Acta Informatica (2017) 54:655–692
DOI 10.1007/s00236-016-0280-3

ORIGINAL ARTICLE

Safraless LTL synthesis considering maximal realizability

Takashi Tomita[1,2] · Atsushi Ueno[2] · Masaya Shimakawa[2,3] · Shigeki Hagihara[2,3] · Naoki Yonezaki[2,4,5]

Received: 16 March 2015 / Accepted: 20 September 2016 / Published online: 7 October 2016

Abstract Linear temporal logic (LTL) synthesis is a formal method for automatically composing a reactive system that realizes a given behavioral specification described in LTL if the specification is realizable. Even if the whole specification is unrealizable, it is preferable to synthesize a best-effort reactive system. That is, a system that maximally realizes its partial specifications. Therefore, we categorized specifications into must specifications (which should never be violated) and desirable specifications (the violation of which may be unavoidable). In this paper, we propose a method for synthesizing a reactive system that realizes all must specifications and strongly endeavors to satisfy each desirable specification. The general form of the desirable specifications without assumptions is $\mathbf{G}\varphi$, which means "φ

✉ Takashi Tomita
tomita@jaist.ac.jp

Atsushi Ueno
ueno@fmx.cs.titech.ac.jp

Masaya Shimakawa
masaya@fmx.cs.titech.ac.jp

Shigeki Hagihara
hagihara@fmx.cs.titech.ac.jp

Naoki Yonezaki
yonezaki@fmx.cs.titech.ac.jp

1 Research Center for Advanced Computing Infrastructure, Japan Advanced Institute of Science and Technology, 1-1, Asahidai, Nomi, Ishikawa 923-1292, Japan

2 Department of Computer Science, Graduate School of Information Science and Engineering, Tokyo Institute of Technology, 2-12-1-W8-67, Ookayama, Meguro, Tokyo 152-8552, Japan

3 Department of Computer Science, School of Computing, Tokyo Institute of Technology, 2-12-1-W8-67, Ookayama, Meguro, Tokyo 152-8552, Japan

4 Graduate School of Information Environment, Tokyo Denki University, 2-1200, Muzai Gakuendai, Inzai, Chiba 270-1382, Japan

5 Tokyo Shibuya Branch, The Open University of Japan, 1-10-7, Dogenzaka, Shibuya, Tokyo 150-0043, Japan

always holds". In our approach, the best effort to satisfy $\mathbf{G}\varphi$ is to maximize the number of steps satisfying φ in the interaction. To quantitatively evaluate the number of steps, we used a mean-payoff objective based on LTL formulae. Our method applies the Safraless approach to construct safety games from given must and desirable specifications, where the must specification can be written in full LTL and may include assumptions. It then transforms the safety games constructed from the desirable specifications into mean-payoff games and finally composes a reactive system as an optimal strategy on a synchronized product of the games.

1 Introduction

1.1 Background

Open systems interact continuously with the external environment. When applied to real problems, they must often be highly reliable. These systems are modeled as *reactive systems*. *Linear Temporal Logic (LTL) synthesis* is a formal method for checking the *realizability* [1,34,35] of a behavioral specification described in LTL [33] and for automatically composing a reactive system realizing the specification if it is realizable. This method can effectively obtain a reliable system because it does not have a phase that introduces bugs.

In traditional LTL synthesis, if a given LTL specification is unrealizable, we must refine it in LTL. One approach for refining an unrealizable LTL specification φ is to add or strengthen the assumptions ψ regarding the environment, such that a refined specification $\psi \rightarrow \varphi$ is realizable. In [14], Chatterjee et al. proposed a method for computing some of these assumptions. However, the computed assumptions are not logical formulae in their naive method and may be difficult to understand intuitively. Additionally, it may be unallowable in a practical sense. Hagihara et al. [25] introduced a method for efficiently extracting an assumption to make a given specification *strong satisfiable*, where strong satisfiability [31] is a necessary condition of realizability. The extracted assumption is the weakest LTL formula in a certain class and easy to understand. Li et al. [30] provided a method for finding an allowable assumption to make a given specification realizable, where the assumption is mined from given LTL templates. Even if we obtain ψ, we must consider many things when synthesizing a reactive system from $\psi \rightarrow \varphi$. A system synthesized from $\psi \rightarrow \varphi$ may stop trying to satisfy φ after an unexpected input that violates ψ, i.e., it may be *intolerant* [26]. Moreover, such a system may violate ψ against the wishes of its environment. In [7], Bloem et al. discussed how to deal with environmental assumptions and surveyed existing approaches.

One approach is to weaken some of the partial specifications that cause the unrealizability of the whole specification, $\varphi = \bigwedge_{\varphi_i \in \Phi} \varphi_i$. In this approach, we first find a subset of these partial specifications, i.e., $\Phi' \subseteq \Phi$ such that $\bigwedge_{\varphi_i \in \Phi'} \varphi_i$ is unrealizable. In [24], Hagihara et al. proposed a method for finding minimal strongly unsatisfiable subsets of specifications. Even if we find these partial specifications, it is difficult (but preferable) to obtain a refined specification that is realizable and close to the original. The intention of an original partial specification might not be preserved in the refined specification. Hence, handling specification re-refinements associated with changes to the original specifications is difficult.

If the whole specification is unrealizable, it is still preferable to synthesize a best-effort reactive system, i.e., a system that *maximally realizes* its partial specifications.

1.2 Our goal and approach

A whole reactive system specification usually consists of some sub-specifications. If the whole specification is unrealizable, we first try to refine some of the sub-specifications

before we drastically reconsider the whole specification. We can divide the specifications into: *must specifications*, which should never be violated, and *desirable specifications*, the violation of which may be unavoidable. More precisely, the sub-specifications have a priority order, and each sub-specification may be violated if it competes with other higher-priority sub-specifications. Must specifications have the highest priority, and hence they should never be violated.

In this paper, we propose a method for synthesizing a reactive system that realizes all given must specifications and endeavors to satisfy the given desirable specifications as much as possible considering their priorities.

In the endeavor, we consider it is reasonable that a desirable specification has no assumption. This is because the system is attempting as best as possible, regardless of whether an assumption holds, even though the best effort will depend on the assumption. The general form of the specifications without assumptions is a **G**-formula $\mathbf{G}\varphi$ which means "φ always holds", i.e., "φ holds at every step during an infinite-step interaction with the environment". In our approach, the best effort at satisfying $\mathbf{G}\varphi$ maximizes the number of steps that satisfy φ.

Therefore, we use a *mean-payoff objective* based on LTL formulae to quantitatively evaluate the number of steps. For each desirable specification $\mathbf{G}\varphi_i$, we basically consider a positive payoff for each occurrence of a step that satisfies φ_i during an interaction. However, it is often difficult to strictly impose this payoff, because whether φ holds at a step generally depends on the entire subsequent behavior. $\mathbf{G}\varphi_i$ can be under-approximated into a safety property by assigning a bound k to a *universal co-Büchi word automaton* (UCWA), which accepts words satisfying $\mathbf{G}\varphi_i$. We then consider a negative payoff for occurrences of *minimal bad prefixes* of a safety property represented by a k-bounded UCWA [22] (k-UCWA) that rejects some of the words accepted by the unbounded UCWA. This approximation was inspired by the *Safraless approach* [8,18,22,27,28]. Our mean-payoff objective $\mathsf{MP}(\sum_{1\leq i\leq n} c_i \cdot t_i)$ is the *limit inferior of averages* for a sequence of weighted sums $\sum_{1\leq i\leq n} c_i \cdot t_i$ of payoffs t_i for desirable specifications $\mathbf{G}\varphi_i$, considering weights c_i according to their priorities.

In our method, we first apply a UCWA-based Safraless method [8,18,22] to construct an under-approximated safety game from a must LTL specification φ. We obtain a winning region $\mathcal{W}$ on the game as a set of reactive systems realizing an under-approximation of φ. Second, we construct mean-payoff games $\mathcal{M}^{t_i}$ from atomic terms t_i in the mean-payoff objective $\mathsf{MP}(\sum_{1\leq i\leq n} c_i \cdot t_i)$, reusing the procedures in the Safraless method. Finally, we compose a reactive system as an optimal strategy on a weighted synchronized product of the region $\mathcal{W}$ and the mean-payoff games $\mathcal{M}^{t_1}, \ldots, \mathcal{M}^{t_n}$. An outline of our approach is depicted in Fig. 1.

Our method does not require any restriction for a must specification and hence can also deal with an assumption-guarantee type formula $\psi \rightarrow \varphi$ as the must specification. In this case, our method produces a reactive system that realizes $\psi \rightarrow \varphi$ and locally maximizes a given mean-payoff objective regardless of whether the environment follows the assumption ψ.

1.3 Plan of the paper

In Sect. 2, we introduce some definitions and concepts concerning reactive systems, LTL, games, and Safraless synthesis. We define our mean-payoff objectives and show how to interpret desirable LTL specification in the mean-payoff objectives in Sect. 3. In Sect. 4, we describe our method for synthesizing a reactive system from a must LTL specification and a mean-payoff objective. We demonstrate the effectiveness of our method using some experiments in Sect. 5. Section 6 contains some information on related works, and Sect. 7 concludes the paper.

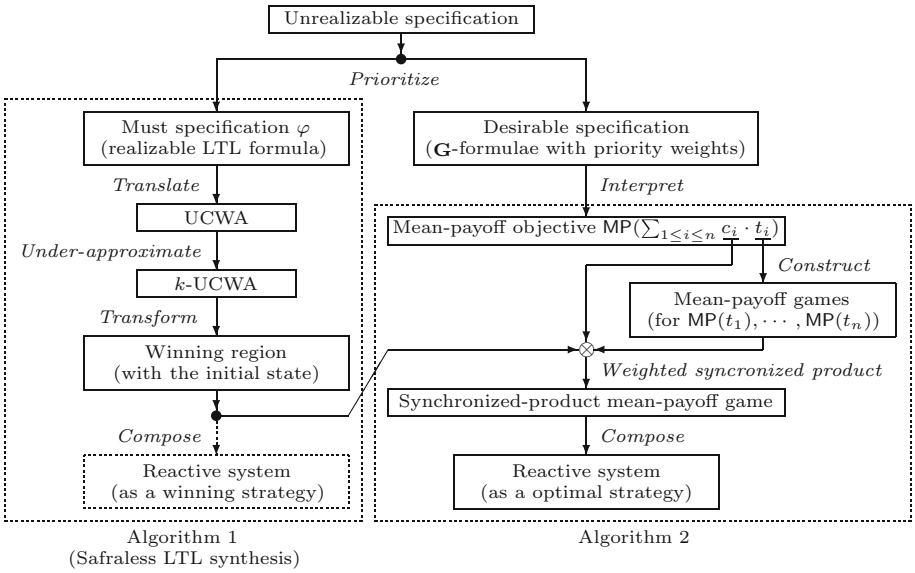

Fig. 1 Outline of our approach

2 Preliminaries

In this section, we first introduce some definitions and concepts regarding reactive systems that interact with the external environment. Second, we briefly introduce LTL [33], which is commonly used to describe formal behavioral specifications of systems. Third, we give some definitions and concepts regarding games, which model sets of interactions between systems and environment, and related decision making processes. Finally, we briefly outline Safraless synthesis [8,18,22,27,28].

2.1 Reactive systems

A *reactive system* models an open system that interacts continuously with an external *environment*. It cannot control a sequence of *inputs* from the environment and hence must choose an *output* at each step based on the history of the interaction thus far.

In this paper, we assume that an interaction starts with an output of the system.[1] Let OAP and IAP be disjoint sets of *atomic propositions* (or *signals*) for outputs and inputs controlled by the system and environment, respectively. The set of all atomic propositions, i.e., the union $IAP \cup OAP$, is denoted by AP.

- An *interaction* between a reactive system and the environment is represented as an infinite *word* on alphabet 2^{AP}, i.e., an infinite sequence $s = (\alpha_0^O \cup \alpha_0^I)(\alpha_1^O \cup \alpha_1^I) \cdots \in (2^{AP})^\omega$.
- A *reactive system* is a function $\mathcal{R} : (2^{IAP})^* \to 2^{OAP}$ that decides an output $\alpha_n^O \in 2^{OAP}$ at the current step, based on a sequence $h \in (2^{IAP})^*$ of inputs thus far.[1]

The interaction produced by $\mathcal{R}$ for a sequence $s = \alpha_0^I \alpha_1^I \ldots \in (2^{IAP})^\omega$ of inputs is denoted as $Intr_s(\mathcal{R})$. That is, $Intr_s(\mathcal{R}) = (\alpha_0^O \cup \alpha_0^I)(\alpha_1^O \cup \alpha_1^I) \cdots$, where $\alpha_0^O = \mathcal{R}(\varepsilon)$ and $\alpha_{i+1}^O =$

[1] When an interaction starts with an input from the environment, a reactive system is defined as $\mathcal{R} : (2^{IAP})^+ \to 2^{OAP}$. Cf. Footnote 3.

$\mathcal{R}(\alpha_0^I \cdots \alpha_i^I)$ for each $i \in \mathbb{N}$. A set $Intr(\mathcal{R}) = \{Intr_s(\mathcal{R}) \in (2^{AP})^\omega \mid s \in (2^{IAP})^\omega\}$ of possible interactions of $\mathcal{R}$ is denoted by $Intr(\mathcal{R})$.

2.2 Linear temporal logic (LTL)

Linear temporal logic [33] is a modal logic used to express temporal properties and is widely used to describe formal behavioral specifications of systems.

2.2.1 Syntax and semantics

Linear temporal logic has standard *logical connectives* ($\neg$, $\vee$, and $\wedge$) and *temporal operators* (the *next* operator, **X**, the *until* operator, **U**, and the *release* operator, **R**).

Definition 1 An *LTL formula* φ on AP has the form

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \mathbf{X}\varphi \mid \varphi \mathbf{U} \varphi \mid \varphi \mathbf{R} \varphi, \tag{1}$$

where $p \in AP$.

Intuitively, $\mathbf{X}\varphi$ means that "φ holds in the *next* step", $\varphi_1 \mathbf{U} \varphi_2$ means that "φ_2 eventually holds, and φ_1 continually holds *until* then" (i.e., φ_1 until φ_2), and $\varphi_1 \mathbf{R} \varphi_2$ means that "φ_2 holds until and including the point when φ_1 holds" (i.e., φ_1 releases φ_2). Note that the release operator **R** is the dual of the until-operator **U**.

The following standard abbreviations for logical connectives and symbols are also commonly used in LTL.

$$\varphi_1 \rightarrow \varphi_2 \equiv \neg\varphi_1 \vee \varphi_2, \tag{2}$$
$$\varphi_1 \leftrightarrow \varphi_2 \equiv (\varphi_1 \rightarrow \varphi_2) \wedge (\varphi_2 \rightarrow \varphi_1), \tag{3}$$
$$\top \equiv \varphi \vee \neg\varphi, \tag{4}$$
$$\bot \equiv \neg\top. \tag{5}$$

We use the following abbreviations for temporal operators.

$$\mathbf{F}\varphi \equiv \top \mathbf{U} \varphi, \tag{6}$$
$$\mathbf{G}\varphi \equiv \bot \mathbf{R} \varphi, \tag{7}$$
$$\varphi_1 \mathbf{W} \varphi_2 \equiv \varphi_2 \mathbf{R} (\varphi_1 \vee \varphi_2). \tag{8}$$

The operator **F** is often denoted by $\Diamond$, and $\mathbf{F}\varphi$ means that "φ holds *eventually* (i.e., in the *future*)". The operator **G** is often denoted by $\Box$, and $\mathbf{G}\varphi$ means that "φ holds *always* (i.e., *globally*)". The operator **W** is called the *weak-until* operator, because the second sub-formula φ_2 may never hold if the first sub-formula φ_1 always holds. We also use the following abbreviations.

$$\mathbf{X}^n \varphi \equiv \begin{cases} \varphi & \text{if } n = 0, \\ \mathbf{X}\left(\mathbf{X}^{n-1}\varphi\right) & \text{otherwise,} \end{cases} \tag{9}$$

$$\varphi_1 \mathbf{U}^{\leq n} \varphi_2 \equiv \begin{cases} \varphi_2 & \text{if } n = 0, \\ \varphi_2 \vee \left(\varphi_1 \wedge \mathbf{X}\left(\varphi_1 \mathbf{U}^{\leq n-1} \varphi_2\right)\right) & \text{otherwise,} \end{cases} \tag{10}$$

$$\varphi_1 \mathbf{R}^{\leq n} \varphi_2 \equiv \begin{cases} \varphi_2 & \text{if } n = 0, \\ (\varphi_1 \wedge \varphi_2) \vee \left(\varphi_2 \wedge \mathbf{X}\left(\varphi_1 \mathbf{R}^{\leq n-1} \varphi_2\right)\right) & \text{otherwise,} \end{cases} \tag{11}$$

$$\mathbf{F}^{\leq n}\varphi_1 \equiv \top\mathbf{U}^{\leq n}\varphi_1, \tag{12}$$
$$\mathbf{G}^{\leq n}\varphi_1 \equiv \bot\mathbf{R}^{\leq n}\varphi_1, \tag{13}$$
$$\varphi_1\mathbf{W}^{\leq n}\varphi_2 \equiv \varphi_2\mathbf{R}^{\leq n}(\varphi_1 \vee \varphi_2). \tag{14}$$

An LTL formula φ is *bounded* if it has only next-operators. Note that $\mathbf{U}^{\leq n}$ and $\mathbf{R}^{\leq n}$ (also $\mathbf{F}^{\leq n}$, $\mathbf{G}^{\leq n}$ and $\mathbf{W}^{\leq n}$) are bounded because of Eqs. (10) and (11).

Linear temporal logic semantics is defined as *satisfaction relation* $\models$ between an infinite word on 2^{AP} and an LTL formula.

