mathcal{L}$, then $\exists i \in \mathbb{N} \cdot \forall \sigma' \in \Sigma^{\omega} \cdot \sigma_{[0,i]} \cdot \sigma' \notin \mathcal{L}$. The class of safety ω-regular languages is denoted by SAFETY.

Let $\mathbb{L}$ be a temporal logic. We say that $\mathbb{L}$ is a *safety fragment* of LTL+P if and only if $\mathbb{L} \subseteq \mathsf{LTL{+}P}$ and $\mathcal{L}(\phi)$ is a safety language (Def. 1), for all formulas $\phi \in \mathbb{L}$. A formula ϕ is called a *safety formula* if $\mathcal{L}(\phi)$ is a safety language.

Besides SAFETY, we introduce the class of (ω-regular) *co-safety* languages, called coSAFETY, which is defined as the dual of SAFETY, i.e., $\mathcal{L} \in \mathsf{coSAFETY}$ if and only if $\overline{\mathcal{L}} \in \mathsf{SAFETY}$, where $\overline{\mathcal{L}}$ is the complement language of $\mathcal{L}$.

We give an alternative, equivalent definition of the SAFETY class of Def. 1, that will be useful in the following:

$$\mathsf{SAFETY} := \{\mathcal{L} \subseteq \Sigma^{\omega} \mid \overline{\mathcal{L}} = K \cdot \Sigma^{\omega} \wedge K \in \mathsf{REG}\}$$

We define the class $\mathsf{SAFETY}^{\mathsf{SF}}$ ($\mathsf{coSAFETY}^{\mathsf{SF}}$) as the set obtained from SAFETY (resp. coSAFETY) by restricting K to be a *star-free* expression, that is, a regular expression devoid of the Kleene star [23]. In particular, $\mathsf{coSAFETY}^{\mathsf{SF}} := \{\mathcal{L} \subseteq \Sigma^{\omega} \mid \mathcal{L} = K \cdot \Sigma^{\omega} \wedge K \in \mathsf{SF}\}$, where $\mathsf{SF} \subseteq \mathsf{REG}$ is the set of star-free regular expressions. With ω-SF we denote the set of star-free ω-regular expressions. We recall that star-free expressions (SF) and pure-past LTL (LTL+P$_\mathsf{P}$) have the same expressive power. The same holds for the ω-SF class and LTL.

Proposition 1 (Thomas [24], Lichtenstein et al. [16]) $[\![\mathsf{LTL{+}P_P}]\!]^{<\omega} = \mathsf{SF}$ *and* $[\![\mathsf{LTL}]\!] = \omega\text{-}\mathsf{SF}$.

We will use the following normal-form theorem, stated in [25], that proves that any LTL-definable safety (resp., co-safety) language can be expressed by a formula of the form $\mathsf{G}\alpha$ (resp., $\mathsf{F}\alpha$), where $\alpha \in \mathsf{LTL{+}P_P}$, and *vice versa*. An independent proof of such a theorem can be derived from the results by Thomas in [24]. Hereafter, we will denote by $[\![\mathsf{G}\alpha]\!]$ (resp., $[\![\mathsf{F}\alpha]\!]$) the set of ω-languages recognized by a formula of the form $\mathsf{G}\alpha$ (resp., $\mathsf{F}\alpha$) with $\alpha \in \mathsf{LTL{+}P_P}$.

Theorem 1 (Chang *et al.* [25]) *It holds that:*

$$[\![\mathsf{LTL}]\!] \cap \mathsf{SAFETY} = [\![\mathsf{G}\alpha]\!] \textit{ and } [\![\mathsf{LTL}]\!] \cap \mathsf{coSAFETY} = [\![\mathsf{F}\alpha]\!]$$

The logic Safety-LTL [8, 25, 26] is defined as the set of LTL formulas such that, when in negated normal form, do *not* contain existential temporal operators (i.e., U and F). Safety-LTL is a *safety fragment* of LTL [26]. In [25], it is proved that Safety-LTL is expressively complete with respect to the safety fragment of LTL+P.

Theorem 2 $[\![\mathsf{Safety\text{-}LTL}]\!] = [\![\mathsf{LTL}]\!] \cap \mathsf{SAFETY}$

A summary of the expressiveness of the considered logics is given in Fig. 1. Note that the intersection of SAFETY, coSAFETY and $[\![\mathsf{LTL{+}P}]\!]$ is *not empty*, since there are languages that are definable in LTL+P and are both safety and co-safety, like, for example, the language of the formula $\mathsf{XXX}(\mathsf{Y}q \wedge \mathsf{Y}p)$.

2.4 The reactivity classes

In the following, we briefly recall the *Temporal Hierachy* defined by Manna and Pnueli in [27]. The top class in the hierarchy is the class *Reactivity(N)*, which consists of all and only the formulas of the form:

$$\bigwedge_{i=1}^{N} \Big(\mathsf{GF}(\alpha_i) \rightarrow \mathsf{GF}(\beta_i) \Big),$$

where $\alpha_i, \beta_i \in \mathsf{LTL{+}P_P}$, for $i = 1, \ldots, N$, for some $N \in \mathbb{N}$.

We denote the classes Reactivity(1), …, Reactivity(N) by R(1), …, R(N), respectively. It is known that LTL is exactly as expressive as the union of Reactivity(N), for all $N \in \mathbb{N}$.

In [7], Bloem et al. introduced a logic obtained from *Reactivity(1)* by increasing the number of subformulas of the form

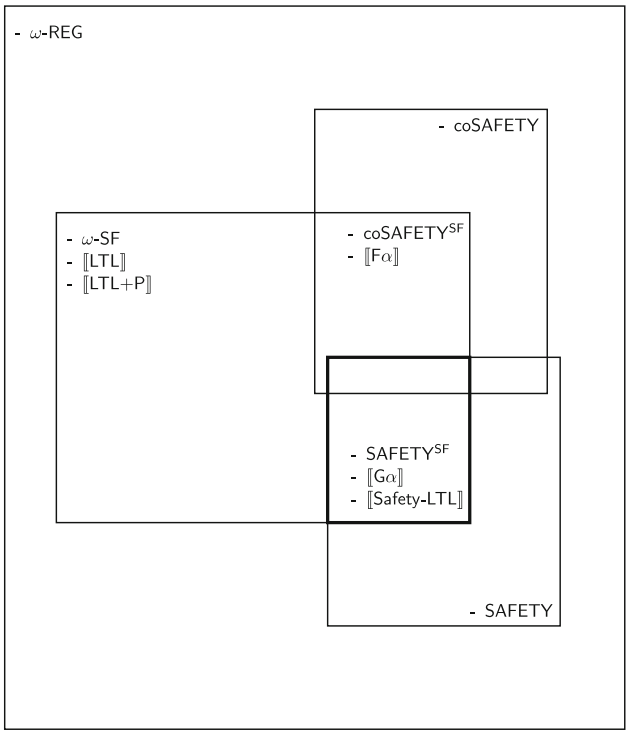


Fig. 1 A comparison of the expressiveness of the considered logics

GF that a formula can contain, maintaining the constraint of having only one implication. The resulting class, called *Generalized Reactivity(1)*, is defined as follows.

Definition 2 (GR(1) [7]) *Generalized Reactivity(1)* (GR(1) for short) is the set of all and only the formulas of the form $(\alpha_I \wedge \mathsf{G}(\alpha_T) \wedge \bigwedge_{i=1}^{m} \mathsf{GF}\alpha_i) \rightarrow (\beta_I \wedge \mathsf{G}(\beta_T) \wedge \bigwedge_{j=1}^{n} \mathsf{GF}\beta_j)$, where (i) α_I, β_I are Boolean formulas, (ii) α_T, β_T are $\mathsf{LTL{+}P}_{\mathsf{BF}}$ formulas, where modality X can only occur in a non-nested form, and (iii) α_i, β_j are $\mathsf{LTL{+}P}_{\mathsf{P}}$ formulas, for all $i \in \{1, \ldots, m\}$ and $j \in \{1, \ldots, n\}$, for some $m, n \in \mathbb{N}$.

2.5 Extended bounded response LTL+P

Extended Bounded Response LTL *with Past* ($\mathsf{LTL}_{\mathsf{EBR}}\mathsf{+P}$, for short) is a fragment of LTL+P, which has been recently introduced in the context of reactive synthesis [9]. Below, we recall its syntax.

Definition 3 (*The logic* $\mathsf{LTL}_{\mathsf{EBR}}\mathsf{+P}$ [9]) Let $a, b \in \mathbb{N}$. An $\mathsf{LTL}_{\mathsf{EBR}}\mathsf{+P}$ formula χ is inductively defined as follows:

$$\eta := p \mid \neg\eta \mid \eta_1 \vee \eta_2 \mid \mathsf{Y}\eta \mid \eta_1 \,\mathsf{S}\, \eta_2 \qquad \textit{PurePastLayer}$$
$$\psi := \eta \mid \neg\psi \mid \psi_1 \vee \psi_2 \mid \mathsf{X}\psi \mid \psi_1 \,\mathsf{U}^{[a,b]}\, \psi_2 \qquad \textit{BoundedFutureLayer}$$
$$\phi := \psi \mid \phi_1 \wedge \phi_2 \mid \mathsf{X}\phi \mid \mathsf{G}\phi \mid \psi \,\mathsf{R}\, \phi \qquad \textit{FutureLayer}$$
$$\chi := \phi \mid \chi_1 \vee \chi_2 \mid \chi_1 \wedge \chi_2 \qquad \textit{BooleanLayer}$$

where the *bounded until* operator $\psi_1 \,\mathsf{U}^{[a,b]}\, \psi_2$ is a shorthand for the LTL formula $\bigvee_{i=a}^{b} (\mathsf{X}^i(\psi_2) \wedge \bigwedge_{j=0}^{i-1} \mathsf{X}^j(\psi_1))$, with $\mathsf{X}^n\psi = \psi$ if $n = 0$, and $\mathsf{X}^n\psi = \mathsf{X}\mathsf{X}^{n-1}\psi$ otherwise (for all $n \in \mathbb{N}$ and all $\psi \in \mathsf{LTL{+}P}$). A *bounded eventually* operator $\mathsf{F}^{[a,b]}\phi_1$ can be defined as $\top \mathsf{U}^{[a,b]}\phi_1$. Similarly, we define the *bounded globally* operator $\mathsf{G}^{[a,b]}\phi_1$ as $\neg\mathsf{F}^{[a,b]}\neg\phi_1$. Since $\mathsf{LTL}_{\mathsf{EBR}}\mathsf{+P}$ features only past temporal modalities and the *tomorrow*, the *release* and the *bounded until* future temporal modalities, it is a *safety* fragment of LTL (see Theorem 3.1 in [26]).

The syntax of $\mathsf{LTL}_{\mathsf{EBR}}\mathsf{+P}$ is articulated over layers, that impose some syntactic restrictions on the formulas that can be generated by the grammar. As an example, $\mathsf{LTL}_{\mathsf{EBR}}\mathsf{+P}$ forces the first argument of any *release* operator to contain no further R or G modalities.[1]

Any formula of $\mathsf{LTL}_{\mathsf{EBR}}\mathsf{+P}$ can be rewritten into an equivalent formula in the following *normal form*.

Definition 4 (*Normal Form of* $\mathsf{LTL}_{\mathsf{EBR}}\mathsf{+P}$ [9]) The *normal form* of $\mathsf{LTL}_{\mathsf{EBR}}\mathsf{+P}$ is the set of all and only the formulas of the following type:

$$\mathsf{X}^{i_1}\alpha_{i_1} \otimes \cdots \otimes \mathsf{X}^{i_j}\alpha_{i_j} \otimes$$
$$\mathsf{X}^{i_{j+1}}\mathsf{G}\alpha_{i_{j+1}} \otimes \cdots \otimes \mathsf{X}^{i_k}\mathsf{G}\alpha_{i_k} \otimes$$
$$\mathsf{X}^{i_{k+1}}(\alpha_{i_{k+1}}\mathsf{R}\beta_{i_{k+1}}) \otimes \cdots \otimes \mathsf{X}^{i_h}(\alpha_{i_h}\mathsf{R}\beta_{i_h})$$

where $\alpha_1, \ldots, \alpha_{i_h}, \beta_{i_{k+1}}, \ldots, \beta_{i_h} \in \mathsf{LTL{+}P}_{\mathsf{P}}$, $\otimes \in \{\wedge, \vee\}$, and $i, j, k, h \in \mathbb{N}$.

We define $\mathsf{LTL}_{\mathsf{EBR}}$ as the fragment of $\mathsf{LTL}_{\mathsf{EBR}}\mathsf{+P}$ devoid of the pure past layer.

Definition 5 (*The logic* $\mathsf{LTL}_{\mathsf{EBR}}$) The logic $\mathsf{LTL}_{\mathsf{EBR}}$ is obtained from $\mathsf{LTL}_{\mathsf{EBR}}\mathsf{+P}$ by removing past temporal operators.

2.6 Automata

The relationships between temporal logic and automata on infinite words have been extensively and successfully investigated in the literature (see, for instance, [28]). In the following, we will focus our attention on symbolic representations, which can be exponentially more succinct than the explicit-state one. Thus, we restrict our attention to symbolic automata, which are defined as follows.

Definition 6 (*Symbolic automaton on infinite words*) A *symbolic automaton on infinite words* over the alphabet Σ is a tuple $\mathcal{A} = (V, I, T, \alpha)$, where (i) $V = X \cup \Sigma$, with X a set of *state variables* and Σ a set of *input variables*, (ii) $I(X)$ and

[1] As a matter of fact, the layered structure of $\mathsf{LTL}_{\mathsf{EBR}}\mathsf{+P}$ formulas was inspired by the steps of the algorithm for the construction of symbolic automata starting from $\mathsf{LTL}_{\mathsf{EBR}}\mathsf{+P}$-formulas. We refer the reader to [9] for details.

$T(X, \Sigma, X')$, with $X' = \{x' \mid x \in X\}$, are Boolean formulas which define the set of initial states and the transition relation, respectively, and (iii) $\alpha(X)$ is an LTL+P formula over the variables in X which defines the accepting condition.

The variables x' in X' are called the *primed version* of the variables x in X, and are supposed to represent the value of x in the next state.

We will make use of *deterministic* symbolic automata, which are defined as follows.

Definition 7 (*Deterministic symbolic automaton on infinite words*) A symbolic automaton $\mathcal{A} = (V, I, T, \alpha)$, with $V = X \cup \Sigma$, is *deterministic* iff (i) the formula I has exactly one satisfying assignment, and (ii) the transition relation is of the form:

$$T(X, \Sigma, X') := \bigwedge_{x \in X} (x' \leftrightarrow \beta_x(V))$$

where $\beta_x(V)$ is a Boolean formula over V, for each $x \in X$.

Symbolic automata over infinite words recognize a set of infinite words, called the *language* of the automaton, which is defined as follows.

Definition 8 (*Language of a symbolic automaton*) Let $\mathcal{A} = (V, I, T, \alpha)$ be a symbolic automaton. A run $\tau = \langle \tau_0, \tau_1, \dots \rangle$ is an infinite sequence of *states* (evaluations of the variables in X) such that any two consecutive states (evaluations) satisfy the formula T, for some assignment to the variables in Σ. A run τ is induced by the word σ iff $\tau_0 \models I$ and $(\tau_i, \sigma_i, \tau_{i+1}) \models T$, for all $i \geq 0$. A run τ is *accepting* iff $\tau \models \alpha$. A word σ is *accepted* by $\mathcal{A}$ iff there exists an accepting run induced by σ in $\mathcal{A}$. The *language* of $\mathcal{A}$, denoted by $\mathcal{L}(\mathcal{A})$, is the set of all and only the infinite words accepted by $\mathcal{A}$.

We define the following classes of symbolic automata over infinite words, which differ from each other in the acceptance condition.

Definition 9 (*Safety Automata,* GR(1) *Automata, and* GR(1) *Automata*) Let $\mathcal{A} = (V, I, T, \alpha)$ be a symbolic automaton over infinite words. We say that $\mathcal{A}$ is

- a *safety automaton* iff $\alpha := \mathsf{G}\beta$;
- a GR(1) *automaton* iff $\alpha := \mathsf{GF}\beta \rightarrow \mathsf{GF}\beta'$;
- a GR(1) *automaton* $\alpha := \bigwedge_{i=1}^{m} \mathsf{GF}\beta_i \rightarrow \bigwedge_{j=1}^{n} \mathsf{GF}\beta'_j$.

where $\beta, \beta', \beta_i, \beta'_j \in \text{LTL+P}_\text{P}$ and $m, n \in \mathbb{N}$.

2.7 Model checking, realizability, and synthesis

We conclude the section by formally stating the problems of model checking, realizability, and synthesis. As a preliminary step, we recall the notion of (finite) Kripke structure.

Definition 10 (*Kripke structure*) A Kripke structure is a tuple $M = (\Sigma, Q, I, T, L)$, where

1. Σ is the input alphabet;
2. Q is the (finite) set of states;
3. $I \subseteq Q$ is the set of initial states;
4. $T \subseteq Q \times Q$ is a *complete* transition relation;
5. $L : Q \rightarrow 2^\Sigma$ is the labeling function that assigns to each state the set of proposition letters in Σ that are true in it.

We denote the number of states in M, i.e., $|Q|$, by $|M|$. A infinite *trace* of M is a sequence of states such that any consecutive pair of states (q_i, q_{i+1}) in it belongs to T. Given a trace $\pi := \langle q_0, q_1, \dots \rangle$ in M, we denote by $L(\pi)$ the sequence $\langle L(q_0), L(q_1), \dots \rangle$. A path π is called *initialized* if and only if $q_0 \in I$.

The *model checking* problem takes as input a Kripke structure and a temporal formula, and verifies whether or not all the initialized traces of the former satisfy the latter.

Definition 11 (*Model checking for* LTL+P) Let M be a Kripke structure and ϕ be an LTL+P temporal formula. The model checking problem is the problem of verifying whether or not for *all* initialized traces π of M, it holds that $L(\pi) \models \phi$, written $M \models \mathsf{A}\phi$, where A is the *"for all paths"* modality of CTL.

Realizability and *reactive synthesis* are, in some sense, more ambitious than model checking, as they aim at establishing whether a given temporal formula ϕ over two sets $\mathcal{U}$ and $\mathcal{C}$ of uncontrollable and controllable variables, respectively, is implementable and, if this is the case, to synthesize a possible *implementation*. Realizability is usually modeled as a two-player game between Environment, who tries to violate the specification, and Controller, who tries to fulfill it. In this setting, an implementation of the specification is represented by a *strategy*.

Definition 12 (*Strategies and languages of strategies*) Let $\mathcal{U}$ and $\mathcal{C}$ be two disjoint sets of *input* (or *uncontrollable*) and *output* (or *controllable*) variables, respectively. A *strategy* g is a function $g : (2^\mathcal{U})^+ \rightarrow 2^\mathcal{C}$ (where $(2^\mathcal{U})^+$ is the set of finite, non-empty sequences of elements in $2^\mathcal{U}$). The *language* of the strategy g, denoted by $\mathcal{L}(g)$, is the set of all and only the sequences $\langle (\mathsf{U}_0 \cup \mathsf{C}_0), (\mathsf{U}_1 \cup \mathsf{C}_1), \dots \rangle$ such that $\mathsf{U}_i \in 2^\mathcal{U}$ and $\mathsf{C}_i = g(\langle \mathsf{U}_0, \dots, \mathsf{U}_i \rangle)$, for all $i \geq 0$.

Definition 13 (*Realizability and synthesis for* LTL+P) Let ϕ be an LTL+P temporal formula over $\Sigma = \mathcal{U} \cup \mathcal{C}$, where $\mathcal{U}$ is the set of input variables, $\mathcal{C}$ the set of output variables, and $\mathcal{U} \cap \mathcal{C} = \emptyset$. We say that ϕ is *realizable* if and only if there exists a strategy $g : (2^\mathcal{U})^+ \rightarrow 2^\mathcal{C}$ such that $\mathcal{L}(g) \subseteq \mathcal{L}(\phi)$. If ϕ is realizable, the synthesis problem is the problem of computing such a strategy.

The strategies we are mainly interested in are those that can be represented finitely. In the literature, there are two main (and equivalent) representations of finite strategies, namely, *Mealy machines* and *Moore machines*. In this paper, we focus on the first ones.

Definition 14 (*Mealy machine*) A *Mealy machine* is a tuple $M = (\Sigma_{\mathcal{U}}, \Sigma_{\mathcal{C}}, Q, q_0, \delta)$, where

1. $\Sigma_{\mathcal{U}}$ and $\Sigma_{\mathcal{C}}$ are the input and output alphabets, respectively;
2. Q is the (finite) set of states and q_0 is the initial state;
3. $\delta : Q \times \Sigma_{\mathcal{U}} \rightarrow \Sigma_{\mathcal{C}} \times Q$ is the *total* transition *function*.