Definition 2 For an infinite word $s = \alpha_0\alpha_1 \cdots \in (2^{AP})^\omega$ on 2^{AP}, the semantics of an LTL formula φ is:

$$s \models \varphi \Leftrightarrow \langle s, 0\rangle \models \varphi. \tag{15}$$

Here, $\langle s, i\rangle \models \varphi$ is a satisfaction relation between the i-th suffix $\alpha_i\alpha_{i+1}\cdots$ of s and φ, such that

$$\langle s, i\rangle \models p \quad \Leftrightarrow \quad p \in \alpha_i, \tag{16}$$
$$\langle s, i\rangle \models \neg\varphi_1 \quad \Leftrightarrow \quad \langle s, i\rangle \not\models \varphi_1, \tag{17}$$
$$\langle s, i\rangle \models \varphi_1 \vee \varphi_2 \quad \Leftrightarrow \quad \langle s, i\rangle \models \varphi_1 \text{ or } \langle s, i\rangle \models \varphi_2, \tag{18}$$
$$\langle s, i\rangle \models \varphi_1 \wedge \varphi_2 \quad \Leftrightarrow \quad \langle s, i\rangle \models \varphi_1 \text{ and } \langle s, i\rangle \models \varphi_2, \tag{19}$$
$$\langle s, i\rangle \models \mathbf{X}\varphi_1 \quad \Leftrightarrow \quad \langle s, i+1\rangle \models \varphi_1, \tag{20}$$
$$\langle s, i\rangle \models \varphi_1\mathbf{U}\varphi_2 \quad \Leftrightarrow \quad \exists j \in \mathbb{N}_{\geq i}\left(\langle s, j\rangle \models \varphi_2 \text{ and } \forall h \in \mathbb{N}_{\geq i} \cap \mathbb{N}_{<j}\left(\langle s, h\rangle \models \varphi_1\right)\right), \tag{21}$$
$$\langle s, i\rangle \models \varphi_1\mathbf{R}\varphi_2 \quad \Leftrightarrow \quad \forall j \in \mathbb{N}_{\geq i}\left(\langle s, j\rangle \models \varphi_2\right) \text{ or}$$
$$\exists j \in \mathbb{N}_{\geq i}\left(\langle s, j\rangle \models \varphi_1 \text{ and } \forall h \in \mathbb{N}_{\geq i} \cap \mathbb{N}_{\leq j}\left(\langle s, h\rangle \models \varphi_2\right)\right). \tag{22}$$

The set $\{s \in (2^{AP})^\omega \mid s \models \varphi\}$ of words satisfying φ is denoted by $\mathcal{L}(\varphi)$. An LTL formula φ is *satisfiable* (or *consistent*) if $\mathcal{L}(\varphi) \neq \emptyset$.

An LTL formula φ has *negation normal form* (NNF) if the negations $\neg$ in φ appear only as negative literals. Any LTL formula φ can be transformed into the equivalent NNF formula $nnf(\varphi)$ via a function nnf recursively defined as follows:

$$nnf(p) = p, \tag{23}$$
$$nnf(\neg p) = \neg p, \tag{24}$$
$$nnf(\neg\neg\varphi) = nnf(\varphi), \tag{25}$$
$$nnf(\varphi_1 \vee \varphi_2) = nnf(\varphi_1) \vee nnf(\varphi_2), \tag{26}$$
$$nnf(\neg(\varphi_1 \vee \varphi_2)) = nnf(\neg\varphi_1) \wedge nnf(\neg\varphi_2), \tag{27}$$
$$nnf(\varphi_1 \wedge \varphi_2) = nnf(\varphi_1) \wedge nnf(\varphi_2), \tag{28}$$
$$nnf(\neg(\varphi_1 \wedge \varphi_2)) = nnf(\neg\varphi_1) \vee nnf(\neg\varphi_2), \tag{29}$$
$$nnf(\mathbf{X}\varphi_1) = \mathbf{X}(nnf(\varphi_1)), \tag{30}$$
$$nnf(\neg\mathbf{X}\varphi_1) = \mathbf{X}(nnf(\neg\varphi_1)), \tag{31}$$
$$nnf(\varphi_1\mathbf{U}\varphi_2) = (nnf(\varphi_1))\mathbf{U}(nnf(\varphi_2)), \tag{32}$$
$$nnf(\neg(\varphi_1\mathbf{U}\varphi_2)) = (nnf(\neg\varphi_1))\mathbf{R}(nnf(\neg\varphi_2)), \tag{33}$$
$$nnf(\varphi_1\mathbf{R}\varphi_2) = (nnf(\varphi_1))\mathbf{R}(nnf(\varphi_2)), \tag{34}$$
$$nnf(\neg(\varphi_1\mathbf{R}\varphi_2)) = (nnf(\neg\varphi_1))\mathbf{U}(nnf(\neg\varphi_2)). \tag{35}$$

An LTL formula φ is *safe* if $nnf(\varphi)$ has no until-operator $\mathbf{U}$ (and also $\mathbf{F}$).

2.2.2 Realizability

A reactive system specification φ described in LTL specifies a set of valid interactions as $\mathcal{L}(\varphi)$. Note that a reactive system must decide an output at each step using the history of the interaction up to then. Thus, a reactive system specification given as an LTL formula φ must be *realizable* [1,34,35] (or *programmable*).

Definition 3 A property $L \subseteq (2^{AP})^\omega$ is *realizable* if there exists a reactive system $\mathcal{R} : (2^{IAP})^* \to 2^{OAP}$ such that

$$Intr(\mathcal{R}) \subseteq L. \tag{36}$$

An LTL formula φ on AP is *realizable* if $\mathcal{L}(\varphi)$ is realizable.

2.2.3 Minimal bad prefixes

For a *language* $L \subseteq \Sigma^\omega$ (i.e., a set of words) on an alphabet Σ, we can sometimes determine if a word s is not in L using only a prefix of s. Such a prefix is called a *bad prefix* of L. Formally, a nonempty finite word $s \in \Sigma^+$ on Σ is a *bad prefix* of $L \subseteq \Sigma^\omega$ if $ss' \notin L$ for any infinite word $s' \in \Sigma^\omega$, and it is *minimal* if any proper prefix of it is not a bad prefix of L. A set of minimal bad prefixes of L is denoted by $BadPref(L)$.

A language $L \subseteq \Sigma^\omega$ is a *safety property* if L is equivalent to the complement $\Sigma^\omega \setminus (BadPref(L)(\Sigma^\omega))$ of a set of words with bad prefixes of L. If φ is a safe LTL formula, $\mathcal{L}(\varphi)$ is a safety property.

2.2.4 LTL-to-automata translation

For verification, an LTL formula φ is often translated into an ω-automaton accepting $\mathcal{L}(\varphi)$. The set of accepting words for an automaton $\mathcal{A}$ is also denoted by $\mathcal{L}(\mathcal{A})$. LTL formula φ is *equivalent* to $\mathcal{A}$ if $\mathcal{L}(\varphi) = \mathcal{L}(\mathcal{A})$.

A *UCWA* accepts a word if *all* of its corresponding runs only visit their rejecting states *finitely many times*. A UCWA is a dual of a *nondeterministic Büchi word automaton* (NBWA). We can therefore translate an LTL formula φ into an equivalent UCWA $\mathcal{A}^\varphi$, using a standard LTL-to-NBWA translating technique [3,17,23,40]. In automata-based approaches, synthesis and probabilistic model checking require that a used automaton is *deterministic*,[2] unlike naive model- and satisfiability-checking. Safra's construction [32,36–38] is used to determinize UCWA (and NBWA) and is very complicated.

A *k-bounded UCWA* (k-UCWA) [22] accepts a word if all of its corresponding runs visit their rejecting states *at most k times*. A UCWA can be under-approximated using a bound k. A larger bound provides a UCWA that is closer to the original. In other words, for any mapping $\mathcal{T}$ from each LTL formula to an equivalent UCWA,

$$\mathcal{L}\left(\mathcal{T}^k(\varphi)\right) \subseteq \mathcal{L}\left(\mathcal{T}^{k+1}(\varphi)\right) \subseteq \mathcal{L}(\varphi), \tag{37}$$

where $\mathcal{T}^k$ denotes a mapping from LTL formula φ to a k-UCWA, which is an under-approximation of $\mathcal{T}(\varphi)$ using bound k. The acceptance language of a k-UCWA is a safety property because of its acceptance condition.

[2] For probabilistic model checking, a deterministic automaton is required for Markov decision processes; however, an *unambiguous* automaton suffices for Markov chains [4].

A *universal safety word automaton* (USWA) is a UCWA with at most one rejecting state, which loops on itself in every case. For any safe LTL formula, there exists a USWA equivalent to the formula because a word violates a given safe property if and only if the word has a (minimal) bad prefix. In this paper, we assume that an LTL-to-UCWA translator $\mathcal{T}$ is reasonable, i.e., for a safe LTL formula φ and any bound k,

$$\mathcal{L}\left(\mathcal{T}\left(\varphi\right)\right) = \mathcal{L}\left(\mathcal{T}^{k}\left(\varphi\right)\right). \tag{38}$$

We can easily construct a deterministic safety automaton equivalent to a k-UCWA or USWA, using a type of powerset (i.e., *Safraless*) construction.

2.3 Games

A *game* models the set of interactions among agents and a related decision-making process. In LTL synthesis, a game is used to model the set of interactions between a reactive system and the environment.

Definition 4 A (two-player) *game* $\mathcal{G}$ is a triple $\langle A, q_{init}, outcome\rangle$, where

- $A = \langle V_0, V_1, \Gamma_0, \Gamma_1, E_0, E_1\rangle$ is an *arena*, where
  - V_0 (resp., V_1) is a disjoint set of *states* for Player 0 (resp., Player 1)
  - Γ_0 (resp., Γ_1) is a set of *actions* for Player 0 (resp., Player 1),
  - $E_0 : V_0 \times \Gamma_0 \to V_1$ (resp., $E_1 : V_1 \times \Gamma_1 \to V_0$) is a (possibly, partial) *transition function* which maps to V_1 (resp., V_0) from V_0 (resp., V_1) and Γ_0 (resp., Γ_1),
- $v_{init} \in V_0$ is the *initial* state,
- $outcome : (E_0E_1)^{\omega} \to \mathbb{R}$ is a function that gives the outcome of a *play* $\rho \in (E_0E_1)^{\omega}$.

When the current state is v in V_0 (resp., V_1), Player 0 (resp., Player 1) chooses its action γ in Γ_0 (resp., Γ_0), and the state moves according to transition function E_0 (resp., E_1). Let $\sigma \in \{0, 1\}$. If $E_\sigma(v, \gamma) = v'$, there is a transition to v' from v with γ. If $E_\sigma(v, \gamma)$ is undefined for some action γ (i.e., E_σ is a partial function), γ is *unavailable* on v. In this paper, we assume that all states have at least one available action; i.e., there is no *dead-end* state. We let E_σ be a set $\{\langle v, \gamma, v'\rangle \mid E_\sigma(v, \gamma) = v'\}$ of triples that represent transitions. For an available transition $e = \langle v, \gamma, v'\rangle$, its predecessor state v, successor state v', and action γ are denoted by $pred(e)$, $succ(e)$, and $act(e)$, respectively.

A *play* ρ on $\mathcal{G}$ is an infinite alternating sequence $e_0e_1e_2e_3\cdots \in (E_0E_1)^{\omega}$ of 0- and 1-transitions, where the starting state $pred(e_0)$ is the initial state v_{init}. It is *available* if $succ(e_i) = pred(e_{i+1})$ for each $i \in \mathbb{N}$. A set of available plays on $\mathcal{G}$ is denoted by $Play(\mathcal{G})$.

An outcome of an available play is evaluated according to function *outcome*. When the outcomes are in a binary set, an outcome is often interpreted as either *win* or *loss*. In this paper, we use two types of game: *safety games* and *mean-payoff games*.

2.3.1 Safety games

The outcome of a *safety game* is evaluated based on a *safety condition* and is mapped to the binary $\{0, 1\}$.

Definition 5 A game $\mathcal{S} = \langle A, v_{init}, outcome^S\rangle$ with an arena $A = \langle V_0, V_1, \Sigma_0, \Sigma_1, E_0, E_1\rangle$ is a *safety game* if, for a set $S \subseteq V_0 \cup V_1$ of *safe states*, the outcome $outcome^S(\rho)$ for an available play $\rho = \langle v_0, \gamma_0, v_1\rangle\langle v_1, \gamma_1, v_2\rangle \in Play(\mathcal{S})$ on $\mathcal{S}$ is

$$outcome^S(\rho) = \begin{cases} 1 & \text{if } v_i \in S \quad \text{for all } i \in \mathbb{N}, \\ 0 & \text{otherwise.} \end{cases} \tag{39}$$

Player 0 *wins* (and Player 1 *loses*) for ρ if $outcome^S(\rho) = 1$, i.e., ρ stays in S forever (*safety condition*). Otherwise, i.e., if ρ reaches the complement of S, Player 0 loses (and Player 1 wins). Therefore, this kind of game is a *reachability game* from the standpoint of Player 1.

The outcome $outcome^S$ is characterized by S, and therefore we use a triple $\langle A, v_{init}, S\rangle$ instead of $\langle A, v_{init}, outcome^S\rangle$.

2.3.2 Mean-payoff games

The outcome of a *mean-payoff game* is evaluated based on the *limit inferior of averages* for a sequence of payoffs assigned at each transition according to a given weighting function.

Definition 6 A game $\mathcal{M} = \langle A, v_{init}, outcome^W\rangle$ with an arena $A = \langle V_0, V_1, \Sigma_0, \Sigma_1, E_0, E_1\rangle$ is a *mean-payoff game* if, for a *weighting function* $W : (E_0 \cup E_1) \to \mathbb{Z}$ that gives an integer weight for a transition, the outcome $outcome^W(\rho)$ for an available play $\rho = e_0 e_1 \cdots \in Play(\mathcal{M})$ on $\mathcal{M}$ is

$$outcome^W(\rho) = \liminf_{n\to\infty} \frac{1}{n+1} \sum_{i=0}^{n} W(e_i). \tag{40}$$

For a mean-payoff game, a threshold k is often given as a winning condition. That is, Player 0 wins (and Player 0 loses) for a play $\rho \in Play(\mathcal{M})$ if $outcome^W(\rho) \geq k$.

The outcome $outcome^W$ is characterized by W, and so we use a triple $\langle A, v_{init}, W\rangle$ instead of $\langle A, v_{init}, outcome^W\rangle$.

2.3.3 Strategies

The goal of Player 0 (resp., Player 1) is to maximize (resp., to minimize) the outcome. Player $\sigma \in \{0, 1\}$ must choose an available action (and its corresponding transition) for their current state v based on a finite sequence $e_0 e_1 \ldots e_{2n+\sigma-1} \in E_0 E_1 \ldots E_{1-\sigma} E_\sigma$ of previous transitions such that $v_{init} = pred(e_0)$, $succ(e_i) = pred(e_{i+1})$ for $0 \leq i \leq 2n + \sigma - 1$, and $v = succ(e_{2n+\sigma-1})$.

A *strategy* $\mu_0 : E_1^* \to E_0$ of Player 0 (resp., $\mu_1 : E_0^+ \to E_1$ of Player 1) on a game $\mathcal{G}$ is a function that decides the next transition e_{2n} (resp., e_{2n+1}) from a set $\{e \in E_0 \mid succ(e_{2n-1}) = pred(e)\}$ (resp., $\{e \in E_1 \mid succ(e_{2n}) = pred(e)\}$) of available transitions from $succ(e_{2n-1})$ (resp., $succ(e_{2n})$). The 0-strategy μ_0 and 1-strategy μ_1 pair derives a play $\rho^{\{\mu_0,\mu_1\}} = e_0 e_1 \ldots \in Play(\mathcal{G})$ where $e_0 = \mu_0(\varepsilon)$, $e_{2i+1} = \mu_1(e_0 e_2 \ldots e_{2i})$, and $e_{2i+2} = \mu_0(e_1 e_3 \ldots e_{2i+1})$.

A 0-strategy μ_0 is *optimal* if μ_0 maximizes the worst-case outcome, i.e., for any 0-strategy μ_0',

$$\inf_{\mu_1} \left\{ outcome\left(\rho^{\{\mu_0,\mu_1\}}\right)\right\} \geq \inf_{\mu_1} \left\{ outcome\left(\rho^{\{\mu_0',\mu_1\}}\right)\right\}. \tag{41}$$

Alternatively, a 1-strategy μ_1 is *optimal* if, μ_1 minimizes the best-case outcome, i.e., for any 1-strategy μ_1',

$$\sup_{\mu_0} \left\{ outcome\left(\rho^{\{\mu_0,\mu_1\}}\right)\right\} \leq \sup_{\mu_0} \left\{ outcome\left(\rho^{\{\mu_0,\mu_1',\}}\right)\right\}. \tag{42}$$

In particular, μ_0 (resp. μ_1) is *winning* if $\mathcal{G}$ is a safety game, and $outcome(\rho^{\{\mu_0,\mu_1\}})$ is 1 (resp. 0) for any 1-strategy μ_1 (resp. any 0-strategy μ_0).

A 0-strategy μ_0 (resp. 1-strategy μ_1) is *memoryless* or *positional* if there exists a function $\nu_0 : V_0 \to E_0$ (resp. $\nu_1 : V_1 \to E_1$) that gives a next available transition for the current state, such that $\mu_0(\varepsilon) = \nu_0(v_{init})$ and $\mu_0(e_0 \dots e_i) = \nu_0(succ(e_i))$ (resp. $\mu_1(e_0 \dots e_i) = \nu_1(succ(e_i))$). In this case, ν_0 (resp. ν_1) is equivalent to μ_0 (resp. μ_1).

For safety games [19] and mean-payoff games [20], each player has a strategy that is both optimal and memoryless. Therefore, we only consider memoryless strategies in the following sections.

2.3.4 Winning regions

For a safety game $\mathcal{G}$ with an arena $A = \langle V_0, V_1, \Gamma_0, \Gamma_1, E_0, E_1\rangle$, an initial state v_{init}, and a safe region $S \subseteq V_0 \cup V_1$, Player 0 will try to choose a transition such that they stay in S.

The *winning region* $\mathcal{W}$ for Player 0 is the maximal sub-graph $\langle V_0', V_1', \Gamma_0, \Gamma_1, E_0', E_1'\rangle$ of A such that $V_0' \subseteq V_0 \cap S$, $V_1' \subseteq V_1 \cap S$, $E_0' \subseteq E_0 \cap (V_0' \times \Gamma_0 \times V_1')$ and $E_1' = E_1 \cap (V_1' \times \Gamma_1 \times V_0)$. Player 0 can ensure that they stay in $\mathcal{W}$ because any 1-state $v \in V_1'$ has no successor outside of S. The region is maximal, and hence Player 0 has a winning strategy if $v_{init} \in V_0'$. In this case, $\mathcal{W}$ represents the set of all winning strategies on $\mathcal{G}$, and we let $\mathcal{W}$ be a sub-game $\langle \mathcal{W}, v_{init}, V_0' \cup V_1'\rangle$ consisting of $\mathcal{W}$.

For a safety game, we can efficiently extract the winning region using a simple fixed-point computation, in time $\mathcal{O}(|E_0| + |E_1|)$ [19].

2.3.5 Reactive systems as strategies

The safety game $\mathcal{G}$ used in Safraless synthesis has the following characteristics.[3]

- Players 0 and 1 represent system-side and environment-side players, respectively.
- Γ_0 is 2^{OAP}, and Γ_1 is 2^{IAP}.
- E_0 may be partial, whereas E_1 must be total.
- An available play $\rho = e_0 e_1 \dots \in Play(\mathcal{G})$ corresponds to an interaction $trace(\rho) \in (2^{AP})^\omega$,
  - where a trace $trace(\rho)$ on ρ is an infinite word $(act(e_0) \cup act(e_1))(act(e_2) \cup act(e_3)) \cdots (act(e_{2i}) \cup act(e_{2i+1})) \dots$.