We say that an infinite word $\sigma = \langle \sigma_0, \sigma_1, \dots \rangle \in (\Sigma_{\mathcal{U}} \cup \Sigma_{\mathcal{C}})^{\omega}$ is *accepted* by M iff there exists a *trace* $\langle (q_0, \sigma_0), (q_1, \sigma_1), \dots \rangle \in (Q \times (\Sigma_{\mathcal{U}} \cup \Sigma_{\mathcal{C}}))^{\omega}$ such that $\delta(q_i, \sigma_i \cap \Sigma_{\mathcal{U}}) = (\sigma_i \cap \Sigma_{\mathcal{C}}, q_{i+1})$, for all $i \geq 0$. The language of M, denoted by $\mathcal{L}(M)$, is the set of all the infinite words accepted by M.

A fundamental feature of the realizability problem for LTL+P is the following *small model property* [5, 14, 29], which ensures that each realizable LTL+P formula has at least one finitely representable strategy.

Proposition 2 (Small model property of LTL+P [5]) *Let ϕ be an* LTL+P *formula and $n = |\phi|$. If ϕ is realizable by a strategy g, then there exists a Mealy machine M_g such that (i) M_g has at most $2^{2^{c \cdot n}}$ states, for some constant $c \in \mathbb{N}$, and (ii) $\mathcal{L}(M_g) \subseteq \mathcal{L}(\phi)$.*

Notice that the constant c in Prop. 2 depends on the algorithms used for building the *nondeterministic Büchi automaton* for the language $\mathcal{L}(\phi)$ and on the algorithms for its determinization (e.g., *Safra's algorithm* [30] or *alternating cycle decomposition* [31]) to obtain an equivalent deterministic Rabin automaton (see [5]). It immediately follows that c can be effectively computed, obtaining a concrete upper bound for the size of the Mealy machine in Prop. 2.

In general, the realizability problem for an LTL+P formula can be reduced to a Büchi game [32], that requires the player Controller to visit infinitely often a state (or a set of states) in the arena. In the case of safety specifications, the full power (and complexity) of Büchi games is not necessary. *Safety games* [33, 34] are indeed a restriction of Büchi games, where the goal of Controller is to visit only states in a given set, called *safe states*.

Definition 15 (*Safety game*) Let $\mathcal{A} = (V, I, T, \alpha)$ be a symbolic deterministic *safety* automaton (see Def. 9) such that $V = X \cup \Sigma$, and $\Sigma = \mathcal{C} \cup \mathcal{U}$, with $\mathcal{C} \cap \mathcal{U} = \emptyset$, and $\alpha := \mathsf{G}(\beta)$, with $\beta \in \text{LTL+P}_{\mathsf{P}}$. A safety game is a triple $G = \langle \mathcal{A}, \mathcal{C}, \mathcal{U} \rangle$. We say that Controller wins G iff there exists a strategy $g : (2^{\mathcal{U}})^{+} \rightarrow 2^{\mathcal{C}}$ such that, for all sequences $\mathsf{U} = \langle \mathsf{U}_0, \mathsf{U}_1, \dots \rangle \in (2^{\mathsf{U}})^{\omega}$, *the* run τ induced by $\mathsf{U} = \langle \mathsf{U}_0 \cup g(\mathsf{U}_0), \mathsf{U}_1 \cup g(\mathsf{U}_0, \mathsf{U}_1), \dots \rangle$ in $\mathcal{A}$ is *accepting*, that is, it only visits states τ (i.e., evaluations of the variables in X) such that $\tau \models \beta$.

In the following, we will reduce the realizability problem for GR-EBR to a sequence of *safety games*.

3 The logic of GR-EBR

In this section, we introduce the logic *Generalized Reactivity(1)* LTL_{EBR}+P (GR-EBR, for short), an extension of LTL_{EBR}+P with fairness conditions of the form $\mathsf{GF}\alpha$, with $\alpha \in \text{LTL+P}_{\mathsf{P}}$, and assumptions/guarantees, in the form of a logical implication.

Formulas of the logic GR-EBR are defined as follows.

Definition 16 (*The logic* GR-EBR) The logic GR-EBR consists of all and only the formulas of the following form:

$$\left(\psi^1_{ebr} \wedge \bigwedge_{i=1}^{m} \mathsf{G}\,\mathsf{F}\alpha_i\right) \rightarrow \left(\psi^2_{ebr} \wedge \bigwedge_{j=1}^{n} \mathsf{G}\,\mathsf{F}\beta_j\right)$$

where $m, n \in \mathbb{N}$, $\psi^1_{ebr}, \psi^2_{ebr} \in \text{LTL}_{\text{EBR}}$+P and $\alpha_i, \beta_j \in \text{LTL+P}_{\mathsf{P}}$, for all $i, j \in \mathbb{N}$.

In order to show the expressive power and naturalness of GR-EBR, we give an example of a meaningful specification that can be formulated in it. More precisely, we take an example that was originally proposed in [9], and we extend it with fairness conditions, assumptions, and guarantees.

Let us consider an arbiter that, given a request from a client i, with $i \in \{1, \dots, n\}$, assigns it a grant in such a way to guarantee the following properties: (1) the grant is assigned at most k time units after the request is issued, for some $k > n$ (*bounded response*); (2) the arbiter can assign a grant to at most one client at a time (*mutual exclusion*). The (conjunction of the) two requirements are the guarantees for the controller. The assumptions for the environment are the following: (1) initially, there are no requests; (2) if a client issues a request at time i, then it cannot issue a new request until time $i + k + 1$; (3) each client issues infinitely many requests.

Let us now show how to specify the expected behavior of the arbiter by means of a GR-EBR formula. First of all, we model the requests coming from the n clients with n (uncontrollable) variables $r_1, \dots, r_n$. Similarly, the grant for the request r_i can be modeled with a (controllable) variable g_i, for each $i \in \{1, \dots, n\}$. The assumption for the environment are expressed by the formula ϕ_e, which is defined as follows:

$$\bigwedge_{i=1}^{n} \neg r_i \;\wedge\; \bigwedge_{i=1}^{n} \mathsf{G}(r_i \rightarrow \mathsf{G}^{[1,k]} \neg r_i) \;\wedge\; \bigwedge_{i=1}^{n} \mathsf{G}\,\mathsf{F} r_i$$

The guarantees for the controller are captured by the formula ϕ_c, which is defined as follows:

$$\bigwedge_{i=1}^{n} \mathsf{G}(r_i \rightarrow \mathsf{F}^{[0,k]} g_i) \ \wedge \ \mathsf{G}\left(\bigwedge_{1 \leq i < j \leq n} \neg(g_i \wedge g_j) \right)$$

The overall specification is the GR-EBR formula $\phi_e \rightarrow \phi_c$. The corresponding GR(1) specification is obtained by substituting the subformulas $r_i \rightarrow \mathsf{G}^{[1,k]} \neg r_i$ and $r_i \rightarrow \mathsf{F}^{[0,k]} g_i$ with the equivalent pure past one $\mathsf{Y}^{k+1} r_i \rightarrow \mathsf{H}^{[1,k]} \neg r_i$ and $\mathsf{Y}^{k+1} r_i \rightarrow \mathsf{O}^{[1,k]} g_i$, respectively, where Y^k, $\mathsf{H}^{[0,k]}$, and $\mathsf{O}^{[0,k]}$ are the past operators specular to X^k, $\mathsf{G}^{[0,k]}$, and $\mathsf{F}^{[0,k]}$, respectively.

4 Expressiveness of GR-EBR

In this section, we study the expressiveness of GR-EBR. As a preliminary step, we work out the case of $\mathsf{LTL}_{\mathsf{EBR}}$+P, on which GR-EBR is based.

We first prove that $\mathsf{LTL}_{\mathsf{EBR}}$+P is expressively complete with respect to the safety fragment of LTL+P. Moreover, we show that *past temporal operators*, which play a crucial role in the proof of the expressive completeness of $\mathsf{LTL}_{\mathsf{EBR}}$+P, are really necessary: we prove that $\mathsf{LTL}_{\mathsf{EBR}}$, that is, $\mathsf{LTL}_{\mathsf{EBR}}$+P devoid of past temporal operator, is *strictly* less expressive than $\mathsf{LTL}_{\mathsf{EBR}}$+P.

Then, we investigate the expressiveness of GR-EBR and we prove that:

1. it is strictly more expressive than $\mathsf{LTL}_{\mathsf{EBR}}$+P, and thus it can express all safety properties definable in LTL+P and beyond;
2. it is expressively equivalent to GR(1).

All the proofs which are not reported in the main body of the paper can be found in the appendix (Sect. A).

4.1 Expressiveness of $\mathsf{LTL}_{\mathsf{EBR}}$+P

In this part, we study the expressiveness of the $\mathsf{LTL}_{\mathsf{EBR}}$+P logic. In particular, we compare the set of languages definable in $\mathsf{LTL}_{\mathsf{EBR}}$+P with the set of safety languages expressible in LTL+P, and prove that the two sets are equal, that is $[\![\mathsf{LTL}_{\mathsf{EBR}}\text{+P}]\!] = [\![\mathsf{LTL}]\!] \cap \mathsf{SAFETY}$. Consequently (see Theorem 2), $\mathsf{LTL}_{\mathsf{EBR}}$+P and Safety-LTL are expressively equivalent (i.e., $[\![\mathsf{LTL}_{\mathsf{EBR}}\text{+P}]\!] = [\![\text{Safety-LTL}]\!]$).

First we recall the normal-form theorem stated in Theorem 1, establishing that $[\![\mathsf{LTL}]\!] \cap \mathsf{SAFETY} = [\![\mathsf{G}\alpha]\!]$. Proving that $[\![\mathsf{LTL}_{\mathsf{EBR}}\text{+P}]\!] = [\![\mathsf{LTL}]\!] \cap \mathsf{SAFETY}$ is straightforward. In [26], Sistla proved that any fragment of LTL+P with only X, R, and G as future temporal modalities defines only safety properties, and thus is a safety fragment of LTL+P. Since $\mathsf{LTL}_{\mathsf{EBR}}$+P-formulas contain only universal (future) temporal operators, it follows that $\mathsf{LTL}_{\mathsf{EBR}}$+P is a safety fragment of LTL+P (this corresponds to the left-to-right direction). For the right-to-left direction it suffices to show that the normal form $\mathsf{G}\alpha$ is syntactically definable in $\mathsf{LTL}_{\mathsf{EBR}}$+P (i.e., $\mathsf{G}\alpha \in \mathsf{LTL}_{\mathsf{EBR}}\text{+P}$ and thus also $\mathcal{L}(\mathsf{G}\alpha) \in [\![\mathsf{LTL}_{\mathsf{EBR}}\text{+P}]\!]$, for any $\alpha \in \mathsf{LTL+P_P}$).

Theorem 8 $[\![\mathsf{LTL}_{\mathsf{EBR}}\text{+P}]\!] = [\![\mathsf{LTL}]\!] \cap \mathsf{SAFETY}$.

4.2 Comparison with Gα

Although we proved the expressive equivalence between $\mathsf{LTL}_{\mathsf{EBR}}$+P and $\mathsf{G}\alpha$, we note that $\mathsf{LTL}_{\mathsf{EBR}}$+P offers a more natural language for safety properties than the $\mathsf{G}\alpha$ fragment. Consider for example the following property, expressed in natural language: either p_3 holds forever, or there exists two time points $t' \leq t$ such that (i) p_1 holds in t, (ii) p_2 holds in t', and (iii) p_2 holds from time point 0 to t. The property can be easily formalized in $\mathsf{LTL}_{\mathsf{EBR}}$+P by the formula $p_1 \mathsf{R}(p_2 \mathsf{R} p_3)$. The equivalent formula in the $\mathsf{G}\alpha$ fragment is $\mathsf{G}(\mathsf{H}(p_3) \vee \mathsf{O}(p_2 \wedge \mathsf{O}(p_1) \wedge \mathsf{H}(p_3)))$, which is arguably more intricate.

4.3 Comparison with Safety-LTL

Safety-LTL is the fragment of LTL (thus with only future temporal modalities) containing all and only the LTL-formulas that, when in negated normal form, do *not* contain any *until* or *eventually* operator. From Theorems 2 and 8, it immediately follows that $\mathsf{LTL}_{\mathsf{EBR}}$+P and Safety-LTL are expressively equivalent, namely $[\![\mathsf{LTL}_{\mathsf{EBR}}\text{+P}]\!] = [\![\text{Safety-LTL}]\!]$.

Unlike $\mathsf{LTL}_{\mathsf{EBR}}$+P, Safety-LTL does *not* impose any syntactic restriction on the nesting of the logical operators; as a matter of fact, $\mathsf{G}(p_1 \vee \mathsf{G}p_2)$ belongs to the syntax of Safety-LTL but not to the syntax of $\mathsf{LTL}_{\mathsf{EBR}}$+P, even though $\mathsf{G}(p_1 \vee \mathsf{G}p_2) \equiv \mathsf{G}(\neg p_2 \rightarrow \mathsf{H}p_1) \in \mathsf{LTL}_{\mathsf{EBR}}\text{+P}$. The restrictions on the syntax of $\mathsf{LTL}_{\mathsf{EBR}}$+P are due to algorithmic aspects: each layer of the syntax of $\mathsf{LTL}_{\mathsf{EBR}}$+P (recall Def. 3) corresponds to a step of the algorithm for the construction of deterministic and symbolic automata starting from $\mathsf{LTL}_{\mathsf{EBR}}$+P-formulas [9]. As a matter of fact, in practice, $\mathsf{LTL}_{\mathsf{EBR}}$+P has shown to avoid an exponential blowup in time with respect to known algorithms for automata construction for safety specifications [9]. Last but not least, the realizability problem of $\mathsf{LTL}_{\mathsf{EBR}}$+P is EXPTIME-complete [9], as opposed to the realizability of LTL+P, which is 2EXPTIME-complete [5, 6]. Consider now $\mathsf{LTL}_{\mathsf{EBR}}$, which is the fragment of $\mathsf{LTL}_{\mathsf{EBR}}$+P devoid of past operators (Def. 5). Since each formula of $\mathsf{LTL}_{\mathsf{EBR}}$ syntactically belongs to Safety-LTL, it immediately follows that $[\![\mathsf{LTL}_{\mathsf{EBR}}]\!] \subseteq [\![\text{Safety-LTL}]\!]$. In Sect. 4.4, we will prove that the converse direction does

not hold, that is $\mathsf{LTL_{EBR}}$ is *strictly* less expressive than $\mathsf{LTL_{EBR}}$+P and Safety-LTL .

4.4 Expressiveness of $\mathsf{LTL_{EBR}}$

In Sect. 4.1, we have proved that the following equivalences hold:

$$[\![\mathsf{LTL_{EBR}}\text{+P}]\!] = [\![\mathsf{G}\alpha]\!] = [\![\mathsf{LTL}]\!] \cap \mathsf{SAFETY} = [\![\text{Safety-LTL}]\!]$$

In particular, thanks to the use of the *pure past layer* (recall Def. 5), $\mathsf{LTL_{EBR}}$+P can easily capture the whole class of $[\![\mathsf{G}\alpha]\!]$, and thus the whole class of $[\![\mathsf{LTL}]\!] \cap \mathsf{SAFETY}$. However, one may wonder whether the pure past layer is really necessary, or whether the class $[\![\mathsf{G}\alpha]\!]$ can be expressed in $\mathsf{LTL_{EBR}}$ without the use of past operators.

Recall from Def. 5 that $\mathsf{LTL_{EBR}}$ is defined as the fragment of $\mathsf{LTL_{EBR}}$+P devoid of the pure past layer. In this part, we prove that the removal of past operators from $\mathsf{LTL_{EBR}}$+P results into a loss of expressive power, namely:

$$[\![\mathsf{LTL_{EBR}}]\!] \subsetneq [\![\mathsf{LTL_{EBR}}\text{+P}]\!] \tag{1}$$

This result proves that *past modalities*, although being not important for the expressiveness of full LTL (since $[\![\mathsf{LTL}]\!] = [\![\mathsf{LTL}\text{+P}]\!]$ [16–18]), can play a crucial role for the expressive power of *fragments* of LTL , like, for instance, $\mathsf{LTL_{EBR}}$.

4.4.1 The general idea

We will prove Eq. (1) by showing that $[\![\mathsf{LTL_{EBR}}]\!] \subsetneq [\![\text{Safety-LTL}]\!]$. The result in Eq. (1) follows from the fact that $[\![\text{Safety-LTL}]\!] = [\![\mathsf{LTL_{EBR}}\text{+P}]\!]$. We shall prove that the language of the Safety-LTL -formula $\varphi_{\mathsf{G}} := \mathsf{G}(p_1 \vee \mathsf{G}(p_2))$ cannot be expressed by any $\mathsf{LTL_{EBR}}$ -formula. The formula φ_{G} belongs *syntactically* to Safety-LTL , and thus $\mathcal{L}(\varphi_{\mathsf{G}}) \in [\![\text{Safety-LTL}]\!]$. We also note that φ_{G} can be expressed in $\mathsf{LTL_{EBR}}$+P . Indeed, it holds that:

$$\mathsf{G}(p_1 \vee \mathsf{G}(p_2)) \equiv \mathsf{G}(\neg p_2 \rightarrow \mathsf{H}(p_1)) \tag{2}$$

Since $\mathsf{G}(\neg p_2 \rightarrow \mathsf{H}(p_1)) \in \mathsf{LTL_{EBR}}\text{+P}$, it holds that $\mathcal{L}(\varphi_{\mathsf{G}}) \in [\![\mathsf{LTL_{EBR}}\text{+P}]\!]$. It is worth noticing the following points: (i) $\mathsf{G}(\neg p_2 \rightarrow \mathsf{H}(p_1))$ is of the form $\mathsf{G}\alpha$, where $\alpha \in \mathsf{LTL+P_P}$ (α is a pure past formula); (ii) the formula φ_{G} is equivalent to $\mathsf{G}(p_2) \vee ((\mathsf{X}\,\mathsf{G}p_2)\mathsf{R}p_1)$, but the latter formula does not syntactically belong to $\mathsf{LTL_{EBR}}$ nor to $\mathsf{LTL_{EBR}}$+P , due to the restriction that forces the leftmost argument of any *release* operator to contain no universal temporal operators (i.e., R and G).

The proof of the undefinability of φ_{G} is based on the fact that all formulas of $\mathsf{LTL_{EBR}}$ can constrain, for any time point i in an infinite state sequence, only a *bounded* prefix before (or interval around) i.

Consider again the formula $\varphi_{\mathsf{G}} := \mathsf{G}(p_1 \vee \mathsf{G}(p_2))$. The language $\mathcal{L}(\varphi_{\mathsf{G}})$ is expressed by the ω-regular expression $(\{p_1\})^\omega + (\{p_1\})^* \cdot (\{p_2\})^\omega$. Written in natural language, each model of φ_{G} cannot contain a position in which $\neg p_2$ holds preceded by a position in which $\neg p_1$ holds.

Remark 1 Let $\sigma \subseteq (2^\Sigma)^\omega$ be a state sequence. It holds that:

$$\sigma \models \varphi_{\mathsf{G}} \;\Rightarrow \neg\exists i, j (j \leq i \wedge \sigma_j \models \neg p_1 \wedge \sigma_i \models \neg p_2)$$

Let ${}^{i,k}\sigma^j$ be *the* state sequence such that at time points i and k it holds $p_1 \wedge \neg p_2$, at time point j it holds $\neg p_1 \wedge p_2$, and for all the other time points $p_1 \wedge p_2$ holds (a formal definition of ${}^{i,k}\sigma^j$ will be given later). The membership of ${}^{i,k}\sigma^j$ to $\mathcal{L}(\varphi_{\mathsf{G}})$ depends on the value of the three indices i, j and k, as follows.

Remark 2 If $i < j$ and $k < j$, then ${}^{i,k}\sigma^j \models \varphi_{\mathsf{G}}$. Conversely, if $i \geq j$ or $k \geq j$, then ${}^{i,k}\sigma^j \not\models \varphi_{\mathsf{G}}$.

As we will see, given a generic formula $\psi \in \mathsf{LTL_{EBR}}$, one can always find some values for the indices i, j and k such that (a) j is chosen sufficiently greater than i; (b) k is chosen sufficiently greater than j; (c) ψ is not able to distinguish the state sequence ${}^{i,i}\sigma^j$ from ${}^{i,k}\sigma^j$. Since, by Remark 2, ${}^{i,i}\sigma^j \in \mathcal{L}(\varphi_{\mathsf{G}})$ but ${}^{i,k}\sigma^j \notin \mathcal{L}(\varphi_{\mathsf{G}})$, this proves the undefinability of φ_{G} in $\mathsf{LTL_{EBR}}$. The rationale is that the $\mathsf{LTL_{EBR}}$ logic combines bounded future formulas (i.e., formulas obtained by a Boolean combination of propositional atoms and X operators) and universal temporal operators (i.e., G and R). This implies the fact that, for a generic model σ of an $\mathsf{LTL_{EBR}}$ -formula ψ, *at each time point* $i \geq 0$ of σ (this corresponds to the universal temporal operators) only a *finite and bounded suffix* after i (this corresponds to the $\mathsf{LTL+P_{BF}}$ -formulas) can be constrained by ψ (this can be thought of as a sort of bounded memory property of this logic). Equivalently, this means that each $\mathsf{LTL_{EBR}}$ -formula is *not* able to constrain any finite but arbitrarly long (unbounded) prefix of a state sequence, contrary, for instance, to the case of the formula $\mathsf{G}(\neg p_2 \rightarrow \mathsf{H}(p_1))$ (that is equivalent to φ_{G}, see Eq. (2)).

4.4.2 The normal form

The limitation of $\mathsf{LTL_{EBR}}$ -formulas mentioned before is more evident in the *normal form* for the $\mathsf{LTL_{EBR}}$ logic, that we will define in this part. We first give some preliminary definitions. We define *Bounded Past* $\mathsf{LTL+P_P}$ ($\mathsf{LTL+P_{BP}}$, for short) as the set of all and only the $\mathsf{LTL_{EBR}}$+P formulas that are a Boolean combination of propositional atoms and *yesterday* operators (Y). We use the shortcut $\mathsf{Y}^n\psi$ for denoting ψ is $n = 0$, and $\mathsf{Y}\,\mathsf{Y}^{n-1}\psi$ otherwise. We use also the shortcut $\psi_1 \mathsf{S}^{[a,b]} \psi_2$ for denoting the formula $\bigvee_{i=a}^{b} (\mathsf{Y}^i(\psi_2) \wedge \bigwedge_{j=0}^{i-1} \mathsf{Y}^j(\psi_1))$. Given a

formula $\alpha \in \mathsf{LTL{+}P_{BP}}$, we define its *temporal depth*, denoted as $D(\alpha)$, as follows:

- $D(p) = 0$, for all $p \in \Sigma$
- $D(\neg\alpha_1) = D(\alpha_1)$
- $D(\alpha_1 \wedge \alpha_2) = \max\{D(\alpha_1), D(\alpha_2)\}$
- $D(\mathsf{Y}\alpha_1) = 1 + D(\alpha_1)$
- $D(\alpha_1 \mathsf{S}^{[a,b]} \alpha_2) = b + \max\{D(\alpha_1), D(\alpha_2)\}$

For each $\alpha \in \mathsf{LTL{+}P_{BP}}$, the language $\mathcal{L}^{<\omega}(\alpha)$ consists only of words of length at most $D(\alpha)+1$. Recall from Sect. 2 that, given an infinite state sequence $\sigma = \langle \sigma_0, \sigma_1, \ldots \rangle$ and some $n \geq 0$, $\sigma_{[n-d,n]}$ is the interval of σ of length *at most* d ending at index n. The crucial property of $\mathsf{LTL{+}P_{BP}}$-formulas, that can be shown with a simple induction, is that their truth over a state sequence σ can be checked by considering only a finite and *bounded* interval of σ, whose length depends on the *temporal depth* of the formula. This is summarized by the following remark.

Remark 3 For any $\alpha \in \mathsf{LTL{+}P_{BP}}$, with temporal depth $d = D(\alpha)$, and for any $n \geq 0$, it holds that $\sigma, n \models \alpha$ *if and only if* $\sigma_{[n-d,n]} \models \alpha$.

We give now the *normal form* for $\mathsf{LTL_{EBR}}$, and we refer to it as $\mathsf{Normal\text{-}LTL_{EBR}}$. The normal form of $\mathsf{LTL_{EBR}}$ forces any universal unbounded operator, like *globally* or *release*, to contain only $\mathsf{LTL{+}P_{BP}}$-formulas. Formally, we define $\mathsf{Normal\text{-}LTL_{EBR}}$ as the normal form described in Def. 4 but such that each α_i, β_i is a *bounded past* LTL formula. For sake of clarity, we write here below the full definition.

Definition 17 (*Normal Form of* $\mathsf{LTL_{EBR}}$) The *normal form* of $\mathsf{LTL_{EBR}}$ is the set of all and only the formulas of the following type:

$$\mathsf{X}^{i_1}\alpha_{i_1} \otimes \cdots \otimes \mathsf{X}^{i_j}\alpha_{i_j} \otimes$$
$$\mathsf{X}^{i_{j+1}}\mathsf{G}\alpha_{i_{j+1}} \otimes \cdots \otimes \mathsf{X}^{i_k}\mathsf{G}\alpha_{i_k} \otimes$$
$$\mathsf{X}^{i_{k+1}}(\alpha_{i_{k+1}}\mathsf{R}\beta_{i_{k+1}}) \otimes \cdots \otimes \mathsf{X}^{i_h}(\alpha_{i_h}\mathsf{R}\beta_{i_h})$$

where each $\alpha_i, \beta_i \in \mathsf{LTL{+}P_{BP}}$, $\otimes \in \{\wedge, \vee\}$, and $i, j, k, h \in \mathbb{N}$.

By applying the same transformation from $\mathsf{LTL_{EBR}{+}P}$ to its normal form given in [9], one obtain the following lemma.

Lemma 4 $[\![\mathsf{LTL_{EBR}}]\!] = [\![\mathsf{Normal\text{-}LTL_{EBR}}]\!]$.

The normal form of $\mathsf{LTL_{EBR}}$ makes it easier to prove Eq. (1). Take, e.g., the formula $\mathsf{XXG}(p \vee \mathsf{Y}p \vee \mathsf{YY}p)$, that belongs to $\mathsf{Normal\text{-}LTL_{EBR}}$. It is clear that, at each time point, this formula can constrain only the interval consisting of the current state and its two previous states (in fact its temporal depth is 3).

4.4.3 The main proof

We now show that the formula φ_{G} is not definable in the logic $\mathsf{Normal\text{-}LTL_{EBR}}$. Undefinability in $\mathsf{LTL_{EBR}}$ follows from Lemma 4.

Given three indices $i, j, k \in \mathbb{N}$ such that $i \neq j$ and $k \neq j$, we formally define the state sequence ${}^{i,k}\sigma^{j} = \langle {}^{i,k}\sigma_0^{j}, {}^{i,k}\sigma_1^{j}, \ldots \rangle$ as follows:

$${}^{i,k}\sigma_h^{j} = \begin{cases} \{p_1\} & \text{if } h \in \{i, k\} \\ \{p_2\} & \text{if } h = j \\ \{p_1, p_2\} & \text{otherwise} \end{cases}$$

The core of the main theorem is based on the fact that any formula of type $\mathsf{G}\alpha$ or $\alpha\mathsf{R}\beta$, where α and β are *bounded past* $\mathsf{LTL{+}P_P}$ formulas, is not able to distinguish the state sequence ${}^{i,i}\sigma^{j}$ with $i < j$ (which is a model of φ_{G}) from ${}^{i,k}\sigma^{j}$ with $k > j$ (which is *not* a model of φ_{G}), for sufficiently large values of i, j and k. The choice for the values of the three indices is based on the values of the *temporal depth* of α and β. Since the *globally* operator is a special case of the *release* operator, that is $\mathsf{G}\alpha \equiv \bot\mathsf{R}\alpha$, it suffices to prove the property for formulas of type $\alpha\mathsf{R}\beta$. We first prove two fundamental properties that show that, for any interval of ${}^{i,i}\sigma^{j}$ of length at most d (for any $d \in \mathbb{N}$), we can find the exact same interval in ${}^{i,k}\sigma^{j}$, and *vice versa*. The two properties are proved by the following lemma. Figure 2 shows the idea of this correspondence.

Lemma 5 Let $d \in \mathbb{N}$. For all $i \geq d$, for all $j \geq i + d$, and for all $k \geq j + d$, it holds that:

Property 1: $\forall n' \geq 0 \cdot \exists n \geq 0 \cdot {}^{i,k}\sigma^{j}_{[n'-d,n']} = {}^{i,i}\sigma^{j}_{[n-d,n]}$

Property 2: $\forall n \geq 0 \cdot \exists n' \geq 0 \cdot {}^{i,i}\sigma^{j}_{[n-d,n]} = {}^{i,k}\sigma^{j}_{[n'-d,n']}$

Lemma 5 allows to prove that the state sequences ${}^{i,i}\sigma^{j}$ and ${}^{i,k}\sigma^{j}$ are indistinguishable for each formula of type $\alpha\mathsf{R}\beta$ (and, consequently, of type $\mathsf{G}\alpha$), with $\alpha, \beta \in \mathsf{LTL{+}P_{BP}}$.

Lemma 6 Let $\alpha, \beta \in \mathsf{LTL{+}P_{BP}}$, and let $d = \max\{D(\alpha), D(\beta)\}$ be the maximum between the temporal depths of α and β. It holds that ${}^{i,i}\sigma^{j} \models \alpha\mathsf{R}\beta$ iff ${}^{i,k}\sigma^{j} \models \alpha\mathsf{R}\beta$, for all $i \geq d$, for all $j \geq i + d$, and for all $k \geq j + d$.

By using Lemma 6 as the proof for the base case, we prove by induction on the structure of the formula that any formula in $\mathsf{Normal\text{-}LTL_{EBR}}$ is not able to distinguish the state sequences ${}^{i,i}\sigma^{j}$ and ${}^{i,k}\sigma^{j}$ for sufficiently large values of i, j, k. In the following, given a formula $\psi \in \mathsf{Normal\text{-}LTL_{EBR}}$, we will denote with m_ψ the maximum number of nested *next* operators in ψ, and with d_ψ the maximum temporal depth between all its $\mathsf{LTL{+}P_{BP}}$-subformulas.

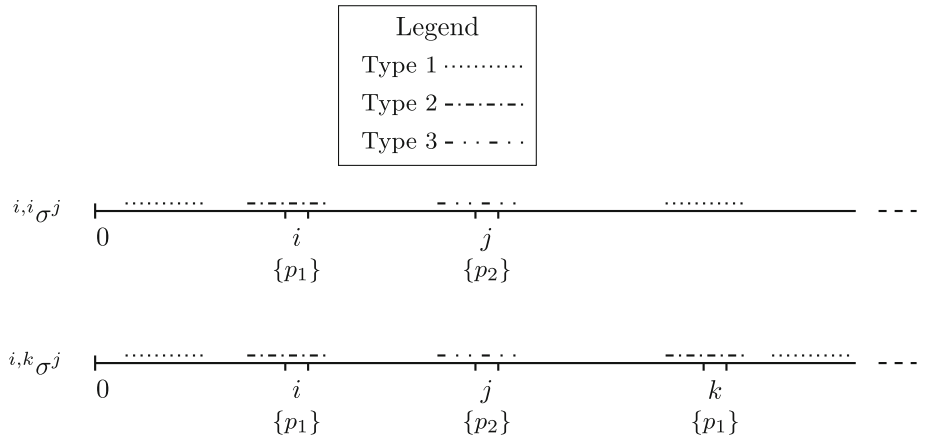


Fig. 2 Graphical exemplification of Lemma 5

Lemma 7 Let $\psi \in \text{Normal-LTL}_{\text{EBR}}$. It holds that $^{i,i}\sigma^j \models \psi$ iff $^{i,k}\sigma^j \models \psi$, for all $i \geq m_\psi + d_\psi$, for all $j \geq i + d_\psi$, and for all $k \geq j + d_\psi$.

Thanks to Lemma 7, it is simple to prove the undefinability of $\mathsf{G}(p_1 \vee \mathsf{G}(p_2))$ in $\mathsf{LTL}_{\mathsf{EBR}}$, establishing that $\mathsf{LTL}_{\mathsf{EBR}}$ is *strictly less expressive* than Safety-LTL.

Theorem 3 $[\![\mathsf{LTL}_{\mathsf{EBR}}]\!] \subsetneq [\![\text{Safety-LTL}]\!]$.

Proof Consider the formula $\varphi_{\mathsf{G}} := \mathsf{G}(p_1 \vee \mathsf{G}(p_2))$. We prove that there does *not* exists a formula $\psi \in \mathsf{LTL}_{\mathsf{EBR}}$ such that $\mathcal{L}(\psi) = \mathcal{L}(\varphi_{\mathsf{G}})$. We proceed by contradiction. Suppose that there exists a formula $\psi \in \mathsf{LTL}_{\mathsf{EBR}}$ such that $\mathcal{L}(\psi) = \mathcal{L}(\varphi_{\mathsf{G}})$. By Lemma 4, there exists a formula $\psi' \in \text{Normal-LTL}_{\mathsf{EBR}}$ such that $\mathcal{L}(\psi) = \mathcal{L}(\psi')$. Let $m_{\psi'}$ be the maximum number of *nested next* operators in ψ', and let $d_{\psi'}$ be the maximum temporal depth between all the $\mathsf{LTL{+}P}_{\mathsf{BP}}$-subformulas in ψ'. Let k, i and j be three indices such that: (i) $i \geq m_{\psi'} + d_{\psi'}$; (ii) $j \geq i + d_{\psi'}$; (iii) and $k \geq j + d_{\psi'}$. Consider the two state sequences $^{i,i}\sigma^j$ and $^{i,k}\sigma^j$. By Lemma 7, $^{i,i}\sigma^j \in \mathcal{L}(\psi')$ if and only if $^{i,k}\sigma^j \in \mathcal{L}(\psi')$, that is $^{i,i}\sigma^j \in \mathcal{L}(\varphi_{\mathsf{G}})$ if and only if $^{i,k}\sigma^j \in \mathcal{L}(\varphi_{\mathsf{G}})$. Since it holds that $^{i,i}\sigma^j \in \mathcal{L}(\varphi_{\mathsf{G}})$ but $^{i,k}\sigma^j \notin \mathcal{L}(\varphi_{\mathsf{G}})$, this is clearly a contradiction. □

Corollary 1 $[\![\mathsf{LTL}_{\mathsf{EBR}}]\!] \subsetneq [\![\mathsf{LTL}_{\mathsf{EBR}}\text{+P}]\!]$.

4.5 Expressiveness of GR-EBR

We are now ready to study the expressiveness of GR-EBR by comparing it with (i) $\mathsf{LTL}_{\mathsf{EBR}}$+P, (ii) Manna and Pnueli's *temporal hierarchy*, and (iii) GR(1).

4.6 Comparison with

$\mathsf{LTL}_{\mathsf{EBR}}$+P and the (co-)safety LTL fragment

We start by comparing GR-EBR with $\mathsf{LTL}_{\mathsf{EBR}}$+P. Each $\mathsf{LTL}_{\mathsf{EBR}}$+P formula ϕ is a GR-EBR formula as well. In fact, the formula ϕ is equivalent to $(\top \wedge \top) \rightarrow (\phi \wedge \top)$ which belongs to GR-EBR, thus: $\mathsf{LTL}_{\mathsf{EBR}}\text{+P} \subseteq$ GR-EBR and of course $[\![\mathsf{LTL}_{\mathsf{EBR}}\text{+P}]\!] \subseteq [\![\text{GR-EBR}]\!]$.

From Theorem 8, it follows that any safety language definable in LTL is definable in GR-EBR as well. In addition, GR-EBR is *strictly* more expressive than $\mathsf{LTL}_{\mathsf{EBR}}$+P, since the former can express also non-safety properties, like $\mathsf{G}p \rightarrow \mathsf{G}q$, and thus:

$$[\![\mathsf{LTL}_{\mathsf{EBR}}\text{+P}]\!] \subsetneq [\![\text{GR-EBR}]\!]$$
$$[\![\text{LTL}]\!] \cap \text{SAFETY} \subsetneq [\![\text{GR-EBR}]\!]$$

Thanks to the implication in the syntax of GR-EBR, we can also define co-safety properties. Actually, it turns out that we can express all the LTL-definable co-safety properties. Take a language $\mathcal{L} \in [\![\text{LTL}]\!] \cap \text{coSAFETY}$. By Theorem 1, $\mathcal{L} = \mathcal{L}(\mathsf{F}\alpha)$, for a given formula $\alpha \in \mathsf{LTL{+}P}_{\mathsf{P}}$. Now, any formula of type $\mathsf{F}\alpha$ (with $\alpha \in \mathsf{LTL{+}P}_{\mathsf{P}}$) can be easily defined with the syntax of GR-EBR by the formula $(\mathsf{G}\neg\alpha) \rightarrow \bot$. Therefore:

$$[\![\text{LTL}]\!] \cap \text{coSAFETY} \subsetneq [\![\text{GR-EBR}]\!]$$

4.7 Comparison with the temporal hierarchy

We consider the *Reactivity(N)* class (R(N)) of the Temporal Hierarchy (see Sect. 2.4) for $N = 1$. We recall that the resulting class, R(1) (i.e., *Reactivity(1)*), comprises all and only the formulas of type:

$$\mathsf{G}\,\mathsf{F}\alpha \rightarrow \mathsf{G}\,\mathsf{F}\beta$$

with $\alpha, \beta \in \mathsf{LTL{+}P}_{\mathsf{P}}$. Since any R(1)-formula is syntactically also a GR(1)-formula, it is straightforward to see that GR-EBR is at least as expressive as R(1).

Proposition 3 $[\![\text{R(1)}]\!] \subseteq [\![\text{GR-EBR}]\!]$.

4.8 Comparison with GR(1)

Recall from Def. 2 that GR(1) is defined as the set of formulas of this type:

$$\left(\alpha_I \wedge \mathsf{G}(\alpha_T) \wedge \bigwedge_{i=1}^{m} \mathsf{G}\,\mathsf{F}\alpha_i\right) \rightarrow \left(\beta_I \wedge \mathsf{G}(\beta_T) \wedge \bigwedge_{j=1}^{n} \mathsf{G}\,\mathsf{F}\beta_j\right)$$

where

(i) α_I, β_I are Boolean formulas,
(ii) α_T, β_T are LTL+P_{BF} formulas where modality X can only occur in a non-nested form, and
(iii) α_i, β_j are LTL+P_{P} formulas, for all $i \in \{1, \ldots, m\}$ and $j \in \{1, \ldots, n\}$, for some $m, n \in \mathbb{N}$.

We prove that GR-EBR is expressively equivalent to GR(1).

Theorem 4 $[\![\text{GR-EBR}]\!] = [\![\text{GR(1)}]\!]$.

Proof We first prove that $[\![\text{GR-EBR}]\!] \subseteq [\![\text{GR(1)}]\!]$. Let ϕ be a formula of GR-EBR. By definition, ϕ is of this type:

$$\left(\psi^1_{ebr} \wedge \bigwedge_{i=1}^{m} \mathsf{G}\,\mathsf{F}\alpha_i\right) \rightarrow \left(\psi^2_{ebr} \wedge \bigwedge_{j=1}^{n} \mathsf{G}\,\mathsf{F}\beta_j\right)$$

By Theorem 8, there exists two LTL+P_{P} formulas α_1, α_2 such that $\psi^1_{ebr} \equiv \mathsf{G}\alpha_1$ and $\psi^2_{ebr} \equiv \mathsf{G}\alpha_2$. Therefore, ϕ is equivalent to:

$$\left(\top \wedge \mathsf{G}\alpha_1 \wedge \bigwedge_{i=1}^{m} \mathsf{G}\,\mathsf{F}\alpha_i\right) \rightarrow \left(\top \wedge \mathsf{G}\alpha_2 \wedge \bigwedge_{j=1}^{n} \mathsf{G}\,\mathsf{F}\beta_j\right)$$

which syntactically belongs to GR(1). This proves that $[\![\text{GR-EBR}]\!] \subseteq [\![\text{GR(1)}]\!]$.

For proving the opposite inclusion, consider a formula $\phi \in \text{GR(1)}$. By definition, ϕ is of the following form:

$$(\alpha_I \wedge \mathsf{G}(\alpha_T) \wedge \bigwedge_{i=1}^{m} \mathsf{G}\,\mathsf{F}\alpha_i) \rightarrow (\beta_I \wedge \mathsf{G}(\beta_T) \wedge \bigwedge_{j=1}^{n} \mathsf{G}\,\mathsf{F}\beta_j)$$

Without loss of generality, consider the left-hand side of the implication (the argument works in the very same way for the right-hand side).

By applying the *pastification method* [9, 35] to α_T, we lift the occurrences of the *next* operator X in α_T to the top level and obtain an equivalent formula of the form $\mathsf{X}\alpha'_T$ such that $\alpha'_T \in \text{LTL+}\mathsf{P}_{\mathsf{P}}$. Now, since, for any state sequence σ, it holds that $\sigma, i \models \mathsf{Y}\top$ if and only if $i > 0$, that is, i is *not* the initial state, it holds that $\mathsf{G}(\mathsf{X}\alpha'_T)$ is equivalent to $\mathsf{G}(\mathsf{Y}\top \rightarrow \alpha'_T)$, which is of the desired form $\mathsf{G}\alpha'$ with $\alpha' := \mathsf{Y}\top \rightarrow \alpha'_T$.