Therefore, a memoryless 0-strategy ν_0 on $\mathcal{G}$ can be regarded as a reactive system $\mathcal{R}^{\nu_0}$ such that

- $\mathcal{R}^{\nu_0}(\varepsilon) = \alpha_0^O$ and $\mathcal{R}^{\mu_0}(\alpha_0^I \dots \alpha_i^I) = \alpha_{i+1}^O$ where, for $0 \le j \le i$,
  - $v_0 = v_{init}$, $v_{2(j+1)} = E_1(E_0(v_{2j}, \alpha_j^O), \alpha_j^I)$, and
  - $\alpha_0^O = act(\nu_0(v_0))$ and $\alpha_{j+1}^O = \nu_0(act(v_{2(j+1)}))$.

[3] When an interaction starts with an input from the environment, the roles of Players 0 and 1 are switched. Cf. Footnote 1.

2.4 Safraless synthesis

Safraless synthesis [8,18,22,27,28] is a mainstream method for LTL synthesis.

The outline of a UCWA-based method [8,18,22] is given in Algorithm 1. For a given LTL formula φ on a set AP ($= IAP \cup OAP$) of atomic propositions, we first construct a UCWA $\mathcal{A}^{\varphi}$ accepting $\mathcal{L}(\varphi)$ (the procedure `LTL2UCWA_translation` at Line 1) and initialize the bound k to 0 (Line 2). Let $\mathcal{T}$ be an LTL-to-UCWA mapping implemented by `LTL2UCWA_translation`, i.e., $\mathcal{A}^{\varphi} = \mathcal{T}(\varphi)$. We next derive an under-approximation to $\mathcal{A}^{\varphi}$ using k and transform the under-approximated k-UCWA $\mathcal{T}^k(\varphi)$ into a safety game $\mathcal{S}^{\varphi,k}$ (the procedure `BUCWA2SG_transformation` at Line 4). This transformation consists of determinizing $\mathcal{A}^{\varphi}$ and dividing its states/transitions into system-side and environment-side ones based on the respective sets OAP and IAP of output and input propositions. $\mathcal{S}^{\varphi,k}$ has characteristics described in Sect. 2.3.5. Additionally, the winning condition of $\mathcal{S}^{\varphi,k}$ corresponds to the under-approximated property $\mathcal{L}(\mathcal{T}^k(\varphi))$ of φ. Therefore, if the system-side player wins for a play ρ on $\mathcal{S}^{\varphi,k}$, $trace(\rho)$ satisfies φ. Then we extract a winning region $\mathcal{W}^{\varphi,k}$ for the system-side player on $\mathcal{S}^{\varphi,k}$ (the `extract_winning_region` at Line 5). If the initial state of $\mathcal{S}^{\varphi,k}$ is in $\mathcal{W}^{\varphi,k}$ (Line 6), there exists a reactive system that realizes φ. Otherwise, k is incremented (Line 9), and we repeat a loop from Line 3. Lines 1–10 check the realizability of φ, and in practice the unrealizability of φ is also checked in parallel. Finally, we concretely compose a winning strategy ν for the system-side player on $\mathcal{W}^{\varphi,k}$ (the procedure `winning_strategy` at Line 11).

In this paper, we have omitted the details of the procedure. For simplicity, we assume that the subprocedure `BUCWA2SG_transformation` produces a safety game $\langle\langle V_0, V_1, \Sigma_0, \Sigma_1, E_0, E_1\rangle, v_{init}, S\rangle$ such that E_0 is total.

3 Mean-payoff objectives

In this section, we focus on desirable specifications and introduce the mean-payoff objective that determines the desirable specifications.

Algorithm 1 Safraless synthesis

Input: A realizable LTL formula φ on atomic propositions $AP = IAP \cup OAP$, and an initial bound k_{init}
Output: A reactive system realizing φ, as a winning strategy ν on a winning region $\mathcal{W}^{\varphi,k}$

1: $\mathcal{A}^{\varphi} :=$ `LTL2UCWA_translation`(φ) // Translating φ into a UCWA $\mathcal{A}^{\varphi}$
2: $k := k_{init}$ // Initializing bound k
3: **loop**
4: $\mathcal{S}^{\varphi,k} :=$ `BUCWA2SG_transformation`$(\mathcal{A}^{\varphi}, k, OAP, IAP)$ // Transforming k-UCWA $\mathcal{A}^{\varphi,k}$ into a safety game $\mathcal{S}^{\varphi,k}$
5: $\mathcal{W}^{\varphi,k} :=$ `extract_winning_region`$(\mathcal{S}^k)$ // Extracting a winning region $\mathcal{W}^{\varphi,k}$ for the system-side player on $\mathcal{S}^{\varphi,k}$
6: **if** The initial state of $\mathcal{S}^{\varphi,k}$ is in $\mathcal{W}^{\varphi,k}$ **then**
7: **break**
8: **end if**
9: $k := k + 1$
10: **end loop** // Lines 1–10 check the realizability of φ
11: $\nu :=$ `winning_strategy`$(\mathcal{W}^{\varphi,k})$ // Composing a (memoryless) winning strategy ν of the system-side player on $\mathcal{W}^{\varphi,k}$
12: **return** $\langle \nu, \mathcal{W}^{\varphi,k} \rangle$

3.1 Basic idea

The general form of desirable specifications without assumptions is a **G**-formula $\mathbf{G}\varphi$. We consider that a preferable reactive system should endeavor to satisfy φ at each step, i.e., to maximize the number of steps satisfying φ to the extent possible.

We fit a mean-payoff to quantitatively evaluate the desirability if the payoff at each step depends on the importance of the desirable specifications $\mathbf{G}\varphi$ and whether φ holds at the step. Therefore, we propose a mean-payoff objective, which covers the set of desirable specifications. We can optimize a reactive system using the mean-payoff objective, which quantitatively specifies the desirability of interactions with the environment.

We are left with the problem of determining whether φ holds at each step. If φ is bounded, we can determine if φ holds at a step using only the behavior of the next n-steps, where n is the depth of the next-operator **X**. That is, the length of a subsequence deciding whether φ holds at each step is less than $n+1$. An example of when $\varphi = a \rightarrow \mathbf{X}(b\mathbf{U}^{\leq 1}c)$ is shown in Fig. 2. A mean-payoff value is preserved, even if every payoff at a step is given bounded steps later. Hence, it is easy to deal with a bounded LTL formula.

If φ is unbounded, we may need to know the entire behavior after a step to determine whether φ holds at the step. It is difficult to strictly apply an unbounded LTL formula, and hence we use the UCWA approximation. For a desirable specification $\mathbf{G}\varphi$ and an LTL-to-UCWA mapping $\mathcal{T}$, consider a b-UCWA $\mathcal{T}^b(\mathbf{G}\varphi)$ that is obtained by giving bound b to UCWA $\mathcal{T}(\mathbf{G}\varphi)$. An interaction satisfies $\mathbf{G}\varphi$ if it has no minimal bad prefix of $\mathcal{L}(\mathcal{T}^b(\mathbf{G}\varphi))$ because $\mathcal{L}(\mathcal{T}^b(\mathbf{G}\varphi)) \subseteq \mathcal{L}(\mathbf{G}\varphi)$. Based on the equivalence between $\mathbf{G}\varphi$ and $\mathbf{GG}\varphi$, trying to avoid violating $\mathbf{G}\varphi$ at each step can be considered as another attempt to satisfy $\mathbf{G}\varphi$. In this sense, maximizing the average length (or minimizing the number) of the minimal bad prefixes occurring on interaction corresponds to the attempt. However, it is difficult to maximize strictly the average length. This is because they are overlapping, and their lengths are unbounded in general. Consider an example of an under-approximated safety property of $\mathbf{G}(p_1 \rightarrow \mathbf{X}(p_2\mathbf{U}p_3))$ is $\mathbf{G}(p_1 \rightarrow \mathbf{X}(p_2\mathbf{U}^{\leq 1}p_3))$ and an interaction is $\{p_1\}\{p_2\}\{p_1, p_3\}\{p_1, p_2\}\{p_2\}\{p_1, p_2\}\{p_1, p_3\}\emptyset\cdots$, as shown in Fig. 3. Figure 2 indicates that in this example only the third, fourth, and seventh occurrences of minimal bad prefixes

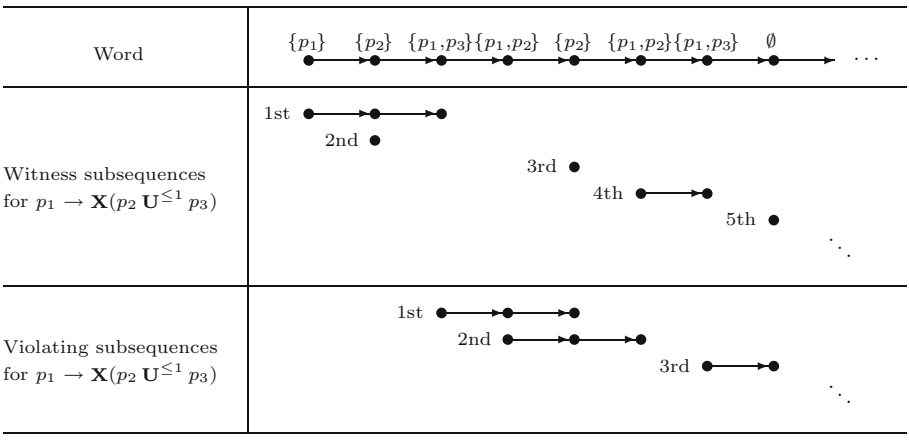

Fig. 2 Witness/violating subsequences for which the first steps satisfy/violate a bounded formula $p_1 \rightarrow \mathbf{X}(p_2\mathbf{U}^{\leq 1}p_3)$ for a word $\{p_1\}\{p_2\}\{p_1, p_3\}\{p_1, p_2\}\{p_2\}\{p_1, p_2\}\{p_1, p_3\}\emptyset\ldots$

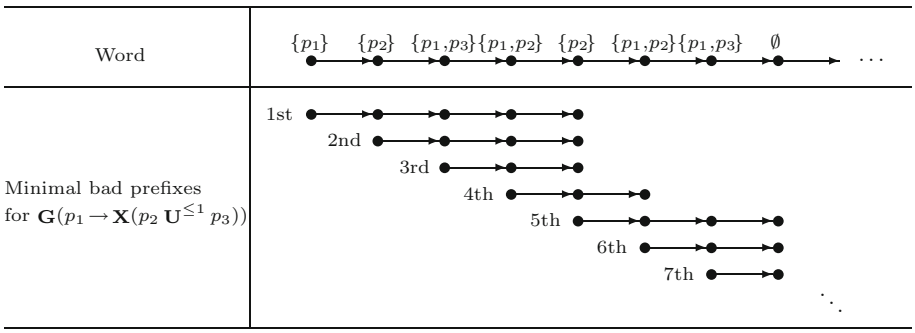

Fig. 3 Counting minimal bad prefixes of an over-approximated safety property $\mathbf{G}(p_1 \rightarrow \mathbf{X}(p_2\mathbf{U}^{\leq 1}p_3))$ of $\mathbf{G}(p_1 \rightarrow \mathbf{X}(p_2\mathbf{U}p_3))$ for a word $\{p_1\}\{p_2\}\{p_1, p_3\}\{p_1, p_2\}\{p_2\}\{p_1, p_2\}\{p_1, p_3\}\emptyset\cdots$

play a role in the violation of $\mathbf{G}(p_1 \rightarrow \mathbf{X}(p_2\mathbf{U}^{\leq 1}p_3))$. The others include them as suffixes. In this paper, we focus on the frequency of occurrences of non-overlapping ones, which is inversely proportional to the average length of them. That is, in Fig. 3, the first and sixth occurrences of minimal bad prefixes are counted and the others are not. The first (resp., sixth) occurrences can be considered as a representative subsequence among those that include the third (resp., seventh) one. We can roughly maximize the average length, by giving a negative payoff for each occurrence of non-overlapping ones and maximizing a mean-payoff under the setting.

3.2 Syntax and semantics

We now introduce the syntax and semantics for mean-payoff objectives, based on the ideas in the previous subsection.

The syntax of mean-payoff objectives is defined, in a similar manner to [39], as follows.

Definition 7 A *payoff term* t specifies a payoff at each step and is defined as

$$t ::= \mathsf{S}(\chi) \mid \mathsf{B}^b(\varphi) \mid t + t \mid c \cdot t, \tag{43}$$

where $c \in \mathbb{Z}$ is an integer constant, χ a bounded LTL formula, φ an LTL formula, $b \in \mathbb{N}$ a natural constant representing an approximation parameter, and $+$ and $\cdot$ are addition and product-by-constant operators.

For a payoff term t, a *mean-payoff objective* is a term $\mathsf{MP}(t)$.

Intuitively, $\mathsf{S}(\chi)$ means that a payoff of 1 is given at a step if χ holds at the step, i.e., $\mathsf{S}(\chi)$ strictly captures whether χ holds at the step. A value of $\mathsf{S}(\chi)$ at a step depends on the behavior after the next n-steps, where n is the depth of $\mathbf{X}$ in χ. However, $\mathsf{B}^b(\varphi)$ means that a payoff of -1 is given for each occurrence of minimal bad prefixes of $\mathcal{L}(\mathcal{T}^b(\varphi))$, where the overlapping of the prefixes is not considered. The value of $\mathsf{B}^b(\varphi)$ at a step depends on the preceding behavior. The coefficient c of the product term $c \cdot t$ is used as a weight that depends on the desirability of t. We abbreviate $c \cdot \mathsf{S}(\top)$ to c and $\mathsf{B}^0(\varphi)$ to $\mathsf{B}(\varphi)$.

Note that a B-term $\mathsf{B}^b(\varphi)$ should be treated attentively because the precise meaning of the B-term depends on an LTL-to-automata translation $\mathcal{T}$ that does not appear in the syntax. From only the syntax, it is impossible to understand the structure of $\mathcal{T}(\varphi)$, and hence it is not clear what is lost in $\mathcal{T}^b(\varphi)$. For example, a certain translator under-approximates $\mathbf{GF}p$ into $\mathbf{GF}^{\leq b}p$, but another translator outputs $\mathbf{GF}^{\leq b+1}p$. The loss depends strongly on $\mathcal{T}$, although,

it becomes smaller when b becomes larger. Even if we do not use sufficiently large b, we can estimate the loss for b when we fix $\mathcal{T}$ and construct experimentally $\mathcal{T}(\varphi)$ in advance. For instance, the size of the state space and the length of the shortest path reaching to rejecting states from the initial state, etc. of $\mathcal{T}(\varphi)$ will be important information for the estimation. Another reason is that a B-term focuses on only non-overlapped minimal bad prefixes. For the example shown in Fig. 3, the first and sixth occurrences of minimal bad prefixes are counted and the others are not. We do not strictly count occurrences of minimal bad prefixes, which play an important role in the violation of $\mathcal{T}^b(\varphi)$.

The semantics of mean-payoff objectives is given in the following.

Definition 8 Let $\mathcal{T}$ be an LTL-to-UCWA mapping. For an interaction $s \in (2^{AP})^\omega$, the value $[\![\mathsf{MP}(t)]\!]_s^{\mathcal{T}}$ of a mean-payoff objective $\mathsf{MP}(t)$ under $\mathcal{T}$ is defined as

$$[\![\mathsf{MP}(t)]\!]_s^{\mathcal{T}} = \liminf_{n\to\infty} \frac{1}{n+1} \sum_{i=0}^{n} \langle\!\langle t \rangle\!\rangle_s^{\mathcal{T}}(i). \tag{44}$$

Here, a function $\langle\!\langle t \rangle\!\rangle_s^{\mathcal{T}} : \mathbb{N} \to \mathbb{Z}$ gives an integer payoff at each step for term t on $s = \alpha_0\alpha_1 \cdots$ under $\mathcal{T}$, such that

$$\langle\!\langle \mathsf{S}(\chi) \rangle\!\rangle_s^{\mathcal{T}}(i) = \begin{cases} 1 & \text{if } \langle s, i\rangle \models \chi, \\ 0 & \text{otherwise,} \end{cases} \tag{45}$$

$$\langle\!\langle \mathsf{B}^b(\varphi) \rangle\!\rangle_s^{\mathcal{T}}(i) = \begin{cases} -1 & \text{if } \alpha_0 \ldots \alpha_i \in \left(\mathit{BadPref}\left(\mathcal{L}\left(\mathcal{T}^b\left(\varphi\right)\right)\right)\right)^+, \\ 0 & \text{otherwise,} \end{cases} \tag{46}$$

$$\langle\!\langle t_1 + t_2 \rangle\!\rangle_s^{\mathcal{T}}(i) = \langle\!\langle t_1 \rangle\!\rangle_s^{\mathcal{T}}(i) + \langle\!\langle t_2 \rangle\!\rangle_s^{\mathcal{T}}(i), \tag{47}$$

$$\langle\!\langle c \cdot t_1 \rangle\!\rangle_s^{\mathcal{T}}(i) = c \cdot \langle\!\langle t_1 \rangle\!\rangle_s^{\mathcal{T}}(i). \tag{48}$$

Note that $\mathsf{S}(\chi)$ is not equivalent to $1 - \mathsf{S}(\neg\chi)$ because the limit inferior is not generally equal to the limit superior. Additionally, note that $-\mathsf{B}(\mathbf{G}\chi)$ is not equivalent to $\mathsf{S}(\neg\chi)$. $\mathsf{S}(\neg\chi)$ strictly counts steps that violate χ, whereas $-\mathsf{B}(\mathbf{G}\chi)$ is an approximate count via the set $\mathit{BadPref}(\mathcal{L}(\mathbf{G}\chi))$ of minimal bad prefixes of $\mathcal{L}(\mathbf{G}\chi)$. Therefore,

$$[\![\mathsf{MP}(-\mathsf{B}(\mathbf{G}\chi))]\!]_s^{\mathcal{T}} \leq [\![\mathsf{MP}(\mathsf{S}(\neg\chi))]\!]_s^{\mathcal{T}}. \tag{49}$$

Additionally, note that we put no limitation on an argument of a B-term.

3.2.1 Choice on LTL-to-automata translators

In formal verification and analysis, it is normally preferable that an LTL-to-automata translator composes an automaton not only efficiently but also with tractable characteristics, e.g., smaller state space, simpler acceptance condition, and certain limitation on transition function. Most existing translators, e.g., LTL2BA[4] [23], SPOT[5] [17] and LTL3BA[6] [3], try to compose such automata. Our semantics of mean-payoff objectives depend strongly on an

[4] Available at http://www.lsv.ens-cachan.fr/~gastin/ltl2ba/.