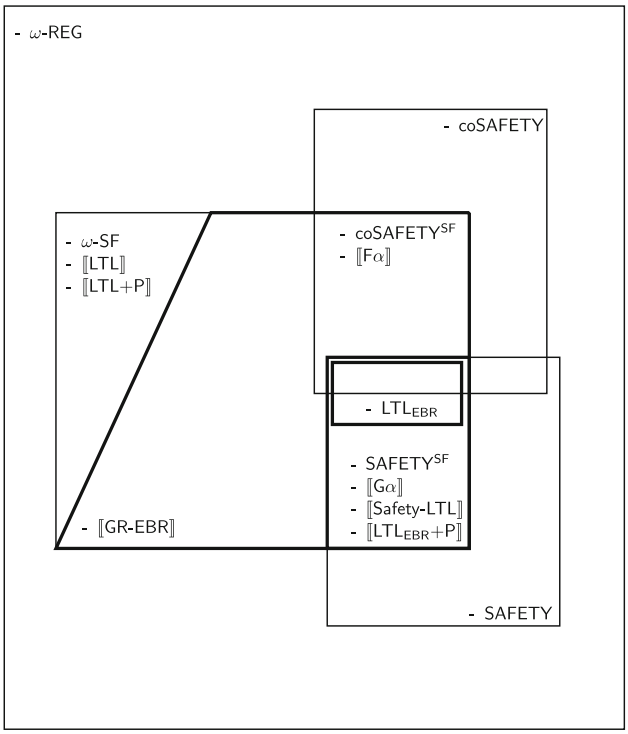


Fig. 3 Comparison of the expressive power of LTL$_{\text{EBR}}$+P, LTL$_{\text{EBR}}$ and GR-EBR. For ease of exposition, we highlighted the rectangles corresponding to LTL$_{\text{EBR}}$+P, LTL$_{\text{EBR}}$ and GR-EBR with thick borders

Therefore, we obtain that any GR(1) formula can be translated into an equivalent one of the form:

$$(\alpha_I \wedge \mathsf{G}(\alpha') \wedge \bigwedge_{i=1}^{m} \mathsf{G}\,\mathsf{F}\alpha_i) \rightarrow (\beta_I \wedge \mathsf{G}(\beta') \wedge \bigwedge_{j=1}^{n} \mathsf{G}\,\mathsf{F}\beta_j)$$

where $\alpha', \beta' \in \text{LTL+}\mathsf{P}_{\mathsf{P}}$. Since both $\alpha_I \wedge \mathsf{G}(\alpha')$ and $\beta_I \wedge \mathsf{G}(\beta')$ syntactically belong to LTL$_{\text{EBR}}$+P, this proves that the whole formula belongs to GR-EBR and thus $[\![\text{GR(1)}]\!] \subseteq [\![\text{GR-EBR}]\!]$. □

Figure 3 shows the comparison between the expressive power LTL$_{\text{EBR}}$+P, LTL$_{\text{EBR}}$ and GR-EBR and the other fragments that we considered.

Our goal is to solve the realizability problem for GR-EBR specifications by reducing it to realizability subproblems for safety specifications. The reduction to safety, which we will give in Sect. 5, generates a safety formula for each integer k, in such a way to guarantee the following important properties: (i) *soundness*, ensuring that the realizability of the k^{th} subproblem implies the realizability of the starting formula, and (ii) *completeness*, establishing the existence of an upper bound μ such that the unrealizability of all the k^{th} subproblems with $k \leq \mu$ implies the unrealizability of the starting formula. In Sect. 5, we describe a symbolic algorithm that performs a safety reduction for the realizability problem

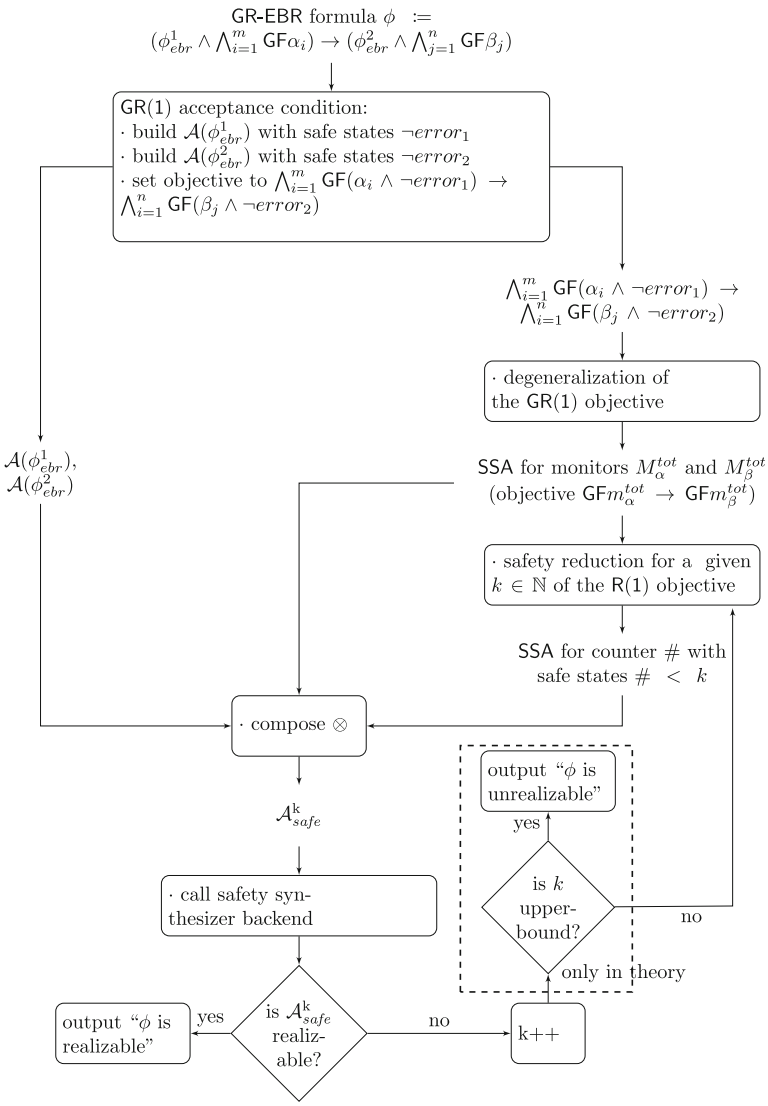


Fig. 4 Low-level view of the procedure for the realizability of GR-EBR form

from GR-EBR specifications. In order to prove completeness of such algorithm, in Sect. 6 we introduce a general framework for deriving (sound and) complete safety reductions for arbitrary fragments of LTL, and we instantiate this framework to the GR-EBR case.

5 GR-EBR synthesis: the algorithm

In this section, we describe the algorithm for solving realizability of GR-EBR specifications.

The procedure consists of three steps. Firstly, we build the product between the two symbolic safety automata for the safety parts of both assumptions and guarantees, and we define a GR(1) accepting condition for it in order to maintain the language-equivalence with the original formula. The second step consists of a so-called *degeneralization*, that, by using deterministic monitors, turns the GR(1) accepting condition into a Reactivity(1) (R(1), for short) condition. The third and last step, that is the core of the procedure, reduces the realizability problem over the above automaton to a *sequence* of safety games (Def. 15), that is, realizability problems over safety (and symbolic) automata $\mathcal{A}^{k}_{safe}$, one for each index $k \in \mathbb{N}$. The structure of the full procedure is depicted in Fig. 4.

Since, in the worst case, the upperbound given by the safety reduction is doubly exponential in the length of the formula, in practice it is useful to use our algorithm in parallel with another one that checks for the unrealizability of the specification. The first that terminates stops the other and, thus, the entire procedure, as summarized in Fig. 5. We remark that we cannot check the unrealizability of ϕ by solving the dualized game (i.e., looking for a Moore-type strategy of Environment) for $\neg\phi$, because GR-EBR and $\mathsf{LTL}_{\mathsf{EBR}}$+P are *not* closed under complementation.

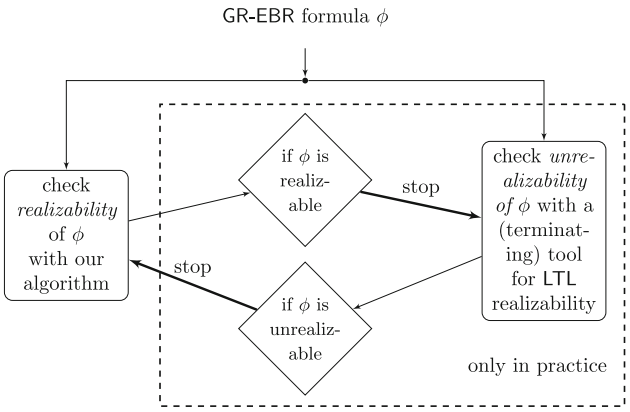


Fig. 5 High-level view of our procedure for the realizability of GR-EBR formulas

Finally, note that, as for now, there is no incrementality between an iteration and the next one, because of the lack of incremental safety synthesizers. The only point that we save between one iteration and the next one is the construction of the two symbolic safety automata, which is performed only once during the procedure.

5.1 Construction of the automaton with a GR(1) condition

In this part, we describe the first step of the algorithm. Starting from a GR-EBR formula $\phi := (\phi^1_{ebr} \wedge \bigwedge_{i=1}^{m} \mathsf{G}\,\mathsf{F}\alpha_i) \rightarrow (\phi^2_{ebr} \wedge \bigwedge_{j=1}^{n} \mathsf{G}\,\mathsf{F}\beta_j)$, the objective is to obtain an automaton $\mathcal{A}$ such that: (i) it has a GR(1) accepting condition, and (ii) it recognizes the same language of ϕ, i.e., $\mathcal{L}(\phi) = \mathcal{L}(\mathcal{A})$. In order to do that, we first build the two symbolic safety automata for the safety parts of both the assumptions and the guarantees, that is for ϕ^1_{ebr} and ϕ^2_{ebr}. Since by definition both are $\mathsf{LTL}_{\mathsf{EBR}}$+P formulas, we use the transformation described in [9], to which the reader is referred for more details.

From now on, let $\mathcal{A}(\phi^1_{ebr})$ and $\mathcal{A}(\phi^2_{ebr})$ be the automata for ϕ^1_{ebr} and ϕ^2_{ebr}, respectively. Let $\mathcal{A}_{ebr}$ be the product automaton $\mathcal{A}(\phi^1_{ebr}) \times \mathcal{A}(\phi^2_{ebr})$. The question is how to set the acceptance condition of $\mathcal{A}_{ebr}$ such that the conditions (i) and (ii) above are fulfilled. We answer this question by examining how the automata $\mathcal{A}(\phi^1_{ebr})$ and $\mathcal{A}(\phi^2_{ebr})$ are made internally. Take for example the formula $\mathsf{G}\,p$ (for some atomic proposition $p \in \Sigma$). The safety automaton corresponding to this formula (but the same holds for all $\mathsf{LTL}_{\mathsf{EBR}}$+P formulas) comprises an *error bit* as one of its state variables, let us call it `error`, which is initially set to be `false` (when the automaton has not start to read any letter of the input word). The transition function for `error` is deterministic and updates `error` to `true` if $\neg p$ holds in the current state, or keeps its value otherwise. The set of safe states comprises all and only those states in which `error` is `false`. In a symbolic setting, this is expressed by the formula $\mathsf{G}\neg\texttt{error}$. In this way, p is forced to hold constantly in all (and only) the words accepted by the automaton.

A crucial property of each error bit is *monotonicity*: once `error` is set to `true`, it can never be set to `false` again. Formally, given a run τ of the automaton, it holds that, if there exists $i \geq 0$ such that $\tau(i) \models \texttt{error}$, then $\tau(j) \models \texttt{error}$, for all $j \geq i$. Monotonicity of error bits allows us to express an accepting condition of type $\mathsf{G}\neg\texttt{error}$ in terms of $\mathsf{G}\,\mathsf{F}\neg\texttt{error}$ (which is of the GR(1) type) by maintaining the equivalence.

Lemma 1 $\mathsf{G}\neg\texttt{error} \equiv \mathsf{G}\,\mathsf{F}\neg\texttt{error}$, *under the assumption that* `error` *is monotone.*

Proof Consider a run τ of an automaton with an accepting condition of the type $\mathsf{G}\neg\texttt{error}$. If $\tau \models \mathsf{G}\neg\texttt{error}$ then of course $\tau \models \mathsf{G}\,\mathsf{F}\neg\texttt{error}$. Suppose now that $\tau \models \mathsf{G}\,\mathsf{F}\neg\texttt{error}$. If by contradiction we suppose that $\tau \not\models \mathsf{G}\neg\texttt{error}$, we have that there exists an $i \geq 0$ such that $\tau(i) \models \texttt{error}$. By the monotonicity property, this would mean that also $\tau(j) \models \texttt{error}$, for all $j \geq i$, that is $\tau \models \mathsf{F}\,\mathsf{G}\texttt{error}$, but this a contradiction with our hypothesis. Therefore, we proved that changing the acceptance condition of an automaton from a $\mathsf{G}\neg\texttt{error}$ to $\mathsf{G}\,\mathsf{F}\neg\texttt{error}$ maintains the equivalence. □

Let $\texttt{error}_1$ and $\texttt{error}_2$ be the error bits of $\mathcal{A}(\phi^1_{ebr})$ and $\mathcal{A}(\phi^2_{ebr})$, respectively. Let $\mathcal{A}^{\mathsf{GR(1)}}_{\text{ebr}}$ be the automaton obtained from $\mathcal{A}_{ebr}$ by replacing its acceptance condition with the following GR(1) condition:

$$(\mathsf{G}\,\mathsf{F}\neg\texttt{error}_1 \wedge \bigwedge_{i=1}^{m} \mathsf{G}\,\mathsf{F}\alpha_i) \rightarrow (\mathsf{G}\,\mathsf{F}\neg\texttt{error}_2 \wedge \bigwedge_{j=1}^{n} \mathsf{G}\,\mathsf{F}\beta_j) \tag{3}$$

The intuition is that $\texttt{error}_1$ and $\texttt{error}_2$ keep track of the *safety* parts of ϕ, that is ϕ^1_{ebr} and ϕ^2_{ebr}. The following lemma proves the equivalence between ϕ and $\mathcal{A}^{\mathsf{GR(1)}}_{\text{ebr}}$.

Lemma 2 *Let* ϕ *be an* GR-EBR *formula. It holds that* $\mathcal{L}(\phi) = \mathcal{L}(\mathcal{A}^{\mathsf{GR(1)}}_{\text{ebr}})$.

Proof Let $\phi \in$ GR-EBR . ϕ is of the following form:

$$(\phi^1_{ebr} \rightarrow \bigwedge_{i=1}^{m} \mathsf{G}\,\mathsf{F}\alpha_i) \rightarrow (\phi^2_{ebr} \rightarrow \bigwedge_{j=1}^{n} \mathsf{G}\,\mathsf{F}\beta_j)$$

By the theorems proved in [9], it holds that $\mathcal{L}(\phi^1_{ebr}) = \mathcal{L}(\mathcal{A}(\phi^1_{ebr}))$ and $\mathcal{L}(\phi^2_{ebr}) = \mathcal{L}(\mathcal{A}(\phi^2_{ebr}))$.

Consider first the left-to-right direction. Let $\sigma \in \mathcal{L}(\phi)$. We prove that $\sigma \in \mathcal{L}(\mathcal{A}^{\mathsf{GR(1)}}_{\text{ebr}})$. Each $\sigma \in \mathcal{L}(\phi)$ is such that: a. either $\sigma \models \neg\phi^1_{ebr} \vee \neg(\bigwedge_{i=1}^{m} \mathsf{G}\,\mathsf{F}\alpha_i)$, b. or $\sigma \models$

$\phi^2_{ebr} \wedge \bigwedge_{j=1}^{n} \mathsf{G}\,\mathsf{F}\beta_j$ Recall that $\mathcal{A}^{\mathsf{GR(1)}}_{\text{ebr}}$ is defined as the product automaton $\mathcal{A}(\phi^1_{ebr}) \times \mathcal{A}(\phi^2_{ebr})$ with the acceptance condition α defined as $(\mathsf{G}\,\mathsf{F}\neg\texttt{error}_1 \wedge \bigwedge_{i=1}^{m} \mathsf{G}\,\mathsf{F}\alpha_i) \rightarrow (\mathsf{G}\,\mathsf{F}\neg\texttt{error}_1 \wedge \bigwedge_{j=1}^{n} \mathsf{G}\,\mathsf{F}\beta_j)$.

Consider case *a*. If $\sigma \models \neg\phi^1_{ebr} \vee \neg(\bigwedge_{i=1}^{m} \mathsf{G}\,\mathsf{F}\alpha_i)$, then *the* run induced by σ in $\mathcal{A}^{\mathsf{GR(1)}}_{\text{ebr}}$ is such that at least one of the following two cases hold:

a.1. either $\exists i \geq 0$ such that $\tau(i) \models \texttt{error}_1$, that is $\tau \models \mathsf{F}(\texttt{error}_1)$. In this case, we exploit *monotonicity* of $\texttt{error}_1$. Since $\tau \models \mathsf{F}(\texttt{error}_1)$, it also holds that $\tau \models \mathsf{F}\,\mathsf{G}(\texttt{error}_1)$, that is $\tau \not\models \mathsf{G}\,\mathsf{F}(\neg\texttt{error}_1)$. As a consequence, $\tau \models \alpha$, where α is the acceptance condition of $\mathcal{A}^{\mathsf{GR(1)}}_{\text{ebr}}$, and thus $\sigma \in \mathcal{L}(\mathcal{A}^{\mathsf{GR(1)}}_{\text{ebr}})$.

a.2. or $\tau \models \neg \bigwedge_{i=1}^{m} \mathsf{G}\,\mathsf{F}\alpha_i$. In this case, of course, $\tau \models \alpha$ (that is, τ satisfies the acceptance condition of $\mathcal{A}^{\mathsf{GR(1)}}_{\text{ebr}}$), and thus $\sigma \in \mathcal{L}(\mathcal{A}^{\mathsf{GR(1)}}_{\text{ebr}})$.

Consider now the case *b*. If $\sigma \models \phi^2_{ebr} \wedge \bigwedge_{j=1}^{n} \mathsf{G}\,\mathsf{F}\beta_j$, then $\sigma \models \phi^2_{ebr}$ and $\sigma \models \bigwedge_{j=1}^{n} \mathsf{G}\,\mathsf{F}\beta_j$. Therefore, *the* run induced by σ in $\mathcal{A}^{\mathsf{GR(1)}}_{\text{ebr}}$ is such that $\tau \models \mathsf{G}(\neg\texttt{error}_2) \wedge \bigwedge_{j=1}^{n} \mathsf{G}\,\mathsf{F}\beta_j$, that implies that $\tau \models \mathsf{G}\,\mathsf{F}(\neg\texttt{error}_2) \wedge \bigwedge_{j=1}^{n} \mathsf{G}\,\mathsf{F}\beta_j$. Therefore, $\tau \models \alpha$, and thus $\sigma \in \mathcal{L}(\mathcal{A}^{\mathsf{GR(1)}}_{\text{ebr}})$. The opposite direction can be proved similarly. □

5.2 Optimization

Instead of considering the acceptance condition described in Eq. (3), we propose to repeat the error bits inside each fairness condition.

$$\left(\bigwedge_{i=1}^{m} \mathsf{G}\,\mathsf{F}(\alpha_i \wedge \neg\texttt{error}_1)) \rightarrow (\bigwedge_{j=1}^{n} \mathsf{G}\,\mathsf{F}(\beta_j \wedge \neg\texttt{error}_2)\right) \tag{4}$$

This may be helpful during the safety game solving, since it creates a redundancy that the solver may exploit during the search. Obviously, this maintains the equivalence.

5.3 Degeneralization

The objective of this part is to transform the GR(1) accepting condition of the automaton $\mathcal{A}^{\mathsf{GR(1)}}_{\text{ebr}}$, that is of the form $\bigwedge_{i=1}^{m} \mathsf{G}\,\mathsf{F}\alpha_i \rightarrow \bigwedge_{j=1}^{n} \mathsf{G}\,\mathsf{F}\beta_j$, into a R(1) accepting condition of the form $\mathsf{G}\,\mathsf{F}\alpha \rightarrow \mathsf{G}\,\mathsf{F}\beta$. In this context, we will use the term *monitor* as a synonym of *symbolic deterministic safety automaton* (recall Def. 6).

In order to accomplish the task, for each α_i (resp. for each β_i), we define a monitor M_{α_i} (resp. M_{β_i}) whose only state variable is set to *true* when α_i (resp. β_i) has been read and is reset to *false* when all the α_i (resp. β_i) have been read. For this last condition, we define the monitors M^{tot}_{α} and M^{tot}_{β}.

Let M_{α_i} and M^{tot}_{α} be the monitors such that (i) their input alphabet is 2^{Σ} (where Σ is the alphabet of the starting GR-EBR formula); (ii) their set of *state variables* are $\{m_{\alpha_i}\}$ and $\{m^{tot}_{\alpha}\}$, respectively; (iii) their set of safe states are their reachable states; and (iv) their transition relations are the following (written in the SMV language [36]).

```
init(m_{α_i}) := 0
next(m_{α_i}) := case
  α_i          :  1
  m^{tot}_α    :  0
  default      :  m_{α_i}
esac
```

```
init(m^{tot}_α) := 0
next(m^{tot}_α) := case
  m_{α_1} ∧ ··· ∧ m_{α_m} : 1
  default                 : 0
esac
```

We define M_{β_j} and M^{tot}_{β} in a similar way as M_{α_i} and M^{tot}_{α}, respectively, but with α_i substituted with β_j and α substituted with β.

Let $\mathcal{A}^{\mathsf{R(1)}}_{\text{degen}}$ be the product between M_{α_i} (for each $0 \leq i \leq m$), M_{β_j}(for each $0 \leq j \leq n$), M^{tot}_{α} and M^{tot}_{β}, whose accepting condition is the R(1) formula $\mathsf{G}\,\mathsf{F}m^{tot}_{\alpha} \rightarrow \mathsf{G}\,\mathsf{F}m^{tot}_{\beta}$. We can prove the following lemma, which states that this step of the algorithm maintains the equivalence.

Lemma 3 $\mathcal{L}(\mathcal{A}^{\mathsf{GR(1)}}_{\text{ebr}}) = \mathcal{L}(\mathcal{A}_{ebr} \times \mathcal{A}^{\mathsf{R(1)}}_{\text{degen}})$.

Proof We prove separately the two directions. Consider first the right-to-left direction. Let σ be an infinite word of $\mathcal{L}(\mathcal{A}_{ebr} \times \mathcal{A}^{\mathsf{R(1)}}_{\text{degen}})$. Then σ is a word in $\mathcal{L}(\mathcal{A}_{ebr})$. Moreover, σ is a word in $\mathcal{L}(\mathcal{A}^{\mathsf{R(1)}}_{\text{degen}})$ and thus there exists a run τ induced by σ such that $\tau \models \mathsf{G}\,\mathsf{F}m^{tot}_{\alpha} \rightarrow \mathsf{G}\,\mathsf{F}m^{tot}_{\beta}$, that is, $\tau \models \mathsf{F}\,\mathsf{G}\neg m^{tot}_{\alpha} \vee \mathsf{G}\,\mathsf{F}m^{tot}_{\beta}$. We divide in cases:

- if $\tau \models \mathsf{F}\,\mathsf{G}\neg m^{tot}_{\alpha}$, then by the semantics of the temporal operators F and G, there exists an $i \geq 0$ such that for all $j \geq i$, $\tau_j \models \neg m^{tot}_{\alpha}$. By construction of the monitors m^{tot}_{α}, this means that there exists an $i \geq 0$ such that for all $j \geq i$, $\tau_j \models \bigvee_{k=1}^{m} \neg m_{\alpha_k}$. This implies that, there exists a $k \in [1, m]$ and an $i \geq 0$ such that for all $j \geq i$, such that $\tau_j \models \neg m_{\alpha_k}$. Indeed, suppose by contradiction that it is not so: then for all $k \in [1, m]$, there exists infinitely many positions $i \geq 0$ such that $\tau_i \models m_{\alpha_k}$. This would mean that the monitor M^{tot}_{α} is set to *true* infinitely many times, that is $\mathsf{G}\,\mathsf{F}m^{tot}_{\alpha}$, but this is a contradiction with our hypothesis. Therefore, it holds that $\tau \models \bigvee_{k=1}^{m} \mathsf{F}\,\mathsf{G}\neg m_{\alpha_k}$, and thus also that $\tau \models \bigwedge_{i=1}^{m} \mathsf{G}\,\mathsf{F}\alpha_i \rightarrow \bigwedge_{j=1}^{n} \mathsf{G}\,\mathsf{F}\beta_j$. Overall, since τ is induced by σ, we have that σ is a word of $\mathcal{L}(\mathcal{A}_{ebr})$ that induces a run τ such that $\tau \models \bigwedge_{i=1}^{m} \mathsf{G}\,\mathsf{F}\alpha_i \rightarrow \bigwedge_{j=1}^{n} \mathsf{G}\,\mathsf{F}\beta_j$, that is $\sigma \in \mathcal{L}(\mathcal{A}^{\mathsf{GR(1)}}_{\text{ebr}})$.

- If otherwise $\tau \models \mathsf{G}\,\mathsf{F} m_\beta^{tot}$, then there exists infinitely many positions $i \geq 0$ such that $\tau_i \models m_\beta^{tot}$. Moreover, it holds that for all $i_1 \geq 0$ and for all $i_2 \geq i_1$, if $\tau_{i_1} \models m_\beta^{tot}$ and $\tau_{i_2} \models m_\beta^{tot}$, then, for all $1 \leq k \leq n$, there exists a $i_1 \leq j \leq i_2$ such that $\tau_j \models m_{b_k}$. Putting together these two points, we have that for all $1 \leq k \leq n$, there exists infinitely many $i \geq 0$ such that $\tau_i \models m_{b_k}$. That is, $\tau \models \bigwedge_{k=1}^{n} \mathsf{G}\,\mathsf{F} m_{b_k}$. By definition of the monitors M_{β_i} and since τ is induced by σ, we have that σ is a word in $\mathcal{L}(\mathcal{A}_{ebr})$ that induces a run τ such that $\tau \models \bigwedge_{i=1}^{m} \mathsf{G}\,\mathsf{F}\alpha_i \rightarrow \bigwedge_{j=1}^{n} \mathsf{G}\,\mathsf{F}\beta_j$. That is, $\sigma \in \mathcal{L}(\mathcal{A}_{\text{ebr}}^{\mathsf{GR(1)}})$.

The proof of the left-to-right direction is specular. □

5.4 Related work

Our degeneralization step is similar to the one proposed in [37] for transforming a GR(1) condition to a *one-pair Streett condition*, in the context of parity game solving. The main difference is that, in this paper, we do *not* fix any order on the visits to the conditions α_i (resp. β_i). For example, M_α^{tot} is set to *true* whenever all the α_i have been read, no matter the order. As noted in [38], this has the potential to be more effective than imposing an order to the visits (like in [37]), for example in the case where the order is $\langle \beta_1, \beta_2, \ldots, \beta_n \rangle$ but Controller can never satisfy fairness β_i after having satisfied first the fairness β_{i+1}.

5.5 Reduction to safety for reactivity(1) objectives

In this part, we describe a safety reduction tailored for Reactivity(1) objectives. In Sect. 6, we will prove its completeness. We will apply this reduction on the automaton $\mathcal{A}_{\text{degen}}^{\mathsf{R(1)}}$ obtained from the previous step. The intuition is to use a *counter* to count and limit the number of positions, after a position in which m_β^{tot} holds, in which $m_\alpha^{tot} \wedge \neg m_\beta^{tot}$ holds. We define the counter as follows.

Definition 18 (*Counter for the reactivity(1) objective*) Let $\mathcal{A}_{\#^{\rightarrow}_{\alpha,\beta}}^{k}$ be the symbolic deterministic safety automaton whose only state variable is $\#^{\rightarrow}_{\alpha,\beta}$, whose set of safe states is represented by the formula $\mathsf{G}(\#^{\rightarrow}_{\alpha,\beta} < k)$ and whose transition relation is the following:

```
init(#→α,β) := 0
next(#→α,β) := case
  m_β^tot   :  0
  m_α^tot   :  #→α,β + 1
  default   :  #→α,β
esac
```

We define $\mathcal{A}_{safe}^{\mathrm{k}} := \mathcal{A}_{ebr} \times \mathcal{A}_{degen} \times \mathcal{A}_{\#^{\rightarrow}_{\alpha,\beta}}^{k}$, where

- $\mathcal{A}_{ebr}$ is the product between ϕ_{ebr}^1 and ϕ_{ebr}^2 (recall Sect. 5.1);
- $\mathcal{A}_{degen}$ is the automaton obtained from $\mathcal{A}_{\text{ebr}}^{\mathsf{GR(1)}}$ (recall Sect. 5.3) by removing its accepting conditions, i.e., setting it to $\top$; and
- $\mathcal{A}_{\#^{\rightarrow}_{\alpha,\beta}}^{k}$ is the monitor defined in Def. 18.

We set the accepting condition of $\mathcal{A}_{safe}^{\mathrm{k}}$ to be the one of $\mathcal{A}_{\#^{\rightarrow}_{\alpha,\beta}}^{k}$, i.e., $\mathsf{G}(\#^{\rightarrow}_{\alpha,\beta} \leq k)$.

The automaton $\mathcal{A}_{safe}^{\mathrm{k}}$ is a symbolic deterministic safety automaton (recall Def. 9), and therefore it can be used as an arena for the safety game $\langle \mathcal{A}_{safe}^{\mathrm{k}}, \mathcal{C}, \mathcal{U} \rangle$ (recall Def. 15). In practice, we check the realizability of $\mathcal{A}_{safe}^{\mathrm{k}}$ by means of a tool for safety synthesis. We start with $k = 0$, and we check the realizabilty of $\mathcal{A}_{safe}^{\mathrm{k}}$: if Controller has a strategy, we stop, otherwise we increment k and we repeat the cycle.

In the next section, we prove the soundness and the completeness of this algorithm.

6 GR-EBR synthesis: correctness via general safety reductions

In this section, we prove the (soundness and) completeness of the safety reduction described in Sect. 5. To this end, we first formalize what a safety reduction is, and then we introduce a general framework that gives sufficient conditions for obtaining complete safety reduction for arbitrary fragments of LTL. Finally, we instantiate the proposed framework to the GR-EBR case in order to prove the completeness of the algorithm of Sect. 5.

6.1 A framework of safety reductions for LTL+P realizability

The central question of this section is: *how can we obtain a complete safety reduction for the realizability problem of specifications written in (fragments of)* LTL ? In the following, we propose a framework to answer it.

6.2 A sound, but incomplete, safety reduction

Devising a sound and complete safety reduction for realizability is not a trivial task. Consider for example LTL. One could be tempted to define a safety reduction that, given any formula $\phi \in$ LTL, turns ϕ in *negated normal form* (NNF, for short) and then transforms each F and U into the corresponding k-bounded operator, that is $\mathsf{F}^{[0,k]}$ and $\mathsf{U}^{[0,k]}$, respectively, obtaining a safety formula. The resulting reduction would be sound, since the obtained formula *implies* the starting one, but it would not be complete. Consider for example the formula $\phi := \mathsf{G}u \leftrightarrow \mathsf{G}c$. The formula obtained from ϕ by means

of this safety reduction is:

$$\phi_k := (\mathsf{F}^{[0,k]}\neg u \vee \mathsf{G}c) \wedge (\mathsf{F}^{[0,k]}\neg c \vee \mathsf{G}u)$$

Although ϕ is realizable, ϕ_k is unrealizable for each $k \in \mathbb{N}$. In fact, Environment can choose u in the first k steps and $\neg u$ in the step $k+1$: in this way, it falsifies both $\mathsf{F}^{[0,k]}\neg u$ and $\mathsf{G}u$. Since Controller can force exactly one between the formulas $\mathsf{G}c$ and $\mathsf{F}^{[0,k]}\neg c$, we have that ϕ_k is unrealizable, and therefore this particular safety reduction is not complete for realizability.

6.3 Definition of safety reduction

The core and the main novelty of our framework is a link with safety reductions for model checking: in order to design a complete reduction for the realizability problem, one can prove that it is complete for the model checking problem and then use our framework to derive completeness for realizability. On one hand, this allows to prove completeness at the level of model checking, which is simpler than proving completeness for realizability. On the other hand, this opens the possibility of using existing safety reductions already devised for model checking for realizability as well. We start by defining what is a safety reduction in the context of our framework.

Definition 19 (*Safety reduction*) Let $\mathcal{S} \subseteq \mathsf{LTL}$ be a fragment of LTL. A *safety reduction for* $\mathcal{S}$ is a function $[\![\cdot]\!]$ such that, for each formula $\phi \in \mathcal{S}$ over the alphabet Σ, it holds that $[\![\phi]\!] = \{\phi_k\}_{k\in\mathbb{N}}$, where ϕ_k is a *safety* formula over the alphabet Σ such that $\phi_k \rightarrow \phi$, for any $k \in \mathbb{N}$. With $[\![\phi]\!]^k$, we will denote the formula ϕ_k of the set $\{\phi_k\}_{k\in\mathbb{N}}$.

6.4 Link between realizability and model checking

The rationale behind the link between realizability and model checking is the following one: since we can easily view Mealy machines as (a particular type of) Kripke structures and viceversa, and since by Prop. 2 we can restrict realizability to the search of finite strategies representable by Mealy machines, the realizability problem of the $\mathsf{LTL+P}$ formula ϕ can be reduced to checking if there exists a Mealy machine M_g such that $M'_g \models \mathsf{A}\phi$, where M'_g is the Kripke structure *corresponding to* M_g.

The Kripke structure M'_g *corresponding to* the Mealy machine $M_g = (2^{\mathcal{U}}, 2^{\mathcal{C}}, Q, q_0, \delta)$ is defined as $M'_g = (2^{\mathcal{U}\cup\mathcal{C}}, Q', I', T', L')$ where:

1. $Q' = Q \times \{q_U \,|\, U \in 2^{\mathcal{U}}\} \times \{q_C \,|\, C \in 2^{\mathcal{C}}\}$;
2. $I' = \{(q_0, q_U, q_C) \in Q' \,|\, \delta(q_0, U) = (C, q')$ for any $U \in 2^{\mathcal{U}}$, $C \in 2^{\mathcal{C}}$ and $q' \in Q\}$,
3. $T' = \{((q, q_U, q_C), (q', q_{U'}, q_{C'})) \,|\, \delta(q, U) = (C, q')$ for any $U, U' \in 2^{\mathcal{U}}$, $C, C' \in 2^{\mathcal{C}}$, and $q, q' \in Q'\}$ and
4. $L'((q, q_U, q_C)) = U \cup C$.

The Kripke structure M'_g is such that each trace of M'_g corresponds to a word of M_g, and viceversa.

In proving the completeness theorem, we will abstract from the concrete safety reduction and give the conditions for a general safety reduction $[\![\cdot]\!]$ (as defined in Def. 19) to be complete. These conditions are formalized in Def. 20.

Definition 20 (*Complete safety reduction*) Let $\mathcal{S} \subseteq \mathsf{LTL}$ be a fragment of LTL, ϕ a formula in S, and $[\![\cdot]\!]$ a *safety reduction for* $\mathcal{S}$. We say that $[\![\cdot]\!]$ is μ*-complete*, for a given function $\mu : \mathbb{N} \rightarrow \mathbb{N}$ if and only if, for all $\phi \in \mathcal{S}$ and for all Kripke structures M:

$$M \models \mathsf{A}\phi \quad \Leftrightarrow \quad \exists k \leq \mu(|M|) \cdot M \models \mathsf{A}[\![\phi]\!]^k$$

We can finally state the main theorem of our framework, which uses Def. 20 and Prop. 2 in order to establish that if a safety reduction is complete for the model checking problem, then it is complete for the realizability problem as well.

Theorem 5 (Soundness and completeness for $\mathsf{LTL+P}$ *realizability*) *Let $\mathcal{S} \subseteq \mathsf{LTL}$ be a fragment of LTL, $\phi \in \mathcal{S}$ a formula over the input variables $\mathcal{U}$ and output variables $\mathcal{C}$ (with $n = |\phi|$) and $[\![\cdot]\!]$ a μ-complete safety reduction for $\mathcal{S}$, for a given function μ. It holds that:*

$$\phi \textit{ is realizable} \quad \Leftrightarrow \quad \exists k \leq \mu(2^{|\mathcal{U}|} \cdot 2^{|\mathcal{C}|} \cdot 2^{2^{c\cdot n}}) \cdot [\![\phi]\!]^k \textit{ is realizable}$$

Proof We first prove the *soundness*, which corresponds to the right-to-left direction. Suppose there exists a $k \leq \mu(2^{|\mathcal{U}|} \cdot 2^{|\mathcal{C}|} \cdot 2^{2^{c\cdot n}})$ such that $[\![\phi]\!]^k$ is realizable. Then, there exists a strategy $g : (2^{\mathcal{U}})^+ \rightarrow 2^{\mathcal{C}}$ such that $\mathcal{L}(g) \subseteq \mathcal{L}([\![\phi]\!]^k)$. By Prop. 2, there exists a Mealy machine $M_g = (2^{\mathcal{U}}, 2^{\mathcal{C}}, Q, q_0, \delta)$ with input alphabet $2^{\mathcal{U}}$ and output alphabet $2^{\mathcal{C}}$ such that $\mathcal{L}(M_g) \subseteq \mathcal{L}([\![\phi]\!]^k)$. Starting from M_g, let $M'_g = (2^{\mathcal{U}\cup\mathcal{C}}, Q', I', T', L')$ be the *corresponding* Kripke structure. The Kripke structure M'_g is such that each trace of M'_g corresponds to a word of M_g, and viceversa. Therefore all the traces π of M'_g are such that $L'(\pi) \models [\![\phi]\!]^k$, that is $M'_g \models \mathsf{A}[\![\phi]\!]^k$. Since by hypothesis $[\![\cdot]\!]$ is a μ-complete safety reduction, by Def. 20, it holds that $M'_g \models \mathsf{A}\phi$. This means that also $\mathcal{L}(M_g) \subseteq \mathcal{L}(\phi)$. Since M_g is a Mealy machine, this implies that ϕ is realizable.

We now prove *completeness*, which corresponds to the left-to-right direction. Suppose that ϕ is realizable. Since $\phi \in S$ and since $S \subseteq \mathsf{LTL}$, ϕ is an LTL formula as well. Therefore, by Prop. 2, there exists a Mealy machine M_g with input alphabet $2^{\mathcal{U}}$ and output alphabet $2^{\mathcal{C}}$ such that $\mathcal{L}(M_g) \subseteq \mathcal{L}(\phi)$ with at most $2^{2^{c\cdot n}}$ states, for some constant $c \in \mathbb{N}$. From M_g,

we build an equivalent Kripke structure M'_g with input alphabet $\Sigma' = 2^{\mathcal{U}\cup\mathcal{C}}$, as described above for the soundness proof. It holds that $M'_g \models \mathsf{A}\phi$. Since by hypothesis $[\![\cdot]\!]$ is a μ-complete safety reduction for S, and since $|Q'| = 2^{|\mathcal{U}|} \cdot 2^{|\mathcal{C}|} \cdot |Q|$ (where Q and Q' are the set of states of M_g and M'_g, respectively), by Def. 20, there exists a $k \leq \mu(2^{|\mathcal{U}|} \cdot 2^{|\mathcal{C}|} \cdot 2^{2^{c \cdot n}})$ such that $M'_g \models \mathsf{A}[\![\phi]\!]^k$. This means that also $\mathcal{L}(M_g) \subseteq \mathcal{L}([\![\phi]\!]^k)$. Since M_g is a Mealy machine, this means that there exists a $k \leq \mu(2^{|\mathcal{U}|} \cdot 2^{|\mathcal{C}|} \cdot 2^{2^{c \cdot n}})$ such that $[\![\phi]\!]^k$ is realizable. □

6.5 Example

Let $[\![\cdot]\!]_{bo}$ and $[\![\phi]\!]^k_{bo}$ be the reduction and the formula ϕ_k described in the example above, respectively (*bo* stands for *bounded operators*). Since $[\![\phi]\!]^k_{bo}$ contains only universal or bounded temporal operators, it is a safety formula; moreover, $[\![\phi]\!]^k_{bo}$ implies ϕ (for any $k \in \mathbb{N}$); thus $[\![\cdot]\!]_{bo}$ is a safety reduction according to Def. 19. We have already seen that the reduction is sound but not complete for realizability. According to our framework, this reduction is not even complete for the model checking problem (i.e., with respect to Def. 20), and indeed a counterexample can be found in Fig. 1 of [11].

6.6 Novelty and usage

As already mentioned before, a distinguished and important feature of our framework is that it provides a link with safety reductions for the *model checking problem*. This opens the possibility to use model checking safety reductions for the realizability problem as well, provided that the reduction fulfills the requirements in Def. 20. In Sect. 6.9, we will prove that the *concrete* safety reduction for GR-EBR specifications that we described in Sect. 5.5 is *complete* with respect to Def. 20. Using Theorem 5, we will derive a corollary for the completeness of our algorithm.

6.7 In practice

The upper bound for the value of $\mu(\cdot)$ (after which we can answer *unrealizable*) is doubly exponential in the size of the initial formula and therefore, in practice, it is prohibitively large. It follows that usually the *completeness* of a safety reduction can be exploited in practice only for making sure that, starting from a *realizable* specification, we will eventually find a $k \in \mathbb{N}$ such that the k^{th} subproblem is realizable. Therefore, like K-Liveness for model checking [11], we can use our algorithm in parallel with another one that checks for the unrealizability of the specification. The first that terminates stops the other and, thus, the entire procedure.

6.8 Formalization of bounded synthesis

It is worth noting that also *bounded synthesis* techniques [13] can be formalized in our framework. This family of techniques works for the entire class of Universal co-Büchi Word automata (UCA, for short), which subsumes LTL+P. However, in this part, we focus only on bounded synthesis applied on LTL+P formulas. Given an LTL+P formula ϕ, bounded synthesis algorithms build the Büchi automaton $\mathcal{A}_{\neg\phi}$ for the negation of ϕ (eq., the Universal co-Büchi automaton for ϕ) and *bound* the number of times Controller player is forced to visit the set of final states of the Büchi automaton $\mathcal{A}_{\neg\phi}$ (eq., the set of rejecting states of the Universal co-Büchi automaton for ϕ). Consider the safety automaton $\mathcal{A}_k$ defined as $\mathcal{A}_{\neg\phi} \times C_k$, where C_k is the safety automaton corresponding to a *counter* that increments if it visits a final state of $\mathcal{A}_{\neg\phi}$, retains its value otherwise, and whose *safe* states are all those states where the counter has value less than k. We define the set of safe states of $\mathcal{A}_k$ as the set of final states of C_k. We can formalize bounded synthesis in our framework by means of a safety reduction, that we call $[\![\cdot]\!]_{bs}$, defined as follows: for any LTL+P formula ϕ and for any $k \in \mathbb{N}$, we define $[\![\phi]\!]^k_{bs}$ to be any safety formula (not necessarily of LTL+P) such that $\mathcal{L}([\![\phi]\!]^k_{bs}) = \mathcal{L}(\mathcal{A}_k)$. Note that (i) $[\![\phi]\!]^k_{bs}$ is a safety formula (since $\mathcal{A}_k$ is a safety automaton); and (ii) $[\![\phi]\!]^k_{bs}$ implies ϕ (for any $k \in \mathbb{N}$); thus $[\![\cdot]\!]_{bs}$ is a safety reduction (Def. 19). This reduction is also *complete* with respect to Def. 20 (see the theorem below). From now, with $\mathsf{id} : \mathbb{N} \to \mathbb{N}$ we denote the *identity* function.