[5] Available at https://spot.lrde.epita.fr/.

[6] Available at http://sourceforge.net/projects/ltl3ba/.

LTL-to-automata translator $\mathcal{T}$, so that the choice for $\mathcal{T}$ is very important, especially when bounds for B-terms are not sufficiently large.

However, the preference in the normal sense may not be suited to our approach. This is because the preference on LTL-to-automata translators in our method should be considered on the premise of using the k-UCWA approximation.

There are questions on the preferable translation in our method; however, they are beyond the scope of this paper.

3.2.2 The relations between mean-payoff objectives and GF-/FG-formulae

The LTL formulae $\mathbf{GF}\varphi$ and $\mathbf{FG}\varphi$ are two of the simplest over-approximated properties of $\mathbf{G}\varphi$. Definition 8 implies, for a mean-payoff objective $\mathsf{MP}(\mathsf{S}(\chi))$,

$$s \models \mathbf{FG}\chi \;\Rightarrow\; [\![\mathsf{MP}(\mathsf{S}(\chi))]\!]^{\mathcal{T}}_s = 1, \tag{50}$$

$$[\![\mathsf{MP}(\mathsf{S}(\chi))]\!]^{\mathcal{T}}_s > 0 \;\Rightarrow\; s \models \mathbf{GF}\chi. \tag{51}$$

They mean that a word with a greater value for $\mathsf{MP}(\mathsf{S}(\chi))$ has a property closer to $\mathbf{FG}\chi$. Note that the converse of Eq. (50) is not generally true. For a word s on which $\neg\chi$ holds infinitely and negligibly often, the right-hand side holds but the left does not. That is, a word may not satisfy $\mathbf{FG}\chi$ even if the word fully maximizes $\mathsf{MP}(\mathsf{S}(\chi))$.

For a safe $\mathbf{G}$-formula $\mathbf{G}\varphi$, Eq. (38) based on our assumption for LTL-to-automata translation implies that $\mathsf{B}(\mathbf{G}\varphi)$ is equivalent to $\mathsf{B}^b(\mathbf{G}\varphi)$. Additionally,

$$s \models \mathbf{FG}\varphi \;\Rightarrow\; [\![\mathsf{MP}(\mathsf{B}(\mathbf{G}\varphi))]\!]^{\mathcal{T}}_s = 0. \tag{52}$$

3.2.3 Mean-payoff objectives for words with periodic suffixes

In many synthesis methods, a synthesized reactive system is a memoryless (or finite-memory) strategy on a game. Such reactive system produces an interaction with a periodic suffix for any input sequence with a periodic suffix. The input sequences are produced by the environment given as a memoryless counter strategy against the reactive system. For a mean-payoff objective, it is sufficient in practice to consider the interaction because in mean-payoff game each player has a strategy that is both optimal and memoryless. For a finite word $s_0 \in (2^{AP})^*$ and a nonempty finite word $s_1 \in (2^{AP})^+$, we have

$$[\![\mathsf{MP}(\mathsf{S}(\chi))]\!]^{\mathcal{T}}_{s_0(s_1)^\omega} = [\![\mathsf{MP}(1-\mathsf{S}(\neg\chi))]\!]^{\mathcal{T}}_{s_0(s_1)^\omega}, \tag{53}$$

$$[\![\mathsf{MP}(\mathsf{S}(\chi))]\!]^{\mathcal{T}}_{s_0(s_1)^\omega} = 1 \;\Leftrightarrow\; s_0(s_1)^\omega \models \mathbf{FG}\chi, \tag{54}$$

$$[\![\mathsf{MP}(\mathsf{B}^k(\mathbf{G}\varphi))]\!]^{\mathcal{T}}_{s_0(s_1)^\omega} = 0 \;\Rightarrow\; s_0(s_1)^\omega \models \mathbf{FG}\varphi. \tag{55}$$

Equation (54) [resp., Eqs. (52) and (55)] justify using $\mathsf{S}(\chi)$ (resp., $\mathsf{B}^b(\mathbf{G}\varphi)$) to make the best effort to satisfy $\mathbf{G}\chi$ (resp., $\mathbf{G}\varphi$), especially in the synthesis of memoryless reactive systems.

3.3 Interpretation

A mean-payoff objective can be constructed from scratch. However, if a conjunction of must LTL specifications is unrealizable, some of them should be downgraded to desirable specifications. In this subsection, we show how to interpret these desirable LTL specifications in a mean-payoff objective. This interpretation may be conducted along with refining the other

must LTL specifications. However, for simplicity, we only consider interpretations without the refinement and assume that all desirable LTL specifications are **G**-formulae.

The set of desirable LTL specifications $\mathbf{G}\varphi_1, \ldots, \mathbf{G}\varphi_n$ can be reasonably interpreted as the mean-payoff objective

$$\mathsf{MP}\left(\sum_{1\leq i\leq n} t_i\right), \tag{56}$$

where each t_i is a payoff term interpreted from the desirable LTL specification $\mathbf{G}\varphi_i$. However, there are many ways of interpreting $\mathbf{G}\varphi_i$ in t_i. We now address the interpretations, with and without considering the structure and intention of the sub-formulae.

3.3.1 General cases

We first address how to interpret each desirable LTL specification $\mathbf{G}\varphi_i$, without considering the detailed structure and intention of its sub-formula φ_i. We provide naive interpretation guidelines, however, the details of the interpretation should be specified based on desirability.

Equation (54) implies that, when φ_i is bounded, $\mathbf{G}\varphi_i$ is naturally interpreted as a payoff term

$$c_i \cdot \mathsf{S}(\varphi_i), \tag{57}$$

where c_i is a weight depending on the priority of $\mathbf{G}\varphi_i$. Maximizing $\mathsf{MP}(\mathsf{S}(\varphi_i))$ corresponds strictly to maximizing the number of steps satisfying φ_i.

We can also reasonably interpret $\mathbf{G}\varphi_i$ for bounded φ_i as the payoff term

$$c_i \cdot \mathsf{B}(\mathbf{G}\varphi_i). \tag{58}$$

Maximizing $\mathsf{MP}(\mathsf{B}(\mathbf{G}\varphi_i))$ corresponds to maximizing the number of steps satisfying φ_i if almost all of the distances of steps violating φ_i are larger than the depth of the next-operators in φ_i.

From the equivalence between $\mathbf{G}\varphi_i$ and $\mathbf{GG}\varphi_i$, when φ_i is unbounded, $\mathbf{G}\varphi_i$ is reasonably interpreted as a payoff term

$$c_i \cdot \mathsf{B}^{b_i}(\mathbf{G}\varphi_i), \tag{59}$$

where $b_i \in \mathbb{N}$ is a bound. If φ_i is safe (resp., unsafe), maximizing $\mathsf{MP}(\mathsf{B}^{b_i}(\mathbf{G}\varphi_i))$ roughly means trying to avoid violating $\mathbf{G}\varphi_i$ (resp., the under-approximation property of φ_i) at each step to the extent possible. If φ_i is unsafe, larger b_i is better. One of the reasons is that Eq. (37), which determines a larger bound for UCWA, gives a tighter approximation. Another is the equivalence between $\mathbf{F}\varphi$ ($\equiv \neg\mathbf{G}\neg\varphi$) and $\varphi \vee \mathbf{XF}\varphi$, which is one of the basic relations for standard techniques in LTL-to-NBWA translation, such as tableau methods [41]. Therefore, if an LTL-to-UCWA translator employs such a method without unusual tunings, it produces a UCWA with an initial state from which rejecting states are no easier to reach than any non-rejecting state. If $\mathcal{T}$ is such a translator, we have

$$[\![\mathsf{MP}(\mathsf{B}^{b_i}(\mathbf{G}\varphi_i))]\!]_s^{\mathcal{T}} \leq [\![\mathsf{MP}(\mathsf{B}^{b_i+1}(\mathbf{G}\varphi_i))]\!]_s^{\mathcal{T}} \tag{60}$$

for any word $s \in (2^{AP})^\omega$, because $\mathcal{T}^{b_i}(\mathbf{G}\varphi_i)$ has the same state space and transition relation as $\mathcal{T}^{b_i+1}(\mathbf{G}\varphi_i)$.

Additionally, Eq. (55) suggests that additional payoff terms $-\mathsf{B}^{b'_i}(\mathbf{G}\neg\varphi_i)$, $\mathsf{B}^{b'_i}(\mathbf{GF}\varphi_i)$, and $-\mathsf{B}^{b'_i}(\mathbf{GF}\neg\varphi_i)$ are also helpful for evaluating the effort made to satisfy $\mathbf{G}\varphi_i$. This is because Eq. (55) implies

$$[\![\mathsf{MP}(-\mathsf{B}^{b'_i}(\mathbf{G}\neg\varphi_i))]\!]^{\mathcal{T}}_{s_0(s_1)^\omega} = 0 \;\Rightarrow\; s_0(s_1)^\omega \not\models \mathbf{GF}\varphi_i, \tag{61}$$

$$[\![\mathsf{MP}(\mathsf{B}^{b'_i}(\mathbf{GF}\varphi_i))]\!]^{\mathcal{T}}_{s_0(s_1)^\omega} = 0 \;\Rightarrow\; s_0(s_1)^\omega \models \mathbf{GF}\varphi_i, \tag{62}$$

$$[\![\mathsf{MP}(-\mathsf{B}^{b'_i}(\mathbf{GF}\neg\varphi_i))]\!]^{\mathcal{T}}_{s_0(s_1)^\omega} = 0 \;\Rightarrow\; s_0(s_1)^\omega \not\models \mathbf{FG}\varphi_i, \tag{63}$$

for a finite word $s_0 \in (2^{AP})^*$ and a nonempty finite word $s_1 \in (2^{AP})^+$. Therefore, $\mathbf{G}\varphi_i$ is reasonably interpreted as another payoff term

$$c_i \cdot \mathsf{B}^{b_i}(\mathbf{G}\varphi_i) + \sum_{1\leq j\leq m} c_{i,j} \cdot t_{i,j}, \tag{64}$$

where each $c_{i,j}$ is a weight according to the importance of $t_{i,j}$, and each $t_{i,j}$ is a payoff term of the form $-\mathsf{B}^{b_{i,j}}(\mathbf{G}\neg\varphi_i)$, $\mathsf{B}^{b_{i,j}}(\mathbf{GF}\varphi_i)$ or $-\mathsf{B}^{b_{i,j}}(\mathbf{GF}\neg\varphi_i)$. Note that $s_0(s_1)^\omega$ satisfies $\mathbf{FG}\varphi_i$ (resp., $\mathbf{GF}\varphi_i$), if the mean-payoff objective $\mathsf{MP}(\mathsf{B}^{b'_i}(\mathbf{G}\varphi_i))$ (resp., $\mathsf{MP}(\mathsf{B}^{b'_i}(\mathbf{GF}\varphi_i))$) is equal to 0 for $s_0(s_1)^\omega$ (i.e., fully maximized by $s_0(s_1)^\omega$). However, $s_0(s_1)^\omega$ may violate $\mathbf{GF}\varphi_i$ (resp., $\mathbf{FG}\varphi_i$) even if $\mathsf{MP}(-\mathsf{B}^{b'_i}(\mathbf{G}\neg\varphi_i))$ (resp., $\mathsf{MP}(-\mathsf{B}^{b'_i}(\mathbf{GF}\neg\varphi_i))$) is greater than 0 for $s_0(s_1)^\omega$ (i.e., not fully minimized by $s_0(s_1)^\omega$).

An alternative interpretation approach is the use of a formula-based approximation on φ_i rather than an automata-based approximation via UCWA. The simplest formula-based approximation is to bound until-operators in φ_i. Let m be the number of unbounded until- and release- operators in φ_i. Consider an LTL formula φ'_i obtained by replacing every occurrence of the respective unbounded until- and release- operators in φ_i by bounded until- and release- operators $\mathbf{U}^{\leq l_j}$ and $\mathbf{R}^{\leq l_{j'}}$ ($j, j' \in \{1, \ldots, m\}$). φ'_i is bounded, so that φ_i is reasonably interpreted as a payoff term

$$c_i \cdot \mathsf{S}(\varphi'_i), \tag{65}$$

Note that φ'_i draws closer to φ_i when bounds $l_1, \ldots, l_m$ become larger; however, φ'_i may be neither an under- nor an over-approximation of φ_i.

Another formula-based approximation is to "safetyize" φ_i. Let m' be the number of unbounded until-operators in an NNF formula $nnf(\varphi_i)$. Consider an LTL formula φ''_i obtained by replacing every occurrence of unbounded until-operator in $nnf(\varphi_i)$ by bounded until-operator $\mathbf{U}^{\leq l'_j}$ ($1 \leq j \leq m'$). φ''_i is safe, so that φ_i is also reasonably interpreted as a payoff term

$$c_i \cdot \mathsf{B}(\mathbf{G}\varphi''_i), \tag{66}$$

Note that φ''_i is an under-approximation of φ_i and becomes closer to φ_i when bounds $l'_1, \ldots, l'_{m'}$ become larger.

The above is briefly summarized as follows:

- Maximizing the mean-payoff for the S-term given by Eq. (57) [resp., Eq. (65)] corresponds to making an effort to satisfy φ_i (resp., the approximated property of φ_i) at each step.
- Maximizing the mean-payoff for the B-term given by Eq. (58) [resp., Eqs. (59), (64) or (66)] corresponds to making an effort to avoid violating $\mathbf{G}\varphi_i$ (resp., the under-approximated property of $\mathbf{G}\varphi_i$) at each step.

Each weight and bound in Eqs. (57)–(59) and (64)–(66) should depend on the priority order of the desirable specifications, the types of terms, and the designer's preferences regarding the behavior. In particular, for a term $c_i \cdot \mathsf{B}^{b_i}(\mathbf{G}\varphi_i)$, it will be preferable that the values of c_i and b_i are chosen based on the characteristics of the UCWA $\mathcal{T}(\mathbf{G}\varphi_i)$ constructed in advance.

3.3.2 Special cases

We now discuss the interpretation of each desirable LTL specification $\mathbf{G}\varphi_i$, considering the detailed structure and intention of its sub-formula φ_i. In this type of approach, the designer should provide details of the interpretation based on the desirability and applications. We only give a simple example of the interpretation. Consider the following desirable LTL specification as an example.

$$\varphi_1 = \mathbf{G}(req \rightarrow \mathbf{X}(\mathbf{F}res)), \tag{67}$$

where $req \in IAP$ and $res \in OAP$ are atomic propositions for input and output, respectively. φ_1 means that "for each occurrence of a request req, a system must eventually return a response res". When using this type of formula, it is often expected that "a system responds *within a certain time* (or rather, *as soon as possible*)".

Considering this implicit requirement "within a certain time", φ_1 can be interpreted as a mean-payoff objective such as

$$\mathsf{MP}\left(\mathsf{S}\left(req \rightarrow \mathbf{X}\left(\mathbf{F}^{\leq l} res\right)\right)\right), \tag{68}$$

by giving bound l for the $\mathbf{F}$-operator. This mean-payoff objective intuitively represents a desirable specification: "for each occurrence of a request req, a system tries to return a response res within the next l-steps to the extent possible". To capture the intention "as soon as possible", for example, φ_1 may be interpreted as follows:

$$\mathsf{MP}\left(\sum_{1 \leq i \leq n} \mathsf{S}\left(req \rightarrow \mathbf{X}\left(\mathbf{F}^{\leq l_i} res\right)\right)\right). \tag{69}$$

In this objective, the response time is evaluated in some stages.

Alternatively, φ_1 can also be interpreted as another mean-payoff objective.[7]

$$\mathsf{MP}(c_b \cdot \mathsf{B}(\mathbf{G}(req \rightarrow \mathbf{X}(wait \mathbf{W} res))) - c_s \cdot \mathsf{S}(wait)), \tag{70}$$

where $wait \in OAP$ is a fresh atomic proposition for output, and c_b and c_s are positive weights that depend on the importance of the respective terms. The $c_b \cdot \mathsf{B}(\mathbf{G}(req \rightarrow \mathbf{X}(wait \mathbf{W} res)))$ part of the payoff term means that "a penalty $-c_b$ is given for each occurrence of minimal bad prefixes of $\mathcal{L}(\mathbf{G}(req \rightarrow \mathbf{X}(wait \mathbf{W} res)))$", where $\mathbf{G}(req \rightarrow \mathbf{X}(wait \mathbf{W} res))$ is a weakened property of $\mathbf{G}(req \rightarrow \mathbf{X}(wait \mathbf{U} res))$ that does not guarantee the response res. The $-c_s \cdot \mathsf{S}(wait)$ part means that "a penalty $-c_s$ is given for each occurrence of $wait$". Therefore, this mean-payoff objective intuitively represents a desirable specification: "for each occurrence of a request req, a system basically tries to return a response res as soon as possible; however, the system is allowed to neglect the request if returning it quickly is difficult". More precisely, if the system cannot return a response for a request within the next $\lceil c_b/c_s \rceil$-steps, it will ignore the request.

[7] In an advanced interpretation, we can use $\mathbf{G}(req \rightarrow \mathbf{X}(wait \mathbf{W} res))$ as an additional must specification and $\mathsf{MP}(-c_s \cdot \mathsf{S}(wait))$ as a mean-payoff objective.

In this type of interpretation, it is sometimes possible to naturally formalize requirements, e.g. "as soon as possible" as above, that are difficult (or impossible) to express in LTL.

3.4 Assumptions

A kind of soft assumption which may be violated by the environment can be included in our mean-payoff objectives.

Consider a desirable LTL specification $\varphi_{Asmp} \rightarrow \varphi_{Grnt}$ with the assumption-guarantee form, where φ_{Asmp} signifies a property that is assumed to be followed by the environment, and φ_{Grnt} signifies a property that should be guaranteed by a system. A normal LTL synthesis method may involve a system that does not guarantee φ_{Grnt} when φ_{Asmp} does not hold. However, it is often required that a system is *robust* [5]. Informally, a robust system produces a small number of system errors for a small number of environment errors. Let φ_{Asmp} and φ_{Grnt} be respective **G**-formulae $\mathbf{G}\varphi_1$ and $\mathbf{G}\varphi_2$. According to our basic idea, a more robust reactive system should produce a smaller difference in the number of steps violating φ_2 from the number of steps violating φ_1. Therefore, the desirable LTL specification can be interpreted as the mean-payoff objective

$$\mathsf{MP}(t_{Grnt} - t_{Asmp}), \tag{71}$$

when t_{Grnt} (resp., t_{Asmp}) is a certain term interpreted from φ_{Grnt} (resp., φ_{Asmp}) according to the ideas in Sect. 3.3.