Theorem 6 *The safety reduction* $[\![\cdot]\!]_{bs}$ *is* id*-complete.*

Interestingly, the proof of Theorem 6 is exactly the completeness proof of the K-Liveness algorithm by Claessen and Sörensson [11], which is a simple but very efficient algorithm for model checking based on safety reductions.

By using the main theorem of our framework (Theorem 5), we can derive the completeness of bounded synthesis for LTL+P [13, 15]. We remark that bounded synthesis works for the full class of UCA and that this is the reason why here we obtain a smaller upper bound with respect to [13, 29].

Corollary 2 (Completeness of Bounded Synthesis) *Let* ϕ *be an* LTL+P *formula over the set of variables* $\Sigma = \mathcal{U}\cup\mathcal{C}$*. It holds that* ϕ *is realizable if and only if there exists* $k \leq 2^{|\mathcal{U}|} \cdot 2^{|\mathcal{C}|} \cdot 2^{2^{c \cdot n}}$ *such that* $[\![\phi]\!]^k_{bs}$ *is realizable.*

In this paper, we do not consider the $[\![\cdot]\!]_{bs}$ safety reduction, which corresponds to bounded synthesis and works with full LTL+P, because we do not know yet if there exists a fully symbolic translation from any $[\![\phi]\!]^k_{bs}$ to a safety automaton. Since one of our *desiderata* is to use *symbolic* techniques and since for $\mathsf{LTL}_{\mathsf{EBR}}$+P there exists a fully symbolic procedure for obtaining an equivalent deterministic automaton [39], we focus on fragments of LTL+P for which we can use

$\mathsf{LTL}_{\mathsf{EBR}}$+P for this task. As we have seen in Sect. 5, GR-EBR is one of these.

6.9 Instantiation to GR-EBR

In this part, we prove the (soundness and) completeness of the safety reduction for GR-EBR realizability described in Sect. 5. In order to do that, we use the framework described in Sect. 6.1. We call $[\![\cdot]\!]_{ebr}$ the safety reduction described in Sect. 5. Since the framework works with formulas rather than with automata, for all $\phi \in$ GR-EBR, we define $[\![\phi]\!]^k_{ebr}$ to be any *safety formula* such that $\mathcal{L}([\![\phi]\!]^k_{ebr}) = \mathcal{L}(\mathcal{A}^{\mathrm{k}}_{safe})$. Recall that with $\mathsf{id} : \mathbb{N} \to \mathbb{N}$ we denote the *identity* function.

Theorem 7 *$[\![\cdot]\!]_{ebr}$ is a* id*-complete safety reduction for* GR-EBR.

Proof We have to prove that, for all $\phi \in$ GR-EBR, for all Kripke structures M and for all $k \in \mathbb{N}$, it holds that:

$$M \models \mathsf{A}\phi \quad \Leftrightarrow \quad \exists k \leq \mathsf{id}(|M|) \cdot M \models \mathsf{A}[\![\phi]\!]^k_{ebr}$$

We prove separately the two directions. Consider first the *soundness* which corresponds to the right-to-left direction. Suppose that $M \models \mathsf{A}[\![\phi]\!]^k_{ebr}$. It holds that, for each initialized trace π of M, $L(\pi) \models [\![\phi]\!]^k_{ebr}$, where $L(\cdot)$ is the labeling function of M. Let π be an initialized trace of M. By definition of $[\![\cdot]\!]_{ebr}$, it holds that there exists a run τ induced by $L(\pi)$ such that: (i) τ is accepting in $\mathcal{A}_{ebr} \times \mathcal{A}_{degen}$, and (ii) τ is accepting in $\mathcal{A}^k_{\#^{\rightarrow}_{\alpha,\beta}}$. From the second point, we have that either:

- $\#^{\rightarrow}_{\alpha,\beta}$ make infinitely many *resets*. This means that there exists infinitely many positions in τ in which m^{tot}_{α} holds and, after at most k occurrences of m^{tot}_{α}, there is a m^{tot}_{β}. Therefore, in particular, there exists infinitely many positions in which m^{tot}_{β} holds, that is $\tau \models \mathsf{G}\,\mathsf{F} m^{tot}_{\beta}$.
- Or, the counter $\#^{\rightarrow}_{\alpha,\beta}$ stops to increment because, because it does not read any m^{tot}_{α}. This means that there exists *finitely* many positions in which m^{tot}_{α} holds, that is $\tau \models \mathsf{F}\,\mathsf{G}\neg m^{tot}_{\alpha}$.

Therefore, it holds that $\tau \models \mathsf{F}\,\mathsf{G}\neg m^{tot}_{\alpha} \vee \mathsf{G}\,\mathsf{F} m^{tot}_{\beta}$, that is $\tau \models \mathsf{G}\,\mathsf{F} m^{tot}_{\alpha} \rightarrow \mathsf{G}\,\mathsf{F} m^{tot}_{\beta}$. Finally, we have that τ is an accepting run of $\mathcal{A}_{ebr} \times \mathcal{A}_{degen}$ such that $\tau \models \mathsf{G}\,\mathsf{F} m^{tot}_{\alpha} \rightarrow \mathsf{G}\,\mathsf{F} m^{tot}_{\beta}$. Since by hypothesis $L(\pi)$ is induced by τ, by definition of $\mathcal{A}^{\mathsf{R}(1)}_{\mathrm{degen}}$, we have that $L(\pi) \in \mathcal{L}(\mathcal{A}_{ebr} \times \mathcal{A}^{\mathsf{R}(1)}_{\mathrm{degen}})$. By concatenating Lemma 2 and Lemma 3, we have that $L(\pi) \in \mathcal{L}(\phi)$, and therefore $\pi \models \phi$. It follows that $M \models \mathsf{A}\phi$.

We now prove *completeness*, which corresponds to the left-to-right direction. Suppose that $M \models \mathsf{A}\phi$, where $\phi \in$ GR-EBR. We prove this case by contradiction. Suppose therefore that for all $k \leq \mathsf{id}(|M|)$, $M \not\models \mathsf{A}[\![\phi]\!]^k_{ebr}$. This means that there exists an initialized trace π in M such that $L(\pi) \notin \mathcal{L}([\![\phi]\!]^k_{ebr})$, for all $k \leq \mathsf{id}(|M|)$. By definition of $[\![\cdot]\!]_{ebr}$, for $k = \mathsf{id}(|M|)$, we have that *for all* runs τ induced by $L(\pi)$ in $\mathcal{A}_{ebr} \times \mathcal{A}_{degen} \times \mathcal{A}^k_{\#^{\rightarrow}_{\alpha,\beta}}$, it holds that $\tau \not\models \mathsf{G}(\#^{\rightarrow}_{\alpha,\beta} \leq k)$. Let τ be one of these runs. There exists a position i in τ such that $\tau_i \models (\#^{\rightarrow}_{\alpha,\beta} = v)$, for some $v > k$. By definition of the counter $\#^{\rightarrow}_{\alpha,\beta}$, the run τ is such that:

$$\begin{aligned}\exists 0 < h_1 < h_2 < \cdots < h_v \cdot (\ & \tau_{h_1} \models m^{tot}_{\alpha} \\ & \wedge \tau_{h_2} \models m^{tot}_{\alpha} \wedge \ldots \tau_{h_v} \models m^{tot}_{\alpha} \wedge \\ & \forall h_1 \leq h \leq h_v \cdot (\tau_j \models \neg m^{tot}_{\beta}))\end{aligned}$$

Recall that τ is a run induced by $L(\pi)$. Since $v > k$, $k = \mathsf{id}(|M|)$ and M is a *finite-state* Kripke structure, the positions $h_1 \ldots h_v$ in π (attention: *not* in τ) cannot be all different. That is, there exists at least two indexes $s, e \in \mathbb{N}$ such that: (i) $1 \leq s < e \leq v$, (ii) $\pi_{h_s} = \pi_{h_e}$, and (iii) $\pi_{h_s} \models m^{tot}_{\alpha}$. Starting from π, we can build a *looping trace* π' that agrees with π in the prefix $\pi_{[0,h_e]}$ and then loops on the interval $\pi_{[h_s,h_e]}$. It holds that π' is an initialized trace of M and it induces a run τ' such that $\tau' \models \mathsf{G}\,\mathsf{F} m^{tot}_{\alpha} \wedge \mathsf{F}\,\mathsf{G}\neg m^{tot}_{\beta}$, that is $\tau' \not\models \mathsf{G}\,\mathsf{F} m^{tot}_{\alpha} \rightarrow \mathsf{G}\,\mathsf{F} m^{tot}_{\beta}$. Nevertheless, since $M \models \mathsf{A}\phi$, by Lemma 2 and Lemma 3, we have that $L(\pi') \in \mathcal{L}(\mathcal{A}_{ebr} \times \mathcal{A}^{\mathsf{R}(1)}_{\mathrm{degen}})$, and therefore this is a contradiction. This means that it has to hold that $L(\pi) \in \mathcal{L}([\![\phi]\!]^k_{ebr})$, that is $\pi \models [\![\phi]\!]^k_{ebr}$ for all the initialized traces π of M, and thus there exists a $k \leq \mathsf{id}(|M|)$ such that $M \models \mathsf{A}[\![\phi]\!]^k_{ebr}$. □

With Theorem 5, we derive the following corollary that proves the completeness of our procedure.

Corollary 3 *For any formula $\phi \in$ GR-EBR, it holds that: ϕ is realizable iff $\exists k \leq \mathsf{id}(2^{|\mathcal{U}|} \cdot 2^{|\mathcal{C}|} \cdot 2^{2^{c \cdot n}})$ such that $[\![\phi]\!]^k_{ebr}$ is realizable.*

7 Related work

GR(1) has been introduced in [7, 40]. It is known that GR(1) is a good candidate for writing specifications of real-world scenarios, with a relatively low complexity: the realizability problem can be solved with at most a quadratic number of symbolic steps in the size of the specification [7]. Moreover, its importance as a specification language is accentuated by the fact that the majority of the patterns that are most commonly used in industrial specifications [41] can be compiled into GR(1) specifications [42]. Nonetheless, GR(1) presents some restrictions that limit its use as a specification language: (i) safety assumptions/guarantees are either Boolean formulas or formulas of the form $\mathsf{G}\alpha$, where the only temporal operator admitted in α is the *next* operator X and no nesting of *next* operators is allowed; (ii) assumptions are syntactically constrained to be formulas *controlled* by

Environment, in the sense that the variables inside the *next* operators of the safety part of the assumptions must be *uncontrollable*. In GR-EBR we relax these syntactical restrictions of GR(1): for example, the safety assumptions and guarantees can be any arbitrary $\mathsf{LTL}_{\mathsf{EBR}}$+P formula, like, for instance, $\mathsf{G}(r \rightarrow \mathsf{F}^{[0,10]} g)$. For this reason, GR-EBR can be considered an extension not only of $\mathsf{LTL}_{\mathsf{EBR}}$+P, but also of GR(1).

On the algorithmic side, the (*strict*) realizability problem for GR(1) is solved in [7] by building a *symbolic* arena (called *game structure*) and by solving a fixpoint computation over it, that requires $\mathcal{O}(N^2)$ symbolic operations, where N is the size of the specification. In this paper, we follow a different approach based on a reduction to safety, that generates a sequence of deterministic safety automata over which the corresponding game can be solved with at most a *linear* number of symbolic steps.

The strict semantics for the realizability problem of GR(1) is supported by many tools, including SLUGS [43] and SPECTRA [44], the last one providing also advanced language features (like, for example, support for bounded counters and arithmetic operations) intended to help the writing of specifications. Interestingly, with the goal of refining the assumptions of an unrealizable GR(1) specification into a realizable one but still maintaining the assumptions as general as possible, also algorithms for computing minimal refinements of the assumptions have been proposed [45].

In [46], Morgenstern and Schneider identify a syntactical fragment of LTL, whose formulas correspond to deterministic Büchi automata. The fragment is defined in such a way that it corresponds to the temporal hierarchy defined in [27] (as a matter of fact, each formula of GR-EBR can be transformed into an equivalent one of that fragment by expanding bounded operators). Realizability is solved by exploiting known algorithms like subset construction and Miyano-Hayashi breakpoint construction for the determinization of the automata. On the contrary, the compilation of GR-EBR to automata is fully symbolic, which has been proved in [9] to be a key point for performance, compared to classical algorithms for determinization.

Bounded synthesis [13, 15] belongs to the class of *Safraless techniques* [14], and it consists in bounding the number of times Controller is forced to visit a rejecting state of a Universal co-Büchi automaton (UCW, for short) for the initial formula. This corresponds to a safety automaton, which can be either (i) made deterministic by a suitable generalization of the classical subset construction [29, 47], or (ii) encoded into a constraint system [13, 15] (e.g., SAT- or SMT-based) which bounds also the *size* of a candidate controller (this also allows one to tackle undecidable problems, for instance in the case of distributed or parametric synthesis). Both choices work for the whole class of UCW, and thus for full LTL. A significant drawback of such an approach is that the UCW, which can be exponentially larger than the initial specification, is explicitly represented. Moreover, in the first case, the algorithm for the determinization turns out to be quite complex, since each state of the resulting automaton is actually a *function*. This can also result into a very large state space, that can be tackled by exploiting either *antichains* [29] or BDDs [47]. In contrast, as we will see, we define a reduction tailored to GR-EBR formulas that allows us to exploit the $\mathsf{LTL}_{\mathsf{EBR}}$+P transformations introduced in [9] for a *fully symbolic* mapping of the initial formula directly into a sequence of symbolic safety automata. In particular, we never build any explicit-state automaton and we avoid the subsequent use of determinization algorithms.

ss

7.1 A short account of the tool BoSy

BoSy is a tool for reactive synthesis from LTL+P specifications based on bounded synthesis [48]. The main algorithm of BoSy takes a temporal formula ϕ in input and consists of the following steps: (i) it builds the Universal co-Büchi automaton (UCW) for ϕ; this automaton is built by executing one of the two tools LTL3BA [49] and SPOT [50], and it is *explicitly* represented; (ii) the automaton is optimized, e.g., by analyzing its strongly connected components [13]; (iii) the optimized automaton, along with the bound k on the visits to its rejecting states, is encoded into a constraint system (e.g., SAT, QBF, SMT) and solved by a corresponding backend. Among the different encodings, the one based on *Quantified Boolean Formulas* (QBF) appears to be the most efficient one in practice [15], and thus it is the default one and the one with which we compare our tool GRACE. Finally, BoSy starts two threads, one checking the realizability of the formula and the other checking the unrealizability. Since we will evaluate GRACE and BoSy only on realizable formulas (the only ones of interest in our context), in order to make fair the comparison with BoSy, we commented the part of the source code of BoSy that starts the thread for the unrealizability check.

8 Experimental evaluation

We implemented the algorithm described in Sect. 5 and summarized in Fig. 4 in a prototype tool called GRACE (which stands for *GR-ebr reAlizability ChEcker*). [2]

We chose SAFETYSYNTH [51] as a BDD-based backend for solving each safety game (Def. 15). SAFETYSYNTH implements the classical BDD-based backward fixpoint for finding a strategy for Controller in a safety game represented in AIGER format [52].

[2] https://es-static.fbk.eu/tools/grace/

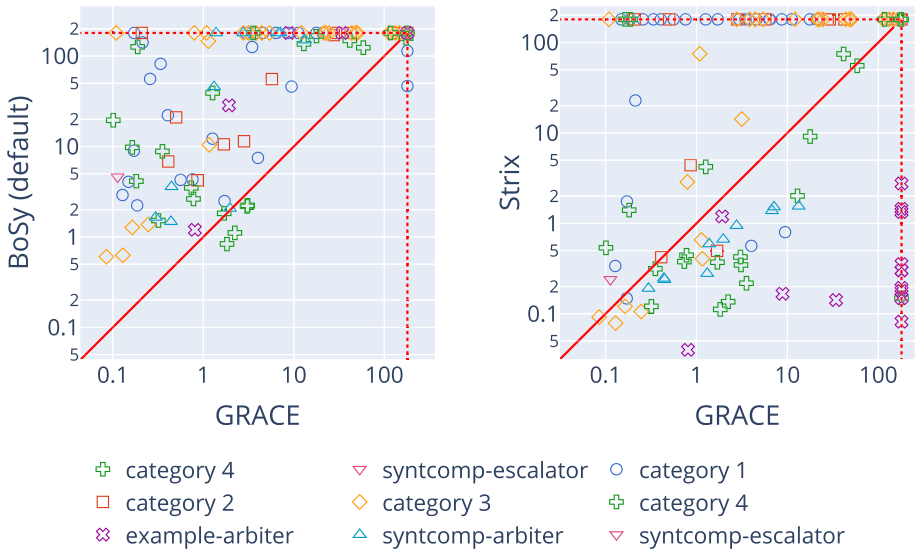


Fig. 6 GRACE compared to BOSY (on the left) and to STRIX (on the right). In both plots, the two axis show the solving time (in seconds) in logarithmic scale. The red dotted lines represent the timeout (180 s)

As competitor tools, we chose BOSY [15, 48, 53] and STRIX [54, 55]. STRIX is based on parity games and is the winner of SYNTCOMP 2018, 2019 and 2020, while BOSY implements the Bounded Synthesis approach: we refer the reader to Sect. 7 for a description of Bounded Synthesis and for a description of the BOSY tool. We set a timeout of 180 seconds. The experiments have been run on a 2.7 GHz AMD 16-cores machine with 64 GB of RAM.

We remark that a comparison with GR(1) synthesis tools is nontrivial. The majority of the tools for GR(1) only support the realizability of the *strict* implication (see for example [43]), not the standard one (which is our case). Therefore, although the latter can be reduced to the former [7], a non-trivial practical effort is required to write an algorithm for this translation.

8.1 Description of the benchmarks set

We considered benchmarks of two types: (i) artificial, and (ii) derived from the SYNTCOMP [51] benchmarks'set. Regarding the artifical benchmarks, we partitioned them in four categories, each containing 30 benchmarks scalable in their dimension N, for a total of 120 formulas. The categories are the following ones:

1. $\mathsf{G}(u_0 \rightarrow \mathsf{X}(u_1 \rightarrow \mathsf{X}(u_2 \rightarrow \cdots \rightarrow \mathsf{X}(u_N)\ldots))) \rightarrow \mathsf{G}(\bigwedge_{i=1}^{N}(u_i \leftrightarrow \mathsf{X}c_i))$
2. $(\mathsf{G}(u_0 \rightarrow \mathsf{X}(u_1 \rightarrow \mathsf{X}(u_2 \rightarrow \cdots \rightarrow \mathsf{X}(u_N)\ldots))) \wedge X^N\mathsf{G}u_N \wedge \mathsf{G}\,\mathsf{F}u_N) \rightarrow (\bigwedge_{i=1}^{N}(u_i \leftrightarrow \mathsf{X}^N c_i) \wedge \mathsf{G}\,\mathsf{F}c_N)$
3. $(\mathsf{G}(u_0) \wedge \mathsf{X}\,\mathsf{G}(u_1) \wedge \cdots \wedge \mathsf{X}^N\mathsf{G}(u_N) \wedge \bigwedge_{i=1}^{N}\mathsf{G}\,\mathsf{F}u_i) \rightarrow (\bigwedge_{i=1}^{N}\mathsf{G}(u_i \leftrightarrow c_i) \wedge \bigwedge_{i=1}^{N}\mathsf{G}\,\mathsf{F}c_i)$
4. $(\neg u_0 \wedge \mathsf{G}^{[0,N]}\neg u_0 \wedge \mathsf{X}^{N+1}\mathsf{G}u_0) \rightarrow (\bigwedge_{i=1}^{N}\mathsf{G}(u_0 \leftrightarrow \mathsf{X}c_i) \wedge \bigwedge_{i=1}^{N}\mathsf{G}\,\mathsf{F}(c_i \wedge u_0))$

The variables starting with u are uncontrollable, while those starting with c are controllable. All the benchmarks are realizable, and were specifically crafted to elicit potential criticalities of GRACE. The formulas in the first category consist of an implication between two $\mathsf{LTL}_{\mathsf{EBR}}$+P formulas. The second category extends the first one by adding to its assumptions (resp. the guarantees) the stabilization constraint $\mathsf{X}^N\mathsf{G}u_N$ (resp. $\mathsf{X}^N\mathsf{G}c_N$) and a *single* fairness $\mathsf{G}\,\mathsf{F}u_N$ (resp. $\mathsf{G}\,\mathsf{F}c_N$). In the third category, the formula corresponding to both the assumptions and the guarantees is such that half of it is safety and the other half is fairness. In particular, there are *multiple* fairnesses, as many as the dimension N. Finally, the benchmarks in the fourth category have been specifically designed in order to force the minimum k of the termination of GRACE to increase with their dimension.

Regarding the benchmarks derived from the SYNTCOMP benchmarks set, we included (i) *simple_arbiter_N* (for each $N \in \{2, 4, 6, 8, 10, 12\}$), *escalator_ bidirectional*, which belong to the SYNTCOMP benchmarks' set, and (ii) our example for an arbiter, with $N \in \{1, \ldots, 15\}$.

8.2 Discussion of the results

Figure 6 shows the comparison of the tools on the solving time of all the benchmarks. All times are in seconds and the scale is logarithmic. From Fig. 6 (left), we can see that almost all points are above the diagonal and, in particular, are located on the uppermost axis of the plot, which represents the timeout for the tool BOSY. This behavior involves formulas of both types (artificial and derived from SYNTCOMP), and of all four categories (of artificial benchmarks). In particular

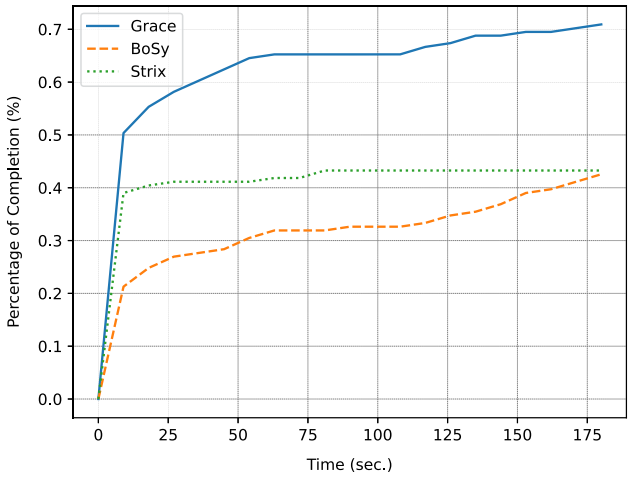


Fig. 7 Survival plot comparing the tools GRACE, BOSY and STRIX on the percentage of benchmarks solved in less than t seconds, for each $t \in \{0, \ldots, 180\}$

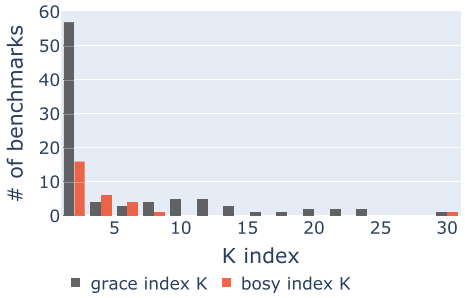


Fig. 8 GRACE versus BOSY on number of safety sub-problems

- on category 1, BOSY takes 0.5 sec. for $N = 4$ and it reaches the timeout with $N = 5$; for $N = 4$, the corresponding automaton (after optimization) has 40 states;
- on category 2 (and similarly for category 3), for $N = 5$ the solving time of BOSY is 49.2 sec., and the corresponding automaton (after optimization) has 48 states; with greater values of N, BOSY reaches the timeout;
- on category 4, the solving times of BOSY on $N = 13, 14$ are 19.4 and 136.3 sec., respectively, and the corresponding automata have 27 and 31 states, respectively.
- on `simple_arbiter_N`, BOSY takes 45.7 sec. for $N = 8$, and reaches the timeout for $N = 10$.

A more precise study of the complexity of BOSY shows that the majority of the time spent by it is due to the construction of the UCW (*Universal coBüchi Word automaton*) corresponding to the input formula, which is the task of the tools LTL3BA and SPOT. On the contrary, it is clear from Fig. 6 (left) that GRACE avoids this bad behavior, most likely due to the fact that the explicit state UCW is never built.

Let us now consider the tool STRIX (Fig. 6, right). It can be noted that, on the category `example-arbiter`, the solving times of STRIX are consistently better than the ones of GRACE. A careful study reveals that all those benchmarks are transformed to the equi-realizable formula `true` by the preprocessor of OWL [56] (a tool for ω-automata manipulation), which STRIX is based on.

In Fig. 7, we reported a survival plot showing, for each $t \in \{0, \ldots, 180\}$, the percentage of benchmarks solved by each of the tools in less than t seconds. An analysis of Fig. 7 shows that, although in Fig. 6 (right) there are some points below the diagonal (representing benchmarks in which STRIX performs better than GRACE), the majority of the points stands above the diagonal.

The plot in Fig. 8 shows, for each index k ranging from 1 to 31 (these correspond to the number of columns), on how many benchmarks (of both types) GRACE or BOSY terminate with index k (this corresponds to the height of a column). The benchmarks in category 4 and the ones of `simple_arbiter_N` force GRACE to terminate with increasing values of k. The plot in Fig. 8 points out that BOSY does not incur in this growth, except for one benchmark. Nevertheless, the solving times of GRACE are still better than the ones of BOSY. For example, for $N = 14$ of category 4, GRACE takes 20.4 sec., while BOSY takes 161 sec. This witnesses the fact that each safety sub-problem generated by GRACE is very simple to solve.

9 Conclusions and future work

In this paper, we proved the expressive completeness of $\mathsf{LTL}_{\mathsf{EBR}}$+P with respect to the safety fragment of LTL+P. We showed also that the removal of past operators from $\mathsf{LTL}_{\mathsf{EBR}}$+P results into a loss of expressive power.

With the objective of expressing properties beyond the safety fragments while retaining an efficient synthesis problem, we introduced the logic of GR-EBR, an extension of $\mathsf{LTL}_{\mathsf{EBR}}$+P [9] adding fairness conditions and assumes-guarantees formulas, and studied its realizability problem. We proposed a general framework to derive complete safety reductions in the context of realizability of (fragments of) LTL, and then we used it as the core of an algorithm for the realizability of GR-EBR formulas. The experimental evaluations showed good performance against other tools for (bounded) synthesis.

We aim at extending the work done in the following directions: (i) as far as we know, there are no safety synthesizers (like SAFETYSYNTH) that are able to exploit *incrementality*; since in our context, the only part of the automaton that changes between one iteration and the next one is the counter, some work may be saved; (ii) *stabilizing constraints* are successfully used in model checking, in particular by the K-Liveness algorithm [11], in order to shrink the search of the search space; we expect that realizability may also ben-

efit from them; (iii) since GR(1) is a very efficient fragment, it is important to investigate whether there is a compilation from GR-EBR to GR(1); (iv) we know that the realizability problem of GR-EBR is bounded below by EXPTIME (by EXPTIME-completeness of $\mathsf{LTL_{EBR}}$+P) and bounded above by EXPTIME [2] (by EXPTIME [2]-completeness of LTL+P): we aim at precisely characterizing its worst-case complexity; (v) last but not least, we aim at exploiting the proposed framework for more expressive logics, such as full LTL.

Funding Open access funding provided by Università degli Studi di Udine within the CRUI-CARE Agreement.

Appendix A Proofs

In this section of the Appendix, there are all the proofs that are missing from the main body of the paper.

Theorem 8 $[\![\mathsf{LTL_{EBR}}\text{+P}]\!] = [\![\mathsf{LTL}]\!] \cap \mathsf{SAFETY}$.

Proof We first prove that $[\![\mathsf{LTL_{EBR}}\text{+P}]\!] \subseteq [\![\mathsf{LTL}]\!] \cap \mathsf{SAFETY}$. Let $\phi \in [\![\mathsf{LTL_{EBR}}\text{+P}]\!]$. By Def. 5, $\phi \in \mathsf{LTL}$+P, and thus, since $[\![\mathsf{LTL}]\!] = [\![\mathsf{LTL}\text{+P}]\!]$, it holds that $\mathcal{L}(\phi) \in [\![\mathsf{LTL}]\!]$. Moreover, since $\mathsf{LTL_{EBR}}$+P contains only universal temporal operators, by Theorem 3.1 in [26], it is a *safety* fragment of LTL, and we have that $\mathcal{L}(\phi) \in \mathsf{SAFETY}$. Therefore, $\mathcal{L}(\phi) \in [\![\mathsf{LTL}]\!] \cap \mathsf{SAFETY}$.

We now prove that $[\![\mathsf{LTL}]\!] \cap \mathsf{SAFETY} \subseteq [\![\mathsf{LTL_{EBR}}\text{+P}]\!]$. Let ϕ be a formula such that $\mathcal{L}(\phi) \in [\![\mathsf{LTL}]\!] \cap \mathsf{SAFETY}$. By Theorem 1, $\mathcal{L}(\phi) \in [\![\mathsf{G}\alpha]\!]$. Now, $\mathsf{G}\alpha$ (for any $\alpha \in \mathsf{LTL}\text{+P}_\mathsf{P}$) is a formula that syntactically belongs to $\mathsf{LTL_{EBR}}$+P, that is $\mathsf{G}\alpha \in \mathsf{LTL_{EBR}}$+P, and thus $[\![\mathsf{G}\alpha]\!] \subseteq [\![\mathsf{LTL_{EBR}}\text{+P}]\!]$. It follows that $\mathcal{L}(\phi) \in [\![\mathsf{LTL_{EBR}}\text{+P}]\!]$. □

Lemma 4 $[\![\mathsf{LTL_{EBR}}]\!] = [\![\mathsf{Normal\text{-}LTL_{EBR}}]\!]$.

Proof Obviously $[\![\mathsf{Normal\text{-}LTL_{EBR}}]\!] \subseteq [\![\mathsf{LTL_{EBR}}]\!]$, since each formula ψ that belongs to $\mathsf{Normal\text{-}LTL_{EBR}}$ can be turned into an equivalent one $\psi' \in \mathsf{LTL_{EBR}}$ by expanding each bounded past operators into conjunctions/disjunctions of *yesterday* operators.

For proving $[\![\mathsf{LTL_{EBR}}]\!] \subseteq [\![\mathsf{Normal\text{-}LTL_{EBR}}]\!]$, it is sufficient to apply the transformations described in [39] for the translation of $\mathsf{LTL_{EBR}}$+P into normal form. In particular, since by definition ψ has no past temporal operators, the only past operators in ψ' are the ones introduced by the *pastification* step described in [39], which are all *bounded*, that is either Y or $\mathsf{S}^{[a,b]}$. □

Lemma 5 *Let $d \in \mathbb{N}$. For all $i \geq d$, for all $j \geq i + d$, and for all $k \geq j + d$, it holds that:*

$$\text{Property1} : \forall n' \geq 0 \cdot \exists n \geq 0 \cdot {}^{i,k}\sigma^{j}_{[n'-d,n']} = {}^{i,i}\sigma^{j}_{[n-d,n]}$$
$$\text{Property2} : \forall n \geq 0 \cdot \exists n' \geq 0 \cdot {}^{i,i}\sigma^{j}_{[n-d,n]} = {}^{i,k}\sigma^{j}_{[n'-d,n']}$$

Proof Take any value for i, j, and k such that: (i) $i \geq d$, (ii) $j \geq i + d$, (iii) $k \geq j + d$. Given any interval of length d of the state sequence ${}^{i,i}\sigma^{j}$, we show how to find an exact same one in ${}^{i,k}\sigma^{j}$, and viceversa.

The constraints above on the three indices ensure that both the state sequences ${}^{i,i}\sigma^{j}$ and ${}^{i,k}\sigma^{j}$ contain *only three* types of intervals of length at most d. Consider ${}^{i,k}\sigma^{j}$ (the case for ${}^{i,i}\sigma^{j}$ is specular). The three types are the following:

Type 1: $(\{p_1, p_2\})^n$ for some $0 \leq n \leq d$;
Type 2: $(\{p_1, p_2\})^n \cdot (\{p_1\}) \cdot (\{p_1, p_2\})^{d-n-1}$, for some $0 \leq n < d$;
Type 3: $(\{p_1, p_2\})^n \cdot (\{p_2\}) \cdot (\{p_1, p_2\})^{d-n-1}$, for some $0 \leq n < d$;

The situation is depicted in Fig. 2. Given any interval of any of the three types above, we show below how to find the very same interval in ${}^{i,i}\sigma^{j}$ (Fig. 2 tries to show visually this correspondence):

- each interval of ${}^{i,k}\sigma^{j}$ of type $(\{p_1, p_2\})^n$ is equal to ${}^{i,i}\sigma^{j}_{[0,n]}$;
- each interval of ${}^{i,k}\sigma^{j}$ of type $(\{p_1, p_2\})^n \cdot (\{p_1\}) \cdot (\{p_1, p_2\})^{d-n-1}$ is equal to ${}^{i,i}\sigma^{j}_{[i-n,i+d-n-1]}$.
- each interval of ${}^{i,k}\sigma^{j}$ of type $(\{p_1, p_2\})^n \cdot (\{p_2\}) \cdot (\{p_1, p_2\})^{d-n-1}$ is equal to ${}^{i,i}\sigma^{j}_{[j-n,j+d-n-1]}$;

This proves *Property 1*.

Similarly, the correspondence between intervals of ${}^{i,i}\sigma^{j}$ and intervals of ${}^{i,k}\sigma^{j}$ is the following:

- each interval of ${}^{i,i}\sigma^{j}$ of type $(\{p_1, p_2\})^n$ is equal to ${}^{i,k}\sigma^{j}_{[0,n]}$;
- each interval of ${}^{i,i}\sigma^{j}$ of type $(\{p_1, p_2\})^n \cdot (\{p_1\}) \cdot (\{p_1, p_2\})^{d-n-1}$ is equal to ${}^{i,k}\sigma^{j}_{[i-n,i+d-n-1]}$.
- each interval of ${}^{i,i}\sigma^{j}$ of type $(\{p_1, p_2\})^n \cdot (\{p_2\}) \cdot (\{p_1, p_2\})^{d-n-1}$ is equal to ${}^{i,k}\sigma^{j}_{[j-n,j+d-n-1]}$;

This proves *Property 2*. □

Lemma 6 *Let $\alpha, \beta \in \mathsf{LTL}\text{+P}_{\mathsf{BP}}$, and let $d = \max\{D(\alpha), D(\beta)\}$ be the maximum between the temporal depths of α and β. It*

holds that ${}^{i,i}\sigma^j \models \alpha \mathsf{R} \beta$ *iff* ${}^{i,k}\sigma^j \models \alpha \mathsf{R} \beta$*, for all* $i \geq d$*, for all* $j \geq i + d$*, and for all* $k \geq j + d$.

Proof Take any value for i, j, and k such that: (i) $i \geq d$, (ii) $j \geq i + d$, (iii) $k \geq j + d$. We first prove the left-to-right direction. Suppose that ${}^{i,i}\sigma^j \models \alpha \mathsf{R} \beta$. We divide in cases:

1. Suppose that ${}^{i,i}\sigma^j, n \models \beta$ for all $n \geq 0$. Since $\beta \in \mathsf{LTL{+}P_{BP}}$ and $D(\beta) \leq d$, it holds that ${}^{i,i}\sigma^j_{[n-d,n]} \models \beta$, for all $n \geq 0$. Suppose by contradiction that there exists some $n' \geq 0$ such that ${}^{i,k}\sigma^j_{[n'-d,n']} \models \neg\beta$. By *Property 1* of Lemma 5, this means that there exists some $n'' \geq 0$ such that ${}^{i,i}\sigma^j_{[n''-d,n'']} \models \neg\beta$. But this is a contradiction. Thus, it holds that ${}^{i,k}\sigma^j_{[n'-d,n']} \models \beta$ for all $n' \geq 0$, that is, for all $n' \geq 0$, and thus ${}^{i,k}\sigma^j \models \alpha \mathsf{R} \beta$.
2. Suppose that $\exists n \geq 0 \cdot ({}^{i,i}\sigma^j, n \models \alpha \wedge \forall 0 \leq m \leq n \cdot {}^{i,i}\sigma^j, m \models \beta)$. We divide again in cases:
   (a) Suppose that $n < k$. Then ${}^{i,i}\sigma^j_{[0,n]} = {}^{i,k}\sigma^j_{[0,n]}$. Clearly, it holds that ${}^{i,k}\sigma^j, n \models \alpha$ and ${}^{i,k}\sigma^j, m \models \beta$ for all $0 \leq m \leq n$. Therefore ${}^{i,k}\sigma^j \models \alpha \mathsf{R} \beta$.
   (b) Suppose that $n \geq k$. In particular, it holds that ${}^{i,i}\sigma^j_{[n-d,n]} \models \alpha \wedge \beta$. We use a *contraction argument* for proving that in this case there exists a smaller index at which the *release* satisfies its existential part (i.e., the formula α). Consider the time point $i - 1$. It holds that ${}^{i,i}\sigma^j_{[i-1-d,i-1]} = {}^{i,i}\sigma^j_{[n-d,n]}$ and thus, since ${}^{i,i}\sigma^j_{[n-d,n]} \models \alpha \wedge \beta$ and $\alpha, \beta \in \mathsf{LTL{+}P_{BP}}$, we have that ${}^{i,i}\sigma^j_{[i-1-d,i-1]} \models \alpha \wedge \beta$. Moreover, ${}^{i,i}\sigma^j_{[0,i-1]}$ is a prefix of ${}^{i,i}\sigma^j_{[0,n]}$, and thus, given that ${}^{i,i}\sigma^j_{[p-d,p]} \models \beta$ for all $0 \leq p \leq n$, it holds that ${}^{i,i}\sigma^j_{[p-d,p]} \models \beta$ for all $0 \leq p \leq i - 1$. From this, it follows that ${}^{i,i}\sigma^j, i - 1 \models \alpha$ and ${}^{i,i}\sigma^j, m \models \beta$ for all $0 \leq m \leq i - 1$. Since $i - 1 < k$, by Item 22a, it holds that ${}^{i,k}\sigma^j \models \alpha \mathsf{R} \beta$.

We now prove the right-to-left direction. Suppose that ${}^{i,k}\sigma^j \models \alpha \mathsf{R} \beta$. We divide in cases:

1. Suppose that ${}^{i,k}\sigma^j, n \models \beta$. This case is specular to Item 1.
2. Suppose that $\exists n \geq 0 \cdot ({}^{i,k}\sigma^j, n \models \alpha \wedge \forall 0 \leq m \leq n \cdot {}^{i,k}\sigma^j, m \models \beta)$. Since $\alpha, \beta \in \mathsf{LTL{+}P_{BP}}$ and $D(\alpha), D(\beta) \leq d$, it holds that $\exists n \geq 0 \cdot ({}^{i,k}\sigma^j_{[n-d,n]} \models \alpha \wedge \forall 0 \leq m \leq n \cdot {}^{i,k}\sigma^j_{[m-d,m]} \models \beta)$. We divide again in cases:
   (a) If $n < k$, then ${}^{i,k}\sigma^j_{[0,n]} = {}^{i,i}\sigma^j_{[0,n]}$ and thus ${}^{i,i}\sigma^j, n \models \alpha$ and ${}^{i,i}\sigma^j, m \models \beta$ for all $0 \leq m \leq n$, that is ${}^{i,i}\sigma^j \models \alpha \mathsf{R} \beta$.
   (b) If $k \leq n \leq k+d$, then ${}^{i,k}\sigma^j_{[n-d,n]} = {}^{i,k}\sigma^j_{[n-k-i-d,n-k-i]}$ (we used again a contraction argument). Since by hypothesis ${}^{i,k}\sigma^j_{[n-d,n]} \models \alpha$, it holds also that ${}^{i,k}\sigma^j_{[n-k-i-d,n-k-i]} \models \alpha$. Moreover, ${}^{i,k}\sigma^j_{[0,n-k-i]}$ is a prefix of ${}^{i,k}\sigma^j_{[0,n]}$, and thus, since by hypothesis ${}^{i,k}\sigma^j_{[p-d,p]} \models \beta$ for all $0 \leq p \leq n$, it also holds that ${}^{i,k}\sigma^j_{[p-d,p]} \models \beta$ for all $0 \leq p \leq n-k-i$. Therefore ${}^{i,k}\sigma^j_{[n-k-i-d,n-k-i]} \models \alpha$ and ${}^{i,k}\sigma^j_{[m-d,m]} \models \beta$ for all $0 \leq m \leq n-k-i$. Since $l+n-i < k$, by Item 22a, it holds that ${}^{i,i}\sigma^j \models \alpha \mathsf{R} \beta$.
   (c) Otherwise $n > k + d$. We have that ${}^{i,k}\sigma^j_{[n-d,n]} = {}^{i,k}\sigma^j_{[i-1,i-1-d]}$ (also in this case we used a contraction argument). Since by hypothesis ${}^{i,k}\sigma^j_{[n-d,n]} \models \alpha$, it also hold that ${}^{i,k}\sigma^j_{[i-1,i-1-d]} \models \alpha$. Moreover ${}^{i,k}\sigma^j_{[0,i-1]}$ is a prefix of ${}^{i,k}\sigma^j_{[0,n]}$ and thus, since by hypothesis ${}^{i,k}\sigma^j_{[p-d,p]} \models \beta$ for all $0 \leq p \leq n$, it also holds that ${}^{i,k}\sigma^j_{[p-d,p]} \models \beta$ for all $0 \leq p \leq i - 1$. Therefore ${}^{i,k}\sigma^j, i - 1 \models \alpha$ and ${}^{i,k}\sigma^j, m \models \beta$ for all $0 \leq m \leq i - 1$. Since $i - 1 < k$, by Item 22a, it holds that ${}^{i,i}\sigma^j \models \alpha \mathsf{R} \beta$.