4 Our synthesis method

In this section, we propose a method for synthesizing a reactive system that realizes all given must specifications and endeavors to satisfy the desirable specifications.

Our problem is formalized to synthesizing a reactive system $\mathcal{R}^{opt}$ that realizes a given must LTL specification φ and is optimal for a given mean-payoff objective $\mathsf{MP}(t)$, which represents the weighted desirable specifications. Note that φ is a non-restricted LTL formula, i.e. it may be an assumption-guarantee type formula $\varphi_1 \rightarrow \varphi_2$. We assume that an LTL-to-UCWA mapping $\mathcal{T}$ is fixed. That is,

$$\mathcal{R}^{opt} = \arg\sup_{\mathcal{R}} \left\{ c \;\middle|\; \forall s \in \mathit{Intr}(\mathcal{R}) \left(s \models \varphi \text{ and } [\![\mathsf{MP}(t)]\!]_s^{\mathcal{T}} \geq c \right) \right\} \tag{72}$$

$\mathsf{MP}(t)$ is described from scratch or is interpreted from desirable LTL specifications using the ideas in Sect. 3.3. Note that, in this formalization, payoff term t may include expressions related to performance requirements.

This problem can be strictly reduced to finding an optimal (or ε-optimal) strategy on a *mean-payoff parity game* [13]. However, this reduction generally requires a deterministic ω-regular automaton, which is very difficult to attain. Furthermore, the algorithm in [13] that solves the mean-payoff parity game is also very complicated, and the optimal (resp., ε-optimal) strategy generally requires infinite (resp., large) memory.

In considering the trade-off among computational cost, size (and finite memory), and optimality of the resulting reactive systems, we propose using the Safraless approach (Lines 1–10 in Algorithm 1), to obtain a winning region $\mathcal{W}^{\varphi,k}$ from φ. We then compose a locally optimal reactive system for $\mathsf{MP}(t)$ from the set of systems given as winning strategies in the winning region $\mathcal{W}^{\varphi,k}$. This is a generalized method of the one in [26], which focuses

on must LTL specifications with the assumption-guarantee form and mean-payoff objectives (interpreted from its guarantee part) without B-terms.

4.1 Outline

The latter part of our method is given in Algorithm 2. Let $\mathsf{MP}(t)$ be a given mean-payoff objective, where $t = \sum_{1 \leq i \leq n} c_i \cdot t_i$ and each t_i is either $\mathsf{S}(\chi_i)$ or $\mathsf{B}^{b_i}(\varphi_i)$. First, for each atomic term t_i (either S-term or B-term), we construct a safety game from t_i by reusing the procedures `LTL2UCWA_translation`[8] and `BUCWA2SG_transformation` in Algorithm 1 (either Lines 3–9 or 12–13), where `LTL2UCWA_translation` implements $\mathcal{T}$. It is then transformed into a mean-payoff game $\mathcal{M}^{t_i}$ (using either procedure `SG2MPG_transformation_S` at Line 10 or `SG2MPG_transformation_B` at Line 14). For each game $\mathcal{M}^{t_i}$, its transition functions are total, and an outcome of any play ρ equals $[\![\mathsf{MP}(t_i)]\!]^{\mathcal{T}}_{trace(\rho)}/2$. Second, we construct a weighted synchronized product of an input winning region $\mathcal{W}^{\varphi,k}$ and the mean-payoff games $\mathcal{M}^{t_1}, \ldots, \mathcal{M}^{t_n}$ (the procedure `weighted_synchronized_product` at Line 17). For the synchronized product mean-payoff game $\mathcal{M}^{\varphi,k,t}$, the outcome of a play ρ is $[\![\mathsf{MP}(t)]\!]^{\mathcal{T}}_{trace(\rho)}/2$. Finally, we find an optimal strategy ν for the system-side player (Player 0) on $\mathcal{M}^{\varphi,k,t}$ using the standard technique for solving mean-payoff games [12,20] (the procedure `optimal_strategy`, Line 18).

4.2 Mean-payoff games for simple mean-payoff objectives

In this subsection, we explain how to construct a mean-payoff game $\mathcal{M}^t$ from an atomic term $t \in \{\mathsf{S}(\chi), \mathsf{B}^k(\varphi)\}$, i.e., Lines 3–10 and 12–14 in Algorithm 2.

4.2.1 Mean-payoff game for MP(S(χ))

Let φ' be a safe LTL formula $\mathbf{G}(\chi \leftrightarrow \mathbf{X}^{\lfloor depth(\chi) \rfloor} \hat{p})$ for a bounded LTL formula χ, where $\hat{p}$ is a fresh atomic proposition, and $depth(\chi)$ is the depth of the next operators in χ considering (system-side and environment-side) turns in the games.

$$depth(\chi) = \begin{cases} 0 & \text{if } \chi \in Bool(OAP), \\ 1/2 & \text{if } \chi \in Bool(AP) \backslash Bool(OAP), \\ \min\{depth(\chi_1), depth(\chi_2)\} & \text{if } \chi = \chi_1 \vee \chi_2 \text{ or } \chi = \chi_1 \wedge \chi_2, \\ depth(\chi_1) + 1 & \text{if } \chi = \mathsf{X}\chi_1. \end{cases} \tag{73}$$

If $depth(\chi) \in \mathbb{N}$ (resp., $depth(\chi) \notin \mathbb{N}$), we can determine if χ holds at a step in a system-side (resp., environment-side) turn after $\lfloor depth(\chi) \rfloor$-steps. φ' implies that $\hat{p}$ is associated with χ at a $\lfloor depth(\chi) \rfloor$-step delay. φ' is safe and hence can be translated into a USWA $\mathcal{A}^{\varphi'}$ $(= \mathcal{T}(\varphi'))$ by the procedure `LTL2UCWA_translation`, implementing $\mathcal{T}$, at Line 4 in Algorithm 2. In the transformation from $\mathcal{A}^{\varphi'}$ into a mean-payoff game $\mathcal{M}^{\mathsf{S}(\chi)}$ for $\mathsf{MP}(\mathsf{S}(\chi))$, we use transitions labeled $\hat{p}$ as guides to set payoffs.

Consider $depth(\chi) \in \mathbb{N}$, implying that the deepest Boolean formula includes no input proposition. In this case, we treat $\hat{p}$ as an output proposition. Let $\mathcal{S}$ be a safety game $\langle\langle V_0, V_1, 2^{OAP \cup \{\hat{p}\}}, 2^{IAP}, E_0, E_1\rangle, v_{init}, S\rangle$, constructed from $\mathcal{A}^{\varphi'}$ using the procedure `BUCWA2SG_transformation` at Line 6 in Algorithm 2. Note that both E_0 and E_1 are

[8] We reuse the LTL-to-automata translator in Algorithm 1 for simplicity. However, we can use another translator in Algorithm 2.

Algorithm 2 The latter part of our synthesis method

Input: A winning region $\mathcal{W}^{\varphi,k}$ extracted using Lines 1–11 in Algorithm 1, and an objective term $t = \sum_{1 \leq i \leq n} c_i \cdot t_i$ where $t_i \in \{\mathsf{S}(\chi_i), \mathsf{B}^{b_i}(\varphi_i)\}$

Output: A reactive system that realizes φ and is optimal for $\mathsf{MP}(t)$ on $\mathcal{W}^{\varphi,k}$, as a memoryless strategy ν on a mean-payoff game $\mathcal{M}^{\varphi,k,t}$, where an LTL-to-UCWA mapping $\mathcal{T}$ is implemented by `LTL2UCWA_translation`.

1: **for** $i = 0$ to n **do**
2: **if** $t_i = \mathsf{S}(\chi_i)$ **then**
3: $\varphi_i' := \mathbf{G}(\chi_i \leftrightarrow \mathbf{X}^{\lfloor depth(\chi_i) \rfloor} \hat{p})$ // $\hat{p}$ is a fresh proposition to bind with χ_i on $\lfloor depth(\chi_i) \rfloor$-step delay
4: $\mathcal{A}^{\varphi_i'} :=$ `LTL2UCWA_translation`(φ_i') // $\mathcal{A}^{\varphi_i'}$ can be seen as a USWA because φ_i' is safe.
5: **if** $depth(\chi_i) \in \mathbb{N}$ **then**
6: $\mathcal{S}^{\varphi_i'} :=$ `BUCWA2SG_transformation`$(\mathcal{A}^{\varphi_i'}, 0, OAP \cup \{\hat{p}\}, IAP)$
7: **else**
8: $\mathcal{S}^{\varphi_i'} :=$ `BUCWA2SG_transformation`$(\mathcal{A}^{\varphi_i'}, 0, OAP, IAP \cup \{\hat{p}\})$
9: **end if**
10: $\mathcal{M}^{t_i} :=$ `SG2MPG_transformation_S`$(\mathcal{S}^{\varphi_i'}, \hat{p})$
// $\mathcal{M}^{t_i}$ has total transition functions for the environment-side player
// and a weighting function that depends on $\mathsf{S}(\chi_i)$
11: **else if** $t_i = \mathsf{B}^{b_i}(\varphi_i)$ **then**
12: $\mathcal{A}^{\varphi_i} :=$ `LTL2UCWA_translation`(φ_i)
13: $\mathcal{S}^{\varphi_i, b_i} :=$ `BUCWA2SG_transformation`$(\mathcal{A}^{\varphi_i}, b_i, OAP, IAP)$
14: $\mathcal{M}^{t_i} :=$ `SG2MPG_transformation_B`$(\mathcal{S}^{\varphi_i, b_i})$
// $\mathcal{M}^{t_i}$ has total transition functions for the environment-side player
// and a weighting function that depends on $\mathsf{B}^{b_i}(\varphi_i)$
15: **end if**
16: **end for**
17: $\mathcal{M}^{\varphi,k,t} :=$ `weighted_synchronized_product`$(\mathcal{W}^{\varphi,k}, \mathcal{M}^{t_0}, \ldots, \mathcal{M}^{t_n}, c_0, \ldots, c_n)$
// $\mathcal{M}^{\varphi,k,t}$ has an extended arena of $\mathcal{W}^{\varphi,k}$ and a weighting function that depends on t
18: $\nu :=$ `optimal_strategy`$(\mathcal{M}^{\varphi,k,t})$
19: **return** $\langle \nu, \mathcal{M}^{\varphi,k,t} \rangle$

total. The mean-payoff game $\mathcal{M}^{\mathsf{S}(\chi)} = \langle\langle V_0 \cap S, V_1 \cap S, 2^{OAP}, 2^{IAP}, E_0', E_1' \rangle, v_{init}, W \rangle$ constructed by the procedure `SG2MPG_transformation_S` at Line 10 in Algorithm 2 is

$$E_0'(v, \alpha^O) = \begin{cases} E_0(v, \alpha^O \cup \{\hat{p}\}) & \text{if } E_0(v, \alpha^O \cup \{\hat{p}\}) \in S, \\ E_0(v, \alpha^O) & \text{otherwise,} \end{cases} \tag{74}$$

$$E_1'(v, \alpha^I) = E_1(v, \alpha^I), \tag{75}$$

$$W(e) = \begin{cases} 1 & \text{if } E_0(pred(e), act(e) \cup \{\hat{p}\}) = E_0'(pred(e), act(e)), \\ 0 & \text{otherwise.} \end{cases} \tag{76}$$

φ' is trivially realizable because a reactive system can easily determine an output signal, either $\hat{p}$ or $\neg\hat{p}$, that satisfies it at each step using information from the past $depth(\chi)$ steps. This fact implies that E_1' is total because, for any $v \in V_1 \cap S$ and $\alpha^I \in 2^{IAP}$, its successor $E_1(v, \alpha)$ is in S. Additionally, for all $v \in V_0 \cap S$ and $\alpha^O \in 2^{OAP}$, either $E_0(v, \alpha^O)$ or $E_0(v, \alpha^O \cup \{\hat{p}\})$ is in S. Therefore, E_0' is also total. For any play $\rho = e_0 e_1 \ldots \in Play(\mathcal{M}^{\mathsf{S}(\chi)})$ and its corresponding interaction $trace(\rho) \in (2^{AP})^\omega$, we have

$$\langle\!\langle \mathsf{S}(\chi) \rangle\!\rangle^{\mathcal{T}}_{trace(\rho)}(i + depth(\chi)) = W(e_{2i}) + W(e_{2i+1}), \text{ for each } i \in \mathbb{N}, \tag{77}$$

$$[\![\mathsf{MP}(\mathsf{S}(\chi))]\!]^{\mathcal{T}}_{trace(\rho)} = 2 \cdot outcome^W(\rho). \tag{78}$$

Consider $depth(\chi) \notin \mathbb{N}$, implying that the deepest Boolean formula includes at least one input proposition. In this case, we treat $\hat{p}$ as an input proposition. Let $\mathcal{S}$ be a safety game $\langle\langle V_0, V_1, 2^{OAP}, 2^{IAP \cup \{\hat{p}\}}, E_0, E_1\rangle, v_{init}, S\rangle$, constructed from $\mathcal{A}^{\varphi'}$ using `BUCWA2SG_transformation` at Line 8 in Algorithm 2. The mean-payoff game $\mathcal{M}^{\mathsf{S}(\chi)} = \langle\langle V_0 \cap S, V_1 \cap S, 2^{OAP}, 2^{IAP}, E_0'', E_1''\rangle, v_{init}, W'\rangle$ constructed by the procedure `SG2MPG_transformation_S` at Line 10 in Algorithm 2 is

$$E_0''(v, \alpha^O) = E_0(v, \alpha^O), \tag{79}$$

$$E_1''(v, \alpha^I) = \begin{cases} E_1(v, \alpha^I \cup \{\hat{p}\}) & \text{if } E_1(v, \alpha^I \cup \{\hat{p}\}) \in S, \\ E_1(v, \alpha^I) & \text{otherwise,} \end{cases} \tag{80}$$

$$W'(e) = \begin{cases} 1 & \text{if } E_1(pred(e), act(e) \cup \{\hat{p}\}) = E_1''(pred(e), act(e)), \\ 0 & \text{otherwise.} \end{cases} \tag{81}$$

In this case, φ' is trivially unrealizable because the environment can easily determine an input signal, either $\hat{p}$ or $\neg\hat{p}$, that violates it at each step, using information from the past $\lfloor depth(\chi) \rfloor$ steps. This fact implies that E_0'' is total because, for any $v \in V_0 \cap S$ and $\alpha^O \in 2^{OAP}$, its successor $E_0(v, \alpha^O)$ is in S. Additionally, for all $v \in V_1 \cap S$ and $\alpha^I \in 2^{IAP}$, either $E_1(v, \alpha^I)$ or $E_1(v, \alpha^I \cup \{\hat{p}\})$ is in S. Therefore, E_1'' is also total. For any play $\rho = e_0 e_1 \ldots \in Play(\mathcal{M}^{\mathsf{S}(\chi)})$ and its corresponding interaction $trace(\rho) \in (2^{AP})^\omega$, we have

$$\langle\!\langle \mathsf{S}(\chi) \rangle\!\rangle^{\mathcal{T}}_{trace(\rho)}(i + \lfloor depth(\chi) \rfloor) = W'(e_{2i}) + W'(e_{2i+1}), \text{ for each } i \in \mathbb{N}, \tag{82}$$

$$[\![\mathsf{MP}(\mathsf{S}(\chi))]\!]^{\mathcal{T}}_{trace(\rho)} = 2 \cdot outcome^{W'}(\rho). \tag{83}$$

4.2.2 Mean-payoff game for $MP(B^b(\varphi))$

For a term $\mathsf{MP}(\mathsf{B}^b(\varphi))$, we can obtain a UCWA $\mathcal{A}^\varphi$ $(= \mathcal{T}(\varphi))$ by the procedure `LTL2UCWA_translation` at Line 12 in Algorithm 2. On a determinized safety word automaton of $\mathcal{T}^b(\varphi)$ (resp., in $\mathcal{A}^\varphi$), reaching (resp., more than b-times visiting) rejecting states on a run means a minimal bad prefix of $\mathcal{L}(\mathcal{T}^b(\varphi))$ has occurred in its word. In the transformation from $\mathcal{A}^\varphi$ with bound b into a mean-payoff game $\mathcal{M}^{\mathsf{B}^b(\varphi)}$ for $\mathsf{MP}(\mathsf{B}^b(\varphi))$, we use rejecting states as guides to set payoffs.

Let $\mathcal{S}$ be a safety game $\langle\langle V_0, V_1, 2^{OAP}, 2^{IAP}, E_0, E_1\rangle, v_{init}, S\rangle$, constructed from $\mathcal{A}^\varphi$ and bound b using `BUCWA2SG_transformation` at Line 13 in Algorithm 2. The mean-payoff game $\mathcal{M}^{\mathsf{B}^b(\varphi)} = \langle\langle V_0, V_1 \cup \{v_{mid}\}, 2^{OAP}, 2^{IAP}, E_0''', E_1'''\rangle, v_{init}, W''\rangle$ constructed by the procedure `SG2MPG_transformation_B` at Line 14 in Algorithm 2 is

$$E_0'''(v, \alpha^O) = \begin{cases} v_{mid} & \text{if } E_0(v, \alpha^O) \notin S, \\ E_0(v, \alpha^O) & \text{otherwise,} \end{cases} \tag{84}$$

$$E_1'''(v, \alpha^I) = \begin{cases} v_{init} & \text{if } v = v_{mid} \text{ or } E_1(v, \alpha^I) \notin S, \\ E_1(v, \alpha^I) & \text{otherwise,} \end{cases} \tag{85}$$

$$W''(e) = \begin{cases} -1 & \text{if } pred(e) = v_{mid} \text{ or } E_1(pred(e), act(e)) \notin S, \\ 0 & \text{otherwise.} \end{cases} \tag{86}$$

Both E_0''' and E_1''' are total because E_0 and E_1 are total. From the definition of W'', for a minimal bad prefix $(\alpha_0^O \cup \alpha_0^I) \cdots (\alpha_n^O \cup \alpha_n^I) \in (2^{AP})^+$ of $\mathcal{L}(\mathcal{T}^b(\varphi))$ and the corresponding

prefix $\langle v_0, \alpha_0^O, v_1\rangle\langle v_1, \alpha_0^I, v_2\rangle \cdots \langle v_{2n}, \alpha_n^O, v_{2n+1}\rangle\langle v_{2n+1}, \alpha_n^I, v_{2(n+1)}\rangle \in (E_0'''E_1''')^+$ of any play,

$$W''(\langle v_{2i}, \alpha_i^O, v_{2i+1}\rangle) + W''(\langle v_{2i+1}, \alpha_i^I, v_{2(i+1)}\rangle) = 0, \quad \text{for each } i < n, \tag{87}$$

$$W''(\langle v_{2n}, \alpha_n^O, v_{2n+1}\rangle) + W''(\langle v_{2n+1}, \alpha_n^I, v_{2(n+1)}\rangle) = -1. \tag{88}$$

Therefore, for any play $\rho = e_0e_1 \ldots \in Play(\mathcal{M}^{\mathsf{B}^b(\varphi)})$ and its corresponding interaction $trace(\rho) \in (2^{AP})^\omega$, we have

$$\langle\!\langle \mathsf{B}^b(\varphi)\rangle\!\rangle^{\mathcal{T}}_{trace(\rho)}(i) = W''(e_{2i}) + W''(e_{2i+1}), \quad \text{for each } i \in \mathbb{N}, \tag{89}$$

$$[\![\mathsf{MP}(\mathsf{B}^b(\varphi))]\!]^{\mathcal{T}}_{trace(\rho)} = 2 \cdot outcome^{W''}(\rho). \tag{90}$$

4.3 Weighted synchronized product

Let $\mathcal{W}^{\varphi,k}$ be a winning region $\langle\langle V_0^0, V_1^0, 2^{OAP}, 2^{IAP}, E_0^0, E_1^0\rangle, v_{init}^0, V_0^0 \cup V_1^0\rangle$ of an input to Algorithm 2. For payoff term $t = \sum_{1\le i\le n} c_i \cdot t_i$, let $\mathcal{M}^{t_i}$ be a mean-payoff game $\langle\langle V_0^i, V_1^i, 2^{OAP}, 2^{IAP}, E_0^i, E_1^i\rangle, v_{init}^i, W^i\rangle$, constructed from t_i using either Line 10 or Line 14 in Algorithm 2.