□

Lemma 7 *Let* $\psi \in \mathsf{Normal\text{-}LTL_{EBR}}$ *. It holds that* ${}^{i,i}\sigma^j \models \psi$ *iff* ${}^{i,k}\sigma^j \models \psi$*, for all* $i \geq m_\psi + d_\psi$*, for all* $j \geq i + d_\psi$*, and for all* $k \geq j + d_\psi$.

Proof Take any value for i, j, and k such that: (i) $i \geq m_\psi + d_\psi$, (ii) $j \geq i + d_\psi$, (iii) $k \geq j + d_\psi$. We proceed by induction on the structure of the formula ψ.

For the base case, we consider three cases: (i) formulas in $\mathsf{LTL{+}P_{BP}}$, that is such that all its temporal operators refer to the past and are bounded; (ii) formulas of type $\mathsf{G}\alpha$, where $\alpha \in \mathsf{LTL{+}P_{BP}}$; (iii) formulas of type $\alpha \mathsf{R} \beta$, where $\alpha, \beta \in \mathsf{LTL{+}P_{BP}}$;

We consider the case of a formula $\alpha \in \mathsf{LTL{+}P_{BP}}$, and suppose that ${}^{i,i}\sigma^j \models \alpha$. By definition of ${}^{i,i}\sigma^j$ and ${}^{i,k}\sigma^j$, it always holds that ${}^{i,i}\sigma^j_0 = {}^{i,k}\sigma^j_0$. Since $\alpha \in \mathsf{LTL{+}P_{BP}}$ refers only to the current state or to the past, it follows that ${}^{i,i}\sigma^j \models \alpha$ if and only if ${}^{i,k}\sigma^j \models \alpha$.

Consider now the case for $\alpha \mathsf{R} \beta$, where $\alpha, \beta \in \mathsf{LTL{+}P_{BP}}$. Since $m_{\alpha \mathsf{R} \beta} = 0$ (i.e., the are no *next* operators in this formula), we can apply Lemma 6, having that ${}^{i,i}\sigma^j \models \alpha \mathsf{R} \beta$ if and only if ${}^{i,k}\sigma^j \models \alpha \mathsf{R} \beta$. Since $\mathsf{G}\alpha = \bot \mathsf{R} \alpha$, this proves also the case for the *globally* operator.

For the inductive step, since by hypothesis ψ belongs to the normal form of $\mathsf{LTL_{EBR}}$, it suffices to consider only the case for the *next operator*, *conjunctions* and *disjunctions*.

Consider first the case for the *next* operator, and suppose that ${}^{i,i}\sigma^j \models \mathsf{X}\psi'$. For any indices k, i and j such that $i \geq m_{\mathsf{X}\psi'} + d_{\mathsf{X}\psi'}$, $j \geq i + d_{\mathsf{X}\psi'}$ and $k \geq j + d_{\mathsf{X}\psi'}$, we want to prove that ${}^{i,k}\sigma^j \models \mathsf{X}\psi'$. By definition of the next operator, it holds that ${}^{i,i}\sigma^j, 1 \models \psi'$. Now, let τ be the state sequence obtained

from ${}^{i,i}\sigma^{j}$ by discarding its initial state, that is $\tau := {}^{i,i}\sigma^{j}_{[1,\infty)}$. Obviously, $\tau \models \psi'$. We observe that τ is equal to the state sequence ${}^{i-1,i-1}\sigma^{j-1}$. Since the maximum number $m_{\psi'}$ of nested next operators in ψ' is $m_{\mathsf{X}\psi'} - 1$ (while $\alpha_{\psi'}$ remains the same), we can apply the inductive hypothesis on ψ', having that ${}^{i-1,k-1}\sigma^{j-1} \models \psi'$.

By definition of τ, it follows that ${}^{i,k}\sigma^{j} \models \mathsf{X}\psi'$.

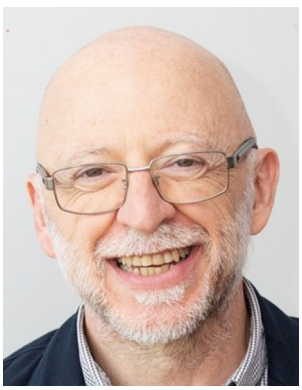
We consider now the case for conjunctions, and suppose that ${}^{i,i}\sigma^{j} \models \psi_1 \wedge \psi_2$, for generic indices k, i and j such that $i \geq m_{\psi_1 \wedge \psi_2} + d_{\psi_1 \wedge \psi_2}$, $j \geq i + d_{\psi_1 \wedge \psi_2}$, and $k \geq j + d_{\psi_1 \wedge \psi_2}$.

It holds that ${}^{i,i}\sigma^{j} \models \psi_1$ and ${}^{i,i}\sigma^{j} \models \psi_2$. Moreover, $m_{\psi_1} \leq m_{\psi_1 \wedge \psi_2}$ and $m_{\psi_2} \leq m_{\psi_1 \wedge \psi_2}$. Similarly, $d_{\psi_1} \leq d_{\psi_1 \wedge \psi_2}$ and $d_{\psi_2} \leq d_{\psi_1 \wedge \psi_2}$. This means that we can apply the inductive hypothesis both on ψ_1 and ψ_2 on the *current* indices k, i and j. By inductive hypothesis, we have that ${}^{i,k}\sigma^{j} \models \psi_1$ and ${}^{i,k}\sigma^{j} \models \psi_2$. It follows that ${}^{i,k}\sigma^{j} \models \psi_1 \wedge \psi_2$. The case for $\psi_1 \vee \psi_2$ is specular. □

References

1. Cimatti, A., Geatti, L., Gigante, N., Montanari, A., Tonetta, S.: Fairness, assumptions, and guarantees for extended bounded response LTL+P synthesis. In: Calinescu, R., Pasareanu, C.S. (eds.) Software Engineering and Formal Methods—19th International Conference, SEFM 2021, Virtual Event, December 6–10, 2021, Proceedings. Lecture Notes in Computer Science, vol. 13085, pp. 351–371. Springer (2021). https://doi.org/10.1007/978-3-030-92124-8_20
2. Cimatti, A., Geatti, L., Gigante, N., Montanari, A., Tonetta, S.: Expressiveness of Extended Bounded Response LTL, vol. 346, pp. 152–165 (2021). https://doi.org/10.4204/EPTCS.346.10
3. Church, A.: Logic, arithmetic, and automata. In: Proceedings of the International Congress of Mathematicians, vol. 1962, pp. 23–35 (1962)
4. Buchi, J.R., Landweber, L.H.: Solving sequential conditions by finite-state strategies. In: The Collected Works of J. Richard Büchi, pp. 525–541. Springer (1990). https://doi.org/10.1016/S0019-9958(66)80013-X
5. Pnueli, A., Rosner, R.: On the synthesis of an asynchronous reactive module. In: International Colloquium on Automata, Languages, and Programming (ICALP), pp. 652–671. Springer (1989). https://doi.org/10.1016/0022-0000(86)90026-7
6. Rosner, R.: Modular synthesis of reactive systems. PhD thesis, PhD thesis, Weizmann Institute of Science (1992)
7. Bloem, R., Jobstmann, B., Piterman, N., Pnueli, A., Saar, 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. Zhu, S., Tabajara, L.M., Li, J., Pu, G., Vardi, M.Y.: A Symbolic Approach to Safety LTL Synthesis. In: Strichman, O., Tzoref-Brill, R. (eds.) Proceedings of the 13th International Haifa Verification Conference. Lecture Notes in Computer Science, vol. 10629, pp. 147–162. Springer (2017). https://doi.org/10.1007/978-3-319-70389-3_10
9. Cimatti, A., Geatti, L., Gigante, N., Montanari, A., Tonetta, S.: Extended bounded response LTL: a new safety fragment for efficient reactive synthesis. In: Formal Methods in System Design, pp. 1–49. Springer (2021)
10. Bloem, R., Könighofer, R., Seidl, M.: Sat-based synthesis methods for safety specs. In: International Conference on Verification, Model Checking, and Abstract Interpretation, pp. 1–20. Springer (2014)
11. Claessen, K., Sörensson, N.: A liveness checking algorithm that counts. In: 2012 Formal Methods in Computer-Aided Design (FMCAD), pp. 52–59. IEEE (2012)
12. Biere, A., Artho, C., Schuppan, V.: Liveness checking as safety checking. Electron. Notes Theor. Comput. Sci. **66**(2), 160–177 (2002). https://doi.org/10.1016/S1571-0661(04)80410-9
13. Finkbeiner, B., Schewe, S.: Bounded synthesis. Int. J. Softw. Tools Technol. Transfer **15**(5–6), 519–539 (2013). https://doi.org/10.1007/s10009-012-0228-z
14. Kupferman, O., Vardi, M.Y.: Safraless decision procedures. In: 46th Annual Symposium on Foundations of Computer Science (FOCS), pp. 531–540. IEEE (2005). https://doi.org/10.1109/SFCS.2005.66
15. Faymonville, P., Finkbeiner, B., Rabe, M.N., Tentrup, L.: Encodings of bounded synthesis. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS), pp. 354–370. Springer (2017)
16. Lichtenstein, O., Pnueli, A., Zuck, L.: The glory of the past. In: Workshop on Logic of Programs, pp. 196–218. Springer (1985). https://doi.org/10.1007/3-540-15648-8_16
17. Gabbay, D.M., Pnueli, A., Shelah, S., Stavi, J.: On the temporal analysis of fairness. In: Abrahams, P.W., Lipton, R.J., Bourne, S.R. (eds.) Conference Record of the Seventh Annual ACM Symposium on Principles of Programming Languages, Las Vegas, Nevada, USA, January 1980, pp. 163–173. ACM Press (1980). https://doi.org/10.1145/567446.567462
18. Markey, N.: Temporal logic with past is exponentially more succinct (2003)
19. Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to automata theory, languages, and computation. ACM SIGACT News **32**(1), 60–65 (2001). https://doi.org/10.1145/568438.568455
20. Büchi, J.R.: Weak second-order arithmetic and finite automata. Math. Log. Q. **6**(1–6), 66–92 (1960). https://doi.org/10.1002/malq.19600060105
21. BUCHI, J.: On a decision method in restricted second-order arithmetics. In: Proceedings of International Congress on Logic, Methodology and Philosophy of Science (1960). Stanford Univ. Press
22. Kupferman, O., Vardi, M.Y.: Model checking of safety properties. Formal Methods Syst. Des. **19**(3), 291–314 (2001). https://doi.org/10.1023/A:1011254632723
23. McNaughton, R., Papert, S.A.: Counter-Free Automata (MIT Research Monograph No. 65). The MIT Press (1971)
24. Thomas, W.: Safety-and liveness-properties in propositional temporal logic: characterizations and decidability. Banach Center Publ. **1**(21), 403–417 (1988). https://doi.org/10.4064/-21-1-403-417
25. Chang, E.Y., Manna, Z., Pnueli, A.: Characterization of temporal property classes. In: Kuich, W. (ed.) Proceedings of the 19th International Colloquium on Automata, Languages and Programming. Lecture Notes in Computer Science, vol. 623, pp. 474–486. Springer (1992). https://doi.org/10.1007/3-540-55719-9_97
26. Sistla, A.P.: Safety, liveness and fairness in temporal logic. Formal Aspects Comput. **6**(5), 495–511 (1994). https://doi.org/10.1007/BF01211865
27. Manna, Z., Pnueli, A.: A hierarchy of temporal properties (invited paper, 1989). In: Proceedings of the 9th Annual ACM Symposium on Principles of Distributed Computing, pp. 377–410 (1990). https://doi.org/10.1145/93385.93442
28. Vardi, M.Y., Wolper, P.: Reasoning about infinite computations. Inf. Comput. **115**(1), 1–37 (1994). https://doi.org/10.1006/inco.1994.1092
29. Filiot, E., Jin, N., Raskin, J.-F.: An antichain algorithm for LTL realizability. In: International Conference on Computer Aided Verification (CAV), pp. 263–277. Springer (2009). https://doi.org/10.1007/978-3-540-78800-3_6

30. Safra, S.: On the complexity of omega-automata. In: Proceedings of 29th IEEE Symposium on Foundation of Computer Science, pp. 319–327 (1988)
31. Casares, A., Colcombet, T., Fijalkow, N.: Optimal transformations of games and automata using muller conditions. In: Bansal, N., Merelli, E., Worrell, J. (eds.) 48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12–16, 2021, Glasgow, Scotland (Virtual Conference). LIPIcs, vol. 198, pp. 123–112314. Schloss Dagstuhl-Leibniz-Zentrum für Informatik (2021). https://doi.org/10.4230/LIPIcs.ICALP.2021.123
32. de Alfaro, L., Henzinger, T.A.: Concurrent omega-regular games. In: Proceedings Fifteenth Annual IEEE Symposium on Logic in Computer Science (Cat. No. 99CB36332), pp. 141–154. IEEE (2000)
33. De Alfaro, L., Henzinger, T.A., Kupferman, O.: Concurrent reachability games. Theoret. Comput. Sci. **386**(3), 188–217 (2007). https://doi.org/10.1016/j.tcs.2007.07.008
34. Jacobs, S., Bloem, R., Brenguier, R., Ehlers, R., Hell, T., Könighofer, R., Pérez, G.A., Raskin, J.-F., Ryzhyk, L., Sankur, O., et al.: The first reactive synthesis competition (syntcomp 2014). Int. J. Softw. Tools Technol. Transfer **19**(3), 367–390 (2017). https://doi.org/10.1016/j.entcs.2013.07.009
35. Maler, O., Nickovic, D., Pnueli, A.: On synthesizing controllers from bounded-response properties. In: International Conference on Computer Aided Verification, pp. 95–107. Springer (2007). https://doi.org/10.1023/A:1008734703554
36. McMillan, K.L.: The smv Language, pp. 1–49. Cadence Berkeley Labs, London (1999)
37. Bloem, R., Chatterjee, K., Greimel, K., Henzinger, T.A., Jobstmann, B.: Robustness in the presence of liveness. In: International Conference on Computer Aided Verification (CAV), pp. 410–424. Springer (2010). https://doi.org/10.1016/S0304-3975(98)00009-7
38. Ehlers, R.: ACTL ∩ LTL synthesis. In: International Conference on Computer Aided Verification (CAV), pp. 39–54. Springer (2012). https://doi.org/10.1007/11609773_24
39. Cimatti, A., Geatti, L., Gigante, N., Montanari, A., Tonetta, S.: Reactive synthesis from extended bounded response LTL specifications. In: 2020 Formal Methods in Computer Aided Design, FMCAD 2020, Haifa, Israel, September 21–24, 2020, pp. 83–92. IEEE (2020). https://doi.org/10.34727/2020/isbn.978-3-85448-042-6_15
40. Piterman, N., Pnueli, A., Sa'ar, Y.: Synthesis of reactive (1) designs. In: International Workshop on Verification, Model Checking, and Abstract Interpretation, pp. 364–380. Springer (2006)
41. Dwyer, M.B., Avrunin, G.S., Corbett, J.C.: Patterns in property specifications for finite-state verification. In: Proceedings of the 21st International Conference on Software Engineering, pp. 411–420 (1999)
42. Maoz, S., Ringert, J.O.: Gr (1) synthesis for ltl specification patterns. In: Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, pp. 96–106 (2015)
43. Ehlers, R., Raman, V.: Slugs: Extensible GR(1) synthesis. In: International Conference on Computer Aided Verification, pp. 333–339. Springer (2016)
44. Maoz, S., Ringert, J.O.: Spectra: a specification language for reactive systems. Softw. Syst. Model. **20**(5), 1553–1586 (2021)
45. Cavezza, D.G., Alrajeh, D., György, A.: Minimal assumptions refinement for realizable specifications. In: Proceedings of the 8th International Conference on Formal Methods in Software Engineering, pp. 66–76 (2020)
46. Morgenstern, A., Schneider, K.: A LTL fragment for GR(1)-synthesis. In: Reich, J., Finkbeiner, B. (eds.) Proceedings International Workshop on Interactions, Games and Protocols, iWIGP 2011, Saarbrücken, Germany, 27th March 2011. EPTCS, vol. 50, pp. 33–45 (2011). https://doi.org/10.4204/EPTCS.50.3
47. Ehlers, R.: Symbolic bounded synthesis. In: International Conference on Computer Aided Verification (CAV), pp. 365–379. Springer (2010). https://doi.org/10.1007/978-3-540-75596-8_33
48. Faymonville, P., Finkbeiner, B., Tentrup, L.: Bosy: An experimentation framework for bounded synthesis. In: International Conference on Computer Aided Verification (CAV), pp. 325–332. Springer (2017). https://doi.org/10.1007/978-3-642-02777-2_24
49. Babiak, T., Křetínskỳ, M., Řehák, V., Strejček, J.: Ltl to büchi automata translation: Fast and more deterministic. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS), pp. 95–109. Springer (2012)
50. Duret-Lutz, A., Lewkowicz, A., Fauchille, A., Michaud, T., Renault, E., Xu, L.: Spot 2.0—a framework for LTL and ω-automata manipulation. In: International Symposium on Automated Technology for Verification and Analysis (ATVA), pp. 122–129. Springer (2016). https://doi.org/10.1007/3-540-60915-6_6
51. Jacobs, S., Bloem, R.: The 5th reactive synthesis competition (SYNTCOMP 2018)
52. Biere, A., Heljanko, K., Wieringa, S.: Aiger 1.9 and beyond (2011). Available at fmv. jku.at/hwmcc11/beyond1. pdf
53. Finkbeiner, B., Hahn, C., Lukert, P., Stenger, M., Tentrup, L.: Synthesizing reactive systems from hyperproperties. In: International Conference on Computer Aided Verification (CAV), pp. 289–306. Springer (2018). https://doi.org/10.1007/978-3-540-78800-3_24
54. Luttenberger, M., Meyer, P.J., Sickert, S.: Practical synthesis of reactive systems from ltl specifications via parity games. Acta Informatica **57**(1), 3–36 (2020)
55. Meyer, P.J., Sickert, S., Luttenberger, M.: Strix: explicit reactive synthesis strikes back! In: International Conference on Computer Aided Verification, pp. 578–586. Springer (2018). https://doi.org/10.1007/978-3-319-41540-6_17
56. Kretínský, J., Meggendorfer, T., Sickert, S.: Owl: a library for ω-words, automata, and LTL. In: Lahiri, S.K., Wang, C. (eds.) Automated Technology for Verification and Analysis—16th International Symposium, ATVA 2018, Los Angeles, CA, USA, October 7–10, 2018, Proceedings. Lecture Notes in Computer Science, vol. 11138, pp. 543–550. Springer (2018). https://doi.org/10.1007/978-3-030-01090-4_34



Alessandro Cimatti is the director of the Center for Digital Industry at Fondazione Bruno Kessler, Trento, Italy. His research interests concern formal verification of industrial critical systems, decision procedures and their applications, safety analysis, diagnosis and diagnosability, planning and runtime verification. Cimatti is the author of more than 250 papers in the fields of formal methods and artificial intelligence. For his fundamental works on Bounded Model Checking and on Satisfiability Modulo Theories, Cimatti has received the TACAS 2014 Most Influential Paper award, the ETAPS 2017 Test of Time award and the CAV Award in 2018 and 2021.

Luca Geatti received his Ph.D. at the University of Udine and Fondazione Bruno Kessler (FBK) in 2022. He spent 1 year at the Free University of Bozen/Bolzano, and he is now a Research Associate in Computer Science at the University of Udine. His research interests span automata theory, temporal logics, automatic synthesis, reactive synthesis, and formal verification.

Nicola Gigante received his Ph.D. at the University of Udine in 2019. After a 2-year postdoc period in Udine, he is now a Researcher at the Free University of Bozen-Bolzano. His research interests include temporal reasoning in formal verification and artificial intelligence, including temporal logic, temporal planning, and reactive synthesis from temporal specifications.

Angelo Montanari is full professor of computer science at the University of Udine, Italy, where he chairs the Data Science and Automatic Verification Laboratory. He received the Ph.D. in logic and computer science from the University of Amsterdam, The Netherlands, in 1996. He has published over 300 contributions in international journals, conferences, and handbook chapters. His main research interests include formal methods, artificial intelligence, and data science.

Stefano Tonetta is head of the Embedded System Unit at Fondazione Bruno Kessler (FBK). His research interests span from formal verification and automated reasoning to runtime verification, automated synthesis and contract-based design. Tonetta is the author of more than 150 papers and received the 2010 Microsoft Research SEIF Award, the 2010 FBK Stringa Award, the 2012 FMCAD Best Paper Award and the 2021 SEFM Best Paper Award.