The mean-payoff game $\mathcal{M}^{\varphi,k,t} = \langle\langle V_0^0 \times V_0^1 \times \cdots \times V_0^n, V_1^0 \times V_1^1 \times \cdots \times V_1^n, 2^{OAP}, 2^{IAP}, E_0'''', E_1''''\rangle, \langle v_{init}^0, v_{init}^1, \ldots, v_{init}^n\rangle, W''''\rangle$ constructed from them by the procedure `weighted_synchronized_product` at Line 17 in Algorithm 2 is, for $\sigma \in \{0, 1\}$,

$$E_\sigma''''(\langle v^0, v^1, \ldots, v^n\rangle, \alpha) = \begin{cases} \text{undefined} & \text{if } \sigma = 0 \text{ and } E_0^0(v^0, \alpha) \text{ is undefined,} \\ \langle E_\sigma^0(v^0, \alpha), E_\sigma^1(v^1, \alpha), \ldots, E_\sigma^n(v^n, \alpha)\rangle & \text{otherwise,} \end{cases} \tag{91}$$

$$W''''(\langle v^0, v^1, \ldots, v^n\rangle, \alpha, \langle E_\sigma^0(v^0, \alpha), E_\sigma^1(v^1, \alpha), \ldots, E_\sigma^n(v^n, \alpha)\rangle) = \sum_{i=0}^{n} c_i \cdot W^i(\langle v^i, \alpha, E(v^i, \alpha)\rangle). \tag{92}$$

From Eqs. (78), (83), and (90), for any play $\rho \in Play(\mathcal{M}^{\varphi,k,t})$ and its corresponding interaction $trace(\rho) \in (2^{AP})^\omega$, we have

$$[\![\mathsf{MP}(t)]\!]^{\mathcal{T}}_{trace(\rho)} = 2 \cdot outcome^{W''''}(\rho). \tag{93}$$

Additionally, because $E_0^1, E_1^1, \ldots, E_0^n, E_1^n$ are total, we have

$$\forall \rho \in Play(\mathcal{M}^{\varphi,k,t}), \exists \rho' \in Play(\mathcal{W}^{\varphi,k}) \left(trace(\rho) = trace(\rho')\right). \tag{94}$$

4.4 Correctness and optimality

We have the following theorem.

Theorem 1 (Correctness of Algorithm 2) *Algorithm 2 returns a reactive system that realizes φ and is optimal for mean-payoff objective $\mathsf{MP}(t)$ on winning region $\mathcal{W}^{\varphi,k}$.*

Proof From Eqs. (93) and (94), the optimal strategy on $\mathcal{M}^{\varphi,k,t}$ is optimal on $\mathcal{W}^{\varphi,k}$, and any play on $\mathcal{W}^{\varphi,k}$ satisfies φ. □

As a result, we can obtain a reactive system that realizes φ and is *locally* optimal for $\mathsf{MP}(t)$ in the sense of Eq. (72). Note that our method does not guarantee that the reactive system is *globally* optimal for $\mathsf{MP}(t)$ as a trade-off for the memory finiteness. This is because the must LTL specification φ is under-approximated in our method. The winning region $\mathcal{W}^{\varphi,k}$ derived from φ is based on the under-approximated safe property via UCWA with bound k, and we find the reactive system from $\mathcal{W}^{\varphi,k}$. There generally exists a gap between the original property φ and the under-approximated property. It can be a matter for any large k in our method, unlike in LTL realizability checking. Consider the simple case when a must LTL specification $\mathbf{GF}p$ and a mean-payoff objective $\mathsf{MP}(\mathsf{S}(\neg p))$, where p is an output proposition. A reactive system realizes $\mathbf{GF}p$ and is globally optimal for $\mathsf{MP}(\mathsf{S}(\neg p))$, i.e., its value is 1, if and only if the system outputs p infinitely often and the distance between p increases further and further. Our method cannot produce such reactive systems requiring infinite memory. $\mathbf{GF}p$ would be under-approximated into $\mathbf{GF}^k p$ via a naive LTL-to-UCWA translator. Therefore, our method employing the naive translation composes a reactive system outputting p at k-step intervals, i.e., its value of $\mathsf{MP}(\mathsf{S}(\neg p))$ is $(k-1)/k$. Any other translator leads to a similar result. The value approaches the global optimum arbitrarily using larger k in Algorithm 1.

Another concern in optimality regards bounds for B-terms in mean-payoff objectives. Our first idea is, for desirable specifications $\mathbf{G}\varphi_1, \ldots, \mathbf{G}\varphi_n$ with priority weights $c_1, \ldots, c_n$, to maximize each number of steps satisfying φ_i taking into account all priority weights, i.e., to maximize $\mathsf{MP}(\sum_{1\leq i\leq n} c_i \cdot \mathcal{S}(\varphi_i))$, where the $\mathcal{S}$-term is an extension of the S-term defined semantically for a non-restricted LTL formula ψ as follows.

$$\langle\!\langle \mathcal{S}(\psi) \rangle\!\rangle_s^{\mathcal{T}}(i) = \begin{cases} 1 & \text{if } \langle s, i\rangle \models \psi, \\ 0 & \text{otherwise.} \end{cases} \tag{95}$$

This objective cannot be directly represented in our mean-payoff objectives defined in Sect. 3. One question is whether there exist an LTL formula φ' and mean-payoff objective $\mathsf{MP}(t)$ such that, if a reactive system realizes φ' and maximizes $\mathsf{MP}(t)$, it also realizes the original must LTL specification φ and maximizes $\mathsf{MP}(\sum_{1\leq i\leq n} c_i \cdot \mathcal{S}(\varphi_i))$. If so (under some restrictions), another question is whether such t can be obtained as a term with a specific form, e.g., $\sum_{1\leq i\leq n} c_i \cdot \mathsf{B}^{b_i}(\mathbf{G}\varphi_i)$. These are open problems that are out of the scope of this paper.

4.5 Complexity

The time complexity of Algorithm 2 is doubly exponential, so it is in the same class as traditional LTL synthesis and realizability-checking [1,34,35].

For each atomic payoff term t_i with the form $\mathsf{S}(\chi_i)$, we can construct $\mathcal{M}^{t_i}$ with at most $2^{P_i \cdot \lfloor depth(\chi_i)\rfloor} \cdot (2^{|OAP|}+1)$ states, where P_i is the number of distinct atomic propositions in χ_i. χ_i can be translated into a deterministic safety automaton with at most $2^{P_i \cdot \lfloor depth(\chi_i)\rfloor}$ states. An m-state deterministic automaton can be transformed into a game with at most $m \cdot (2^{|OAP|}+1)$ states. For each atomic payoff term t_i with the form $\mathsf{B}^{b_i}(\varphi_i)$, we can construct a mean-payoff game $\mathcal{M}^{t_i}$ with at most $2^{2^{\mathcal{O}(|\varphi_i|)} \cdot \log(b_i+2)} \cdot (2^{|OAP|}+1)$ states, where $|\varphi_i|$ is the size of φ_i. φ_i (resp., $\neg\varphi_i$) can be translated into an equivalent UCWA (resp., NBWA) with at most $2^{\mathcal{O}(|\varphi_i|)}$ states [40]. An n_i-state b_i-UCWA can be transformed into an equivalent deterministic safety automaton with at most $2^{n_i \cdot \log(b_i+2)}$ states. We can construct the weighted synchronized product mean-payoff game from input winning region $\mathcal{W}^{\varphi,k}$ and mean-payoff objective $\mathsf{MP}(t)$, which has at most $|\mathcal{W}^{\varphi,k}| \cdot T$ states and a largest absolute weight less than or equal to the possible weights of t. Here, T is a product of the sizes of the mean-payoff games for atomic terms in t. The time complexity for solving an m-state l-transition mean-payoff

game is $\mathcal{O}(m^2 \cdot l \cdot d \cdot (\log m + \log d))$ [12], where d is the largest absolute value of the weights. The above discussion implies that the time complexity of Algorithm 2 is doubly exponential.

As mentioned in Sect. 4.4, there is a trade-off between the optimality of the resulting system for $\mathsf{MP}(t)$ and its computational cost. The size of $\mathcal{W}^{\varphi,k}$ (and the resulting system) grows polynomially in k. Therefore, the mean-payoff value of the resulting system approaches the global optimum arbitrary; however, its computational cost increases polynomially as k increases.

5 Experimental evaluation

To confirm that our algorithm produces preferable reactive systems, we implemented a prototype of our synthesis method and performed a simple experiment. We also demonstrate that our method can treat non-trivial scale instances. Additionally, we discuss the advantage of our method.

5.1 Implementation and experimental environment

The prototype is implemented in C++ and only supports payoff terms without B-terms. We can employ an existing LTL-to-NBWA translator, e.g., LTL2BA [23], SPOT [17] and LTL3BA [3], as an LTL-to-UCWA translator. The prototype uses LTL3BA for the LTL-to-UCWA mapping $\mathcal{T}$. A direction of the implementation is the same as the one in [26]. Each state of automata and games is dealt with explicitly whereas transitions from the state are represented by one multi-terminal *binary decision diagram* (BDD). We employ CUDD[9] for manipulating BDDs. On solving mean-payoff games (at Line 18 in Algorithm 2), we employ an iterative algorithm proposed in [42] and introduce a heuristic to check the optimality of a tentative solution after a certain number of iterations, which depends on the size of the games. Furthermore, for solving them efficiently, an abstracted mean-payoff game is constructed from the original game $\mathcal{M}^{\varphi,k,t}$. Actions of transitions are omitted in the abstracted game, and thus the state space of the abstracted game can be reduced. That is, its optimal strategy can be computed efficiently. From the optimal strategy and $\mathcal{M}^{\varphi,k,t}$, the optimal reactive system is composed as a transition system with the same format to Acacia+[10] [8–10] that is a well-known LTL synthesis tool.

The experiments were performed on a MacBook Pro (Retina, Mid 2012) with OS X Yosemite 10.10.5, 2.6-GHz CPU (Intel Core i7), and 16-GB memory (1600-MHz DDR3).

5.2 Experiments

Instance 1 First, we focus on the preferability of synthesized systems. Consider synthesizing a reactive system that realizes the nearest property of LTL formula $\varphi = \varphi_1 \wedge \varphi_2 \wedge \varphi_3$. This is a conjunction of the LTL formulae

$$\varphi_1 = \mathbf{G}(req_1 \rightarrow \mathbf{X} res), \tag{96}$$

$$\varphi_2 = \mathbf{G}(req_2 \rightarrow \mathbf{X} \neg res), \tag{97}$$

$$\varphi_3 = \mathbf{G}(res \leftrightarrow \mathbf{X} res), \tag{98}$$

[9] Available at http://vlsi.colorado.edu/~fabio/.

[10] Available at http://lit2.ulb.ac.be/acaciaplus/.

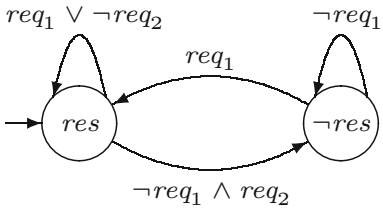

Fig. 4 A reactive system (Moore machine format) synthesized by the proposed method for Instance 1

where $req_1, req_2 \in IAP$ are atomic propositions for the input, and $res \in OAP$ is an atomic proposition for the output. Intuitively, φ_1 means that "a system must output res in the next step in response to an input req_1", and φ_2 means that "a system must output $\neg res$ in the next step in response to an input req_2". φ_3 means that "a system must maintain its current output in the next step", where φ_3 is a desirable specification derived from the performance requirement that "the output of a system should be maintained *if unnecessary*, to reduce signal switching". The non-necessity depends on other specifications, and it is impossible to express the meaning of "*if unnecessary*" in LTL. Therefore, φ_3 is a stronger property than this requirement. φ (or, more precisely, $\varphi_1 \wedge \varphi_2$) is unrealizable, because if both res_1 and res_2 are simultaneously inputs at a certain step, output res (resp., $\neg res$) in the next step violates φ_2 (resp., φ_1). We assume that a specification φ_i has a higher priority than another specification φ_j if $i < j$. In our approach, a must specification is an LTL formula φ_1, and desirable specifications φ_2 and φ_3 are transformed into a payoff term dependent on their priorities. The discussion in Sect. 3.3 implies that this payoff term is, for example,

$$t_{23} = 2 \cdot \mathsf{S}(req_2 \rightarrow \mathbf{X}\neg res) + \mathsf{S}(res \leftrightarrow \mathbf{X}res). \tag{99}$$

This term suggests that φ_2 has a higher priority than φ_3, because their weights are 2 and 1, respectively.

Result 1 Figure 4 shows a reactive system synthesized by our prototype for an input pair to LTL formula φ_1 and mean-payoff objective $\mathsf{MP}(t_{23})$. The system realizes φ_1 and tries to satisfy φ_2 (resp., φ_3) without violating φ_1 (resp., φ_1 or φ_2).

Discussion 1 To refine an unrealizable specification by adding assumptions, we can obtain an assumption $\mathbf{G}(\neg(req_1 \wedge req_2))$ for unrealizable $\varphi_1 \wedge \varphi_2$ that is reasonable in some cases. For φ, we can derive assumptions $\mathbf{G}\neg req_1$ and $\mathbf{G}\neg req_2$; however, they are trivially unallowable in a practical sense even if the refined formulae $(\mathbf{G}\neg req_1) \rightarrow \varphi$ and $(\mathbf{G}\neg req_2) \rightarrow \varphi$ are realizable. This is because φ_3 does not appropriately express the requirement. However, there are many natural requirements that cannot be represented in LTL. As in the above example, we often fail to obtain allowable assumptions. We can also attempt to refine an unrealizable specification by weakening its partial specifications. Unrealizable φ can be refined into, for example, $\varphi' = \varphi_1 \wedge \varphi_2' \wedge \varphi_3'$, under the assumption of a priority order for the specifications. Here, φ_2' (resp., φ_3') is a weakened specification for φ_2 (resp., φ_3), and

$$\varphi_2' = \mathbf{G}(((\neg req_1) \wedge req_2) \rightarrow \mathbf{X}\neg res), \tag{100}$$

$$\varphi_3' = \mathbf{G}((\neg(req_1 \vee req_2)) \rightarrow (res \leftrightarrow \mathbf{X}res)). \tag{101}$$

φ' is realized by the reactive system shown in Fig. 4. This example is very simple, and the unrealizable formula φ can be easily refined into the realizable formula φ'. In general, we

must repeatedly refine a specification and check its realizability. However, in our approach, we need to extract a must specification and interpret the desirable specifications using a mean-payoff objective that is based on their priorities. If the must specification is realizable, we can synthesize a concrete reactive system that considers the mean-payoff objective. Additionally, a mean-payoff objective can easily capture some requirements that are impossible to express in LTL, such as "*if possible*".

Instance 2 Next, we focus on the scalability of our implementation. We use a specification of an n-client load balancing system [18] as an instance. The specification is given as a combination of some formulae (for details, see "Appendix 1"). We denote each formula by φ_i^{LB} where i is its index. As shown in [18], the following formulae $\varphi_{\mathrm{a}}^{\mathrm{LB}}$, $\varphi_{\mathrm{b}}^{\mathrm{LB}}$ and $\varphi_{\mathrm{c}}^{\mathrm{LB}}$ are realizable.

$$\varphi_{\mathrm{a}}^{\mathrm{LB}} = \bigwedge_{i\in\{1,2\}} \varphi_i^{\mathrm{LB}}, \tag{102}$$

$$\varphi_{\mathrm{b}}^{\mathrm{LB}} = \bigwedge_{i\in\{1,2,4\}} \varphi_i^{\mathrm{LB}}, \tag{103}$$

$$\varphi_{\mathrm{c}}^{\mathrm{LB}} = \left(\bigwedge_{i\in\{6,7\}} \varphi_i^{\mathrm{LB}}\right) \rightarrow \bigwedge_{i\in\{1,2,5,8\}} \varphi_i^{\mathrm{LB}} \tag{104}$$

However, $\bigwedge_{i\in\{1,2,3\}} \varphi_i^{\mathrm{LB}}$, $\bigwedge_{i\in\{1,2,4,5\}} \varphi_i^{\mathrm{LB}}$ or $(\bigwedge_{i\in\{6,7\}} \varphi_i^{\mathrm{LB}}) \rightarrow \bigwedge_{i\in\{1,2,5,8,9\}} \varphi_i^{\mathrm{LB}}$ are not. According to the naive ideas on interpretation given in Sect. 3.3.1, we interpret φ_3^{LB}, φ_5^{LB} and φ_9^{LB} into the following payoff terms, respectively.

$$t_3^{\mathrm{LB}} = \sum_{0\leq i<n} \mathsf{B}\left(\mathbf{F}^{\leq 3} g_i\right), \tag{105}$$

$$t_5^{\mathrm{LB}} = \sum_{0\leq i<n} \mathsf{B}((\mathbf{X} g_i) \rightarrow \mathit{job}), \tag{106}$$

$$t_9^{\mathrm{LB}} = \mathsf{B}\left(\left(\bigvee_{1\leq i<n} \mathbf{X} g_i\right) \rightarrow \neg r_0\right). \tag{107}$$

In the interpretation for φ_3^{LB}, we use a formula-based approximation because the current implementation does not support B-terms.

Result 2 Table 1 shows an experimental result for the must LTL specification $\varphi_{\mathrm{a}}^{\mathrm{LB}}$ and mean-payoff objective $\mathsf{MP}(t_3^{\mathrm{LB}})$ when $n \in \{2, \ldots, 7\}$. Because $\varphi_{\mathrm{a}}^{\mathrm{LB}}$ is safe, there is no approximation in Algorithm 1, i.e., k is meaningless in this case. The numbers of clients are given in column "n". Column "α" (resp. "β") gives execution times for transforming UCWAs, which are constructed from the must specification (resp. payoff terms), into safety games (resp. mean-payoff games). This transforming includes a procedure for minimizing state-spaces of the games. Column "γ" gives execution times for solving weighted synchronized product games, i.e., constructing and minimizing action-abstracted games and solving them. Total execution times (including execution times for constructing UCWAs, etc.) are given in column "Total". We denote timeouts (>20 min) by "TO". Column "$|\mathcal{W}|$" (resp. "$|\mathcal{M}|$") gives the sizes of winning regions which are eventually obtained from the must specifications (resp., synchronized products of mean-payoff games, which are eventually obtained from the payoff terms). The sizes of weighted synchronized products of the winning regions

Table 1 Experimental result for the must LTL specification $\varphi_{\mathrm{a}}^{\mathrm{LB}}$ and mean-payoff objective $\mathsf{MP}(t_3^{\mathrm{LB}})$

n	Execution time (s)				Size				
	α	β	γ	Total	$\lvert\mathcal{W}\rvert$	$\lvert\mathcal{M}\rvert$	$\lvert\mathcal{P}\rvert$	$\lvert\mathcal{P}^{\mathrm{abs}}\rvert$	$\lvert\mathcal{R}\rvert$
2	0.022	0.035	0.027	0.201	5	59	116	59	5
3	0.026	0.061	0.063	0.278	9	199	460	84	17
4	0.022	0.472	0.183	0.818	17	707	1684	92	47
5	0.023	20.56	0.604	21.38	33	2599	5710	92	99
6	0.025	884.6	1.726	887.0	65	9779	18136	92	179
7	0.026	TO	–	–	129	–	–	–	–

Table 2 Experimental result for the must LTL specification $\varphi_{\mathrm{b}}^{\mathrm{LB}}$ and mean-payoff objective $\mathsf{MP}(t_5^{\mathrm{LB}})$

n	Execution time (s)				Size				
	α	β	γ	Total	$\lvert\mathcal{W}\rvert$	$\lvert\mathcal{M}\rvert$	$\lvert\mathcal{P}\rvert$	$\lvert\mathcal{P}^{\mathrm{abs}}\rvert$	$\lvert\mathcal{R}\rvert$
2	0.029	0.031	0.152	0.335	55	4	94	75	15
3	0.047	0.029	1.765	1.965	471	4	854	315	87
4	0.714	0.029	112.7	113.5	6449	4	12150	1624	711
5	61.12	0.026	TO	–	125,309	4	240,484	8353	–

and mean-payoff games are given in column "$|\mathcal{P}|$". Column "$|\mathcal{P}^{\mathrm{abs}}|$" gives the sizes of the abstracted and minimized games for the weighted synchronized products. The sizes of resulting systems are listed in column "$|\mathcal{R}|$". Table 2 (resp. Table 3) shows an experimental result for the must LTL specification $\varphi_{\mathrm{b}}^{\mathrm{LB}}$ (resp. $\varphi_{\mathrm{c}}^{\mathrm{LB}}$) and mean-payoff objective $\mathsf{MP}(t_5^{\mathrm{LB}})$ (resp. $\mathsf{MP}(t_9^{\mathrm{LB}})$) when $n \in \{2, \ldots, 5\}$ (resp. $n \in \{2, \ldots, 6\}$). We set the initial value for k as $n+1$ (resp. n) in this case. This is because it is the minimum value which makes $\mathcal{L}(\mathcal{T}^k(\varphi_{\mathrm{b}}^{\mathrm{LB}}))$ (resp. $\mathcal{L}(\mathcal{T}^k(\varphi_{\mathrm{c}}^{\mathrm{LB}}))$) realizable. These results suggest that our implementation can treat non-trivial scale instances.

Discussion 2 The current implementation is just a prototype and does not support B-terms. We conjecture that non-trivial scale instances with B-terms will be treated by a complete version. The values in column "$|\mathcal{W}|$" in Tables 1, 2 and 3 are used as a basis for the conjecture. This is because constructing a mean-payoff game from a B-term is nearly the same as constructing a safety game from an LTL formula with a UCWA approximation. Additionally, Tables 1, 2 and 3 suggest it is very effective to use an abstracted and minimized game of a weighted synchronized product mean-payoff game. The state spaces are significantly reduced in many cases. In Table 1, the sizes of the abstracted and minimized games are the same when the number n of clients is greater than the bound (i.e., 3) for the $\mathbf{F}$ operators in the payoff term t_3^{LB}. This fact suggests it may be possible to efficiently treat some types of S-terms with many $\mathbf{X}$ operators (and also B-terms with large bounds).

Instance 3 Finally, we confirm the trade-off between the optimality of synthesized systems and bound k in Algorithm 1 (and its computational cost) using a non-trivial scale instance. We use a specification of a 2-floor elevator system [2] as an instance. The specification is given as a combination of some formulae (for details, see "Appendix 2"). We denote by φ^{ELV}

Table 3 Experimental result for the must LTL specification φ_c^{LB} and mean-payoff objective $\mathsf{MP}(t_9^{\mathrm{LB}})$

n	Execution time (s)				Size				
	α	β	γ	Total	$\lvert\mathcal{W}\rvert$	$\lvert\mathcal{M}\rvert$	$\lvert\mathcal{P}\rvert$	$\lvert\mathcal{P}^{\mathrm{abs}}\rvert$	$\lvert\mathcal{R}\rvert$
2	0.025	0.029	0.019	0.085	20	4	23	18	8
3	0.047	0.030	0.132	0.362	101	4	110	54	25
4	0.686	0.029	0.750	1.620	810	4	861	188	148
5	25.03	0.031	13.29	38.76	8693	4	9126	691	1299
6	TO	–	–	–	–	–	–	–	–

Table 4 Experimental result for the must LTL specification φ^{ELV} and mean-payoff objective $\mathsf{MP}(t^{\mathrm{ELV}})$

k	Execution time (s)				Size				MP
	α	β	γ	Total	$\lvert\mathcal{W}\rvert$	$\lvert\mathcal{P}\rvert$	$\lvert\mathcal{P}^{\mathrm{abs}}\rvert$	$\lvert\mathcal{R}\rvert$	value
3	0.468	0.029	0.065	0.763	165	197	58	37	−0.500
6	5.246	0.027	1.995	7.281	1194	1185	323	93	−0.200
9	23.15	0.029	4.055	27.86	3015	3159	855	153	−0.125
12	68.12	0.024	33.51	103.1	5852	6069	1638	213	−0.091
15	156.7	0.031	85.16	224.9	9635	9915	2673	273	−0.071
18	352.0	0.032	432.1	790.9	14,354	14,697	3960	333	−0.059
19	415.3	0.040	350.6	779.8	16,135	16,499	4444	353	−0.056
20	535.3	0.156	650.6	1157	18,020	18,405	4958	373	−0.053
21	631.7	0.314	TO	–	20,009	20,415	5499	–	–

the specification. φ^{ELV} includes the following constraint.

$$\mathbf{G} \bigwedge_{i \in \{1,2\}} (\mathit{LocBtn}_i \rightarrow \mathbf{FX}\mathit{Loc}_i), \tag{108}$$

where $\mathit{LocBtn}_1, \mathit{LocBtn}_2 \in \mathit{IAP}$ and $\mathit{Loc}_1, \mathit{Loc}_2 \in \mathit{OAP}$. This formula means that, if the button on the i-th floor is pushed (LocBtn_i), the system must eventually move the cage to the i-th floor (Loc_i). Consider a desirable specification $\mathsf{MP}(t^{\mathrm{ELV}})$, where

$$t^{\mathrm{ELV}} = -\mathsf{S}(\mathit{Loc}_1 \wedge \mathbf{X}\mathit{Loc}_2) - \mathsf{S}(\mathit{Loc}_2 \wedge \mathbf{X}\mathit{Loc}_1). \tag{109}$$

Maximizing $\mathsf{MP}(t^{\mathrm{ELV}})$ means that the system tries to keep its cage on the same floor as long as possible. This type of desirable specification is required to save power consumption. The desirable specification $\mathsf{MP}(t^{\mathrm{ELV}})$ conflicts with the unsafe property given by Equation (108) (and also the must specification φ^{ELV}).

Result 3 Table 4 shows an experiment result for the must LTL specification φ^{ELV} and mean-payoff objective $\mathsf{MP}(t^{\mathrm{ELV}})$ when $k_{init} \in \{3, 6, 9, 12, 15, 18, 19, 20, 21\}$. Column "MP value" gives mean-payoff values of resulting systems. The other columns are the same as Tables 1, 2 and 3. The size of the product of mean-payoff games for payoff terms in t^{ELV} is 9. This result suggests that the mean-payoff value of the resulting system approaches the global optimum (0, for this instance) and its computational cost grows non-linearly, when k increases. For this instance, reasonably good systems are obtained in times that are practical.

5.3 Summary

We confirmed some advantages of our method with several experiments. They are summarized as follows.

- We can obtain a best-effort system from an unrealizable specification, by interpreting low-priority subspecifications into a mean-payoff objective.
- Our method can treat non-trivial scale instances in times that are practical.
- We can obtain a reasonably good system by choosing bound k (which is an over-approximation parameter for a given must LTL specification) considering the available computational resource and desired mean-payoff value.

6 Related work

6.1 LTL synthesis

In a naive game-based method for LTL synthesis, a *deterministic* ω-regular word automaton (e.g., deterministic parity word automaton) $\mathcal{D}^{\varphi}$ is translated from a given LTL specification φ. Next, an ω-regular game $\mathcal{G}^{\varphi}$ is transformed from $\mathcal{D}^{\varphi}$. Finally, a reactive system is composed as a winning strategy on $\mathcal{G}^{\varphi}$. The first phase can be performed by translating φ to a non-deterministic ω-regular word automaton $\mathcal{N}^{\varphi}$ [3,17,23,40], and, using Safra's construction [32,36–38], to determinize $\mathcal{N}^{\varphi}$ to $\mathcal{D}^{\varphi}$. However, Safra's construction is very complicated and difficult to implement efficiently. Recently, Esparza and Křetínský proposed another method for directly constructing a deterministic ω-regular automaton from an LTL formula [21]. Their method constructs transition-based automata, which are in most cases smaller than state-based automata obtained by Safra's construction, but is also complicated.

Therefore, some *Safraless* LTL synthesis methods were proposed [8,18,22,27,28]. In these methods, φ is appropriately under-approximated into a tractable automaton (e.g., safety automaton), which is used to construct a reactive system. Some tools are available for Safraless LTL synthesis, e.g., Lily[11] [27], Acacia+ [8–10], and Unbeast[12] [18]. In some Safraless LTL synthesis methods [8,18,22], φ is under-approximated by giving a bound k to UCWA $\mathcal{A}^{\varphi}$ equivalent to φ. A k-UCWA $\mathcal{A}^{\varphi,k}$ represents a safety property and can easily be determinized by a type of powerset (i.e., Safraless) construction. Therefore, a reactive system is composed as a winning strategy on a safety game $\mathcal{S}^{\varphi,k}$ corresponding to $\mathcal{A}^{\varphi,k}$. The state space of this winning strategy is typically minimized [19].

In our method, we apply the UCWA-based Safraless method to construct an under-approximated safety game from a must LTL specification and then compose a reactive system as a winning strategy on the game. This is also optimal for a given mean-payoff objective that represents weighted desirable specifications.

6.2 Maximum satisfiability problem

Our problem is similar to the *weighed partial maximum satisfiability (MAX-SAT) problem* [29].[13]

[11] Available at http://www.iaik.tugraz.at/content/research/design_verification/lily/.

[12] Available at http://www.react.uni-saarland.de/tools/unbeast/.

[13] The term *partial* means that some clauses are hard.

The weighed partial MAX-SAT problem considers a set of *hard clauses*, which must be satisfied, and a set of weighted *soft clauses*, which are satisfied if possible. The solution is a valuation that satisfies all the hard clauses and maximizes the total weight of the satisfied soft clauses. The input formulae of the problem are not temporal; the objective is given as the total weight of the satisfied soft clauses, and the problem considers satisfiability.

However, our problem considers a must LTL specification and a mean-payoff objective that represents weighted desirable specifications. The solution is a synthesized reactive system that realizes all the must specifications and maximizes the objective. The input formula for our problem is temporal, the objective is given as a mean-payoff of a sequence of weights that depend on LTL formulae, and our problem considers realizability.

6.3 Synthesis considering the mean-payoff objectives/constraints

In our approach, a set of weighted desirable specifications is represented by a naive mean-payoff objective based on LTL formulae with weights. The syntax of our objective is based on that of the mean-payoff constraints in [39]. In this paper, an argument of a S-term must be bounded, but it may be unbounded in [39] (i.e., $\mathcal{S}$-terms defined in Equation (95) are considered). This restriction is derived from the difficulty of dealing with non-determinacy (on payments) in the synthesis. [39] studied an LTL with multiple mean-payoff constraints and methods for model- and satisfiability-checking. Non-determinacy does not affect the model- or satisfiability-checking. In this paper, we instead used a new type of term, $\mathsf{B}^b(\varphi)$, to capture the violation of the approximated safety property using UCWA. Therefore, the expressive power of payoff terms in this paper is incomparable with that in [39]. The semantics of the mean-payoff constraints in [39] was purely based on LTL formulae. However, our objective in this paper is also based on an LTL-to-UCWA translator.

Our method synthesizes an optimal reactive system in a set of reactive systems that realize a certain under-approximation of a must LTL specification. This is an optimal memoryless strategy on a naive mean-payoff game that is constructed from a mean-payoff objective. This strategy can be efficiently computed [12]. Our problem can be strictly reduced to finding an optimal (or ε-optimal) strategy on a *mean-payoff parity game* [13], which is a synchronized product of a parity game constructed from the must LTL specification and the naive mean-payoff game. However, this reduction generally requires that we construct a deterministic parity word automaton that accepts words satisfying the must LTL specification. That is, Safra's or the Esparza-Křetínský construction.[14] Furthermore, the algorithm for solving the game in [13] requires recursively solving parity and mean-payoff games; its optimal (resp., ε-optimal) strategy generally requires infinite (resp., large) memory.

In [6], Bloem et al. studied *lexicographic mean-payoff (parity) games* with multi-dimensional weights. The mean-payoffs are lexicographically ordered based on the priority of the dimensions. Any lexicographic mean-payoff (parity) game can be reduced to a naive mean-payoff (parity) game [6]. Therefore, our method can be easily extended to allow such multi-dimensional lexicographical weighting. However, the time complexity of extending our method is exponential to the number of dimensions [6]. Bloem et al. also proposed a method for reducing an automata-based mean-payoff objective (and ω-regular specification) into a lexicographic mean-payoff (parity) game [6]. They composed an optimal strategy for the game. However, our method deals with an LTL specification and a formula-based mean-payoff objective.

[14] More precisely, the naive Safra's construction [36] and the Esparza-Křetínský method construct deterministic (generalized) Rabin/Streett automata, and hence parityizing [11] is also required.

In [15], Chatterjee et al. studied multi-dimensional mean-payoff games, and proposed a method for finding a winning strategy that guarantees that the multi-dimensional mean-payoff for any play is greater than or equal to a given threshold vector. Acacia+ supports synthesis from an LTL formula with multiple mean-payoff constraints [9] (based on [15]) and also one with optimization for a naive mean-payoff objective under a *Markov decision process* environment [10]. However, the mean-payoff constraints and objectives supported in Acacia+ are only based on S-terms. Strictly speaking, Acacia+ only supports payoffs for literals. Hence, if we use $\mathsf{S}(\chi)$ with a non-Boolean and bounded argument χ, we need to add a fresh atomic proposition $\hat{p} \in 2^{OAP}$ for output and an additional LTL specification $\mathbf{G}(\chi \leftrightarrow \mathbf{X}^{\lceil depth(\chi) \rceil} \hat{p})$. Our mean-payoff objective can be expressed by B-terms that capture the violations of approximated safety properties using UCWAs.

6.4 Synthesis considering assumptions

A reactive system specification is often given as an LTL formula with the implication form $\varphi_{Asmp} \rightarrow \varphi_{Grnt}$, where φ_{Asmp} is an assumption regarding the behavior of the environment, and φ_{Grnt} is a property that the system should guarantee. In [7], Bloem et al. presented some goals for a reliable system synthesized from such a specification. The *be-correct* goal is "to fulfill φ_{Grnt} if the environment fulfills φ_{Asmp}". This is the aim for a reactive system obtained by traditional LTL synthesis. Some approaches for the other goals were surveyed in [7]. In our method, we can include assumptions in must specifications in a naive sense.

As suggested in [7], synthesis with mean-payoff optimization can work against the *don't-be-lazy* goal, which is "to fulfill φ_{Grnt} as well as possible for as many situations as possible, even when φ_{Asmp} are not fulfilled". In our approach, the *don't-be-lazy* goal is accomplished by a system synthesized from the must LTL specification $\varphi_{Asmp} \rightarrow \varphi_{Grnt}$ and the mean-payoff objective interpreted from (sub-formulae of) φ_{Grnt}. The synthesized system realizes $\varphi_{Asmp} \rightarrow \varphi_{Grnt}$ and tries to satisfy (the sub-formulae of) φ_{Grnt} to the extent possible, even if φ_{Asmp} is violated. In [26], we focused on this type of synthesis and regarded the goal as maximizing the degree of *environmental tolerance* which is given by the mean-payoff objective with $\mathcal{S}$-terms and without B-terms, as in [39]. However, atomic terms occurring in the mean-payoff objective for a synthesis method in [26] is restricted to be S-terms.

Our approach also satisfies the *never-give-up* goal, which is "to try to satisfy φ_{Grnt} when you can if you cannot satisfy φ_{Grnt} for every environment behavior". In [7], Bloem et al. suggested that this goal can be achieved by adding a reasonable property $L \subseteq (2^{AP})^{\omega}$, such that $\mathcal{L}(\varphi_{Grnt}) \cup L$ is realizable and then synthesizing a system that realizes $\mathcal{L}(\varphi_{Grnt}) \cup L$. Existing approaches can define this property, L. For example, the negation of a new assumption ψ_{Asmp} on the behavior of the environment such that $\psi_{Asmp} \rightarrow \varphi_{Grnt}$ ($\equiv \neg\psi_{Asmp} \vee \varphi_{Grnt}$) is realizable. In [14], Chatterjee et al. proposed a method for computing this assumption. However, we must still check that it is reasonable. In [16], Damm and Finkbeiner introduced the *admissibility* of specifications and proposed a method for synthesizing a distributed system consisting of *dominant* reactive systems. A specification φ is *admissible* if there exists a dominant reactive system for φ. A reactive system $\mathcal{R}$ (or strategy) is *dominant* for φ if, for any sequence $s \in (2^{IAP})^{\omega}$ of inputs, $Intr_s(\mathcal{R}) \models \varphi$ if there exists $\mathcal{R}'$ such that $Intr_s(\mathcal{R}') \models \varphi$. In other words, this reactive system realizes $\mathcal{L}(\varphi) \cup CSS(\varphi)$, where $CSS(\varphi)$ is a set of possible interactions produced by counterexample input-sequences for *strong satisfiability* [31] of φ. That is,

$$CSS(\varphi) = \{(\alpha_0^O \cup \alpha_0^I)(\alpha_1^O \cup \alpha_1^I) \cdots \in (2^{IAP})^{\omega} \mid Intr_{\alpha_0^I \alpha_1^I \cdots}(\mathcal{R}') \not\models \varphi \text{ for any } \mathcal{R}'\}. \quad (110)$$

Similar to the above studies, Damm and Finkbeiner used $CSS(\varphi_{Grnt})$ as an additional property, L. Both of these existing methods do not satisfy the *don't-be-lazy* goal. For the additional

property L (i.e., $\mathcal{L}(\neg\psi_{Asmp})$ or $CSS(\varphi_{Grnt})$) considered in their methods, a system synthesized from a supplemented specification $\mathcal{L}(\varphi_{Grnt}) \cup L$ may stop trying to satisfy φ_{Grnt} if L is satisfied. Our method can be extended to fit this problem. For example, a system realizing $\mathcal{L}(\varphi_{Grnt}) \cup L$ and optimizing a mean-payoff objective based on (the sub-formulae of) φ_{Grnt} is expected to satisfy the *don't-be-lazy* and *never-give-up* goals. The dominant reactive system is also a type of best-effort system. However, note the difference between the interpretations of "best effort". In their view, a system can stop trying to satisfy φ for only the worst input-sequences in $CSS(\varphi)$. This type of "best effort" will be reasonable in some cases, e.g., in compositional synthesis for distributed systems, as in [16]. In our view, φ should be divided into must specifications and desirable specifications in the first place if φ is unrealizable. Additionally, we assume that each desirable specification has no assumption because a system tries to make the best effort possible regardless of whether the assumption holds. **G**-formula is basic form of reactive system specifications without assumptions, so that the best effort to satisfy a desirable specification $\mathbf{G}\varphi_i$ can be regarded to maximize the number of steps that satisfy φ_i. That is, try to satisfy φ_i at each step, even if φ_i is repeatedly violated. Synthesizing robust systems is another approach to obtain the *don't-be-lazy* goal. In [5], Bloem et al. proposed a method to synthesize a robust reactive system from a combination of safety- and liveness- properties. For safe φ_{Asmp} and φ_{Grnt}, Bloem et. al. focused on the ratio of the number of violations for φ_{Grnt} to the number of violations for φ_{Asmp}; a synthesized robust system that makes the ratio as small as possible. They suggested some types of violations for safety properties, and their *reset-on-error* heuristic corresponds to our idea of B-terms, which is based on the non-overlapped minimal bad prefixes. In our approach, minimizing the ratio for the heuristic can be interpreted as maximizing the mean-payoff objective given by Eq. (71).

7 Conclusions

In this work, we divided specifications into must specifications and desirable specifications. We proposed a method for efficiently synthesizing a reactive system, which realizes all the must specifications and endeavors as best as possible to satisfy the desirable specifications.

We derived a mean-payoff objective to encapsulate a set of desirable specifications. The syntax of the objective is given in LTL formulae, and its semantics is based on the UCWA-based approximation used in Safraless synthesis. The precise meaning of the mean-payoff objective depends on an LTL-to-automata translator. However, we can easily and flexibly describe the objective considering desirable specifications with weights.

In the proposed method, we first construct a safety game from a must LTL specification in the same way as UCWA-based Safraless synthesis. Note that assumptions can naively be included in the must specification. Then, mean-payoff games are constructed from atomic terms in the mean-payoff objective by reusing procedures from the Safraless synthesis. Note that the LTL-to-automata translator for mean-payoff objectives may be different from one for constructing the safety game from the must LTL specification. Finally, a reactive system is composed as an optimal strategy on a weighted synchronized product of games. We can obtain a preferable reactive system considering the objective, if the must LTL specification is realizable. We implemented a prototype of our method and performed several experiments. The results of these experiments suggest that our method can treat non-trivial scale instances and produce reasonably good systems in times that are practical.

The **G**-formula is a basic form of reactive system specifications without assumptions. This reasonably implies that the best effort to satisfy $\mathbf{G}\varphi$ is to maximize the number of

steps that satisfy φ on an infinite-step interaction with the environment. Our method can construct a reactive system that realizes a given must LTL specification and keeps trying to the extent possible to satisfy φ at each step for each desirable specification $\mathbf{G}\varphi$, even when $\mathbf{G}\varphi$ is violated. Such systems are expected in many practical applications. We expect that our approach can be further refined, so that it can be applied as a formal synthesis method for practical systems.

A direction for further refinement is to discuss what kind of LTL-to-automata translators is fit to our mean-payoff objectives. We believe that a framework to reflect the designer's intention on LTL-to-automata translation is required. Additionally, the implementation/method need to be complete/refined along with a series of experiments. The complexity of our synthesis method is doubly exponential time. Nevertheless, it may be possible to solve practical instances using state-of-the-art techniques.

As stated in Sect. 3.4, our method can synthesize a reactive system, giving consideration to a number of soft assumptions of the form $\mathbf{G}\varphi$ and the frequency with which their violation occurs or is witnessed. However, our method does not treat hard assumptions, which are strictly followed by the environment, on desirable specifications. Another direction is an extension for adding the hard assumptions on desirable specifications. A desirable specification in our method is given via a mean-payoff objective, and hence the hard assumptions should be mean-payoff (or probabilistic) constraints. This extension is required for obtaining a reactive system that is optimal under a certain situation satisfying the hard assumptions.

Appendix 1: Specification of load balancing system

A specification of an n-client load balancing system [18] is given as a combination of the following LTL formulae.

$$\varphi_1^{\mathrm{LB}} = \bigwedge_{0 \le i < n} \mathbf{G}\left((\mathbf{X} g_i) \to r_i\right), \tag{111}$$

$$\varphi_2^{\mathrm{LB}} = \bigwedge_{0 \le i < n} \mathbf{G}\left(g_i \to \left(\bigwedge_{j \in \{0,\ldots,n\} \setminus \{i\}} \neg g_j\right)\right), \tag{112}$$

$$\varphi_3^{\mathrm{LB}} = \bigwedge_{0 \le i < n} \mathbf{GF} g_i, \tag{113}$$

$$\varphi_4^{\mathrm{LB}} = \bigwedge_{0 \le i < n} \left((\mathbf{GF} r_i) \to \mathbf{GFX} g_i\right), \tag{114}$$

$$\varphi_5^{\mathrm{LB}} = \mathbf{G}\left(\left(\bigvee_{0 \le i < n} \mathbf{X} g_i\right) \to \mathit{job}\right), \tag{115}$$

$$\varphi_6^{\mathrm{LB}} = \mathbf{GF}\mathit{job}, \tag{116}$$

$$\varphi_7^{\mathrm{LB}} = \mathbf{G}\left(\left(job \wedge \bigwedge_{0\le i\le n} \mathbf{X}\neg g_i\right) \rightarrow \mathbf{X}job\right), \tag{117}$$

$$\varphi_8^{\mathrm{LB}} = \bigwedge_{0\le i\le n} \neg\mathbf{FG}\left(r_i \wedge \mathbf{X}\neg g_i\right), \tag{118}$$

$$\varphi_9^{\mathrm{LB}} = \mathbf{G}\left(\left(\bigvee_{1\le i<n} \mathbf{X}g_i\right) \rightarrow \neg r_i\right), \tag{119}$$

where $r_0, \ldots, r_{n-1}, job \in IAP$ and $g_0, \ldots, g_{n-1} \in OAP$. We added $\mathbf{X}$ for each output literal in the formulae with both input and output atomic propositions because an interaction starts with an output of a system in this paper (resp. an input from an environment in [18]). We omit $\varphi_{10}^{\mathrm{LB}}$ because it is not used in Sect. 5.

8 Appendix 2: Specification of 2-floor elevator system

A specification $\varphi_{Grnt}^{\mathrm{ELV}}$ of a 2-floor elevator system [2] is given as a conjunction of the following LTL formulae.

$$\mathbf{G}(Loc_1 \vee Loc_2), \tag{120}$$

$$\mathbf{G}((Loc_1 \rightarrow \neg Loc_2) \wedge (Loc_2 \rightarrow \neg Loc_1)), \tag{121}$$

$$\mathbf{G} \bigwedge_{i\in\{1,2\}} (LocBtn_i \rightarrow ((\mathbf{FX}Loc_i) \wedge (((\mathbf{X}Loc_i) \wedge \mathbf{X}ReqLoc_i)\mathbf{RX}ReqLoc_i))), \tag{122}$$

$$\mathbf{G} \bigwedge_{i\in\{1,2\}} ((Loc_i \wedge ReqLoc_i) \rightarrow (Open \wedge (Movable\mathbf{R}Loc_i))), \tag{123}$$

$$\mathbf{G} \bigwedge_{i\in\{1,2\}} ((((\mathbf{X}Loc_i) \wedge \mathbf{X}Movable)) \rightarrow (LocBtn_i\mathbf{RX}\neg ReqLoc_i))), \tag{124}$$

$$\mathbf{G} \bigwedge_{i\in\{1,2\}} ((Loc_i \wedge \neg ReqLoc_i) \rightarrow \neg Open)), \tag{125}$$

$$\mathbf{G}(Open \rightarrow ((\neg Open)\mathbf{R}\neg Movable)), \tag{126}$$

$$\mathbf{G}((\neg Open) \rightarrow (Open\mathbf{R}Movable)), \tag{127}$$

$$\mathbf{G}(Open \rightarrow \mathbf{F}OpenTimeout), \tag{128}$$

$$\mathbf{G}((OpenBtn \wedge \mathbf{X}\neg OpenTimeout) \rightarrow \mathbf{X}ReqOpen), \tag{129}$$

$$\mathbf{G}(OpenTimeout \rightarrow \neg Open), \tag{130}$$

$$\mathbf{G}((CloseBtn \wedge \mathbf{X}\neg ReqOpen) \rightarrow \mathbf{X}\neg Open), \tag{131}$$

$$\mathbf{G}((ReqOpen \wedge \neg Movable) \rightarrow Open), \tag{132}$$

where $LocBtn_1, LocBtn_2, OpenBtn, CloseBtn \in IAP$ and $Loc_1, Loc_2, ReqLoc_1, ReqLoc_2, Open, Movable, OpenTimeout, ReqOpen \in OAP$. We added $\mathbf{X}$ in the same manner as in the specification of a load balancing system. Equation (108) is a weakened property of Eq. (122).

$\varphi_{Grnt}^{\mathrm{ELV}}$ is unrealizable and thus we consider the following assumption in this paper.

$$\varphi_{Asmp}^{\mathrm{ELV}} = \mathbf{G} \bigwedge_{i\in\{1,2\}} (LocBtn_i \rightarrow \mathbf{X}\neg LocBtn_i). \tag{133}$$

That is, the specification φ^{ELV} used in Sect. 5 is given as follows.

$$\varphi^{\mathrm{ELV}} = \varphi^{\mathrm{ELV}}_{Asmp} \rightarrow \varphi^{\mathrm{ELV}}_{Grnt}. \tag{134}$$

References

1. Abadi, M., Lamport, L., Wolper, P.: Realizable and unrealizable specifications of reactive systems. In: Ausiello, Dezani-Ciancaglini, G.M., Della Rocca, S. (eds.) Automata, Languages and Programming, Lecture Notes in Computer Science, vol. 372, pp. 1–17. Springer, Berlin (1989). doi:10.1007/BFb0035748
2. Aoshima, T., Yonezaki, N.: Verification of reactive system specifications with outer event conditional formula. In: Proceedings of International Symposium on Principles of Software Evolution, 2000, pp. 189–193. IEEE (2000). doi:10.1109/ISPSE.2000.913238
3. Babiak, T., Křetínský, M., Řehák, V., Strejček, J.: LTL to Büchi automata translation: fast and more deterministic. In: Flanagan, König, C.B. (eds.) Tools and Algorithms for the Construction and Analysis of Systems, Lecture Notes in Computer Science, vol. 7214, pp. 95–109. Springer, Berlin (2012). doi:10.1007/978-3-642-28756-5_8
4. Benedikt, M., Lenhardt, R., Worrell, J.: LTL model checking of interval Markov chains. In: Piterman, N., Smolka, S. (eds.) Tools and Algorithms for the Construction and Analysis of Systems, Lecture Notes in Computer Science, vol. 7795, pp. 32–46. Springer, Berlin (2013). doi:10.1007/978-3-642-36742-7_3
5. Bloem, R., Chatterjee, K., Greimel, K., Henzinger, T.A., Hofferek, G., Jobstmann, B., Könighofer, B., Könighofer, R.: Synthesizing robust systems. Acta Inform. **51**(3–4), 193–220 (2014). doi:10.1007/s00236-013-0191-5
6. Bloem, R., Chatterjee, K., Henzinger, T., Jobstmann, B.: Better quality in synthesis through quantitative objectives. In: Bouajjani, A., Maler, O. (eds.) Computer Aided Verification, Lecture Notes in Computer Science, vol. 5643, pp. 140–156. Springer, Berlin (2009). doi:10.1007/978-3-642-02658-4_14
7. Bloem, R., Ehlers, R., Jacobs, S., Könighofer, R.: How to handle assumptions in synthesis. In: Chatterjee, K., Ehlers, R., Jha, S. (eds.) Proceedings 3rd Workshop on Synthesis, Electronic Proceedings in Theoretical Computer Science, vol. 157, pp. 34–50. Open Publishing Association (2014). doi:10.4204/EPTCS.157.7
8. Bohy, A., Bruyére, V., Filiot, E., Jin, N., Raskin, J.F.: Acacia+, a tool for LTL synthesis. In: Madhusudan, P., Seshia, S. (eds.) Computer Aided Verification, Lecture Notes in Computer Science, vol. 7358, pp. 652–657. Springer, Berlin (2012). doi:10.1007/978-3-642-31424-7_45
9. Bohy, A., Bruyére, V., Filiot, E., Raskin, J.F.: Synthesis from LTL specifications with mean-payoff objectives. In: Piterman, N., Smolka, S. (eds.) Tools and Algorithms for the Construction and Analysis of Systems, Lecture Notes in Computer Science, vol. 7795, pp. 169–184. Springer, Berlin (2013). doi:10.1007/978-3-642-36742-7_12
10. Bohy, A., Bruyère, V., Raskin, J.F.: Symblicit algorithms for optimal strategy synthesis in monotonic Markov decision processes. In: Chatterjee, K., Ehlers, R., Jha, S. (eds.) Proceedings 3rd Workshop on Synthesis, Electronic Proceedings in Theoretical Computer Science, vol. 157, pp. 51–67. Open Publishing Association (2014). doi:10.4204/EPTCS.157.8
11. Boker, U., Kupferman, O., Steinitz, A.: Parityizing Rabin and Streett. In: Lodaya, K., Mahajan, M. (eds.) IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2010), Leibniz International Proceedings in Informatics (LIPIcs), vol. 8, pp. 412–423. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl (2010). doi:10.4230/LIPIcs.FSTTCS.2010.412. http://drops.dagstuhl.de/opus/volltexte/2010/2882
12. Brim, L., Chaloupka, J., Doyen, L., Gentilini, R., Raskin, J.: Faster algorithms for mean-payoff games. Form. Methods Syst. Des. **38**(2), 97–118 (2011). doi:10.1007/s10703-010-0105-x
13. Chatterjee, K., Henzinger, T., Jurdzinski, M.: Mean-payoff parity games. In: Proceedings of 20th Annual IEEE Symposium on Logic in Computer Science, 2005. LICS 2005, pp. 178–187 (2005). doi:10.1109/LICS.2005.26
14. Chatterjee, K., Henzinger, T.A., Jobstmann, B.: Environment assumptions for synthesis. In: van Breugel, F., Chechik, M. (eds.) CONCUR 2008—Concurrency Theory, Lecture Notes in Computer Science, vol. 5201, pp. 147–161. Springer, Berlin (2008). doi:10.1007/978-3-540-85361-9_14
15. Chatterjee, K., Randour, M., Raskin, J.F.: Strategy synthesis for multi-dimensional quantitative objectives. Acta Inform. **51**(3–4), 129–163 (2014). doi:10.1007/s00236-013-0182-6
16. Damm, W., Finkbeiner, B.: Automatic compositional synthesis of distributed systems. In: Jones, C., Pihlajasaari, P., Sun, J. (eds.) FM 2014: Formal Methods, Lecture Notes in Computer Science, vol. 8442, pp. 179–193. Springer, Berlin (2014). doi:10.1007/978-3-319-06410-9_13

17. Duret-Lutz, A., Poitrenaud, D.: SPOT: an extensible model checking library using transition-based generalized Büchi automata. In: Proceedings of the IEEE Computer Society's 12th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, 2004 (MASCOTS 2004), pp. 76–83. IEEE Computer Society (2004). doi:10.1109/MASCOT.2004.1348184
18. Ehlers, R.: Symbolic bounded synthesis. Form. Methods Syst. Des. **40**(2), 232–262 (2012). doi:10.1007/s10703-011-0137-x
19. Ehlers, R., Moldovan, D.: Sparse positional strategies for safety games. In: Peled, D., Schewe, S. (eds.) Proceedings First Workshop on Synthesis, Berkeley, California, 7th and 8th July 2012. Electronic Proceedings in Theoretical Computer Science, vol. 84, pp. 1–16. Open Publishing Association (2012). doi:10.4204/EPTCS.84.1
20. Ehrenfeucht, A., Mycielski, J.: Positional strategies for mean payoff games. Int. J. Game Theory **8**, 109–113 (1979). doi:10.1007/BF01768705
21. Esparza, J., Křetínský, J.: From LTL to deterministic automata: a Safraless compositional approach. In: Biere, A., Bloem, R. (eds.) Computer Aided Verification, Lecture Notes in Computer Science, vol. 8559, pp. 192–208. Springer, Berlin (2014). doi:10.1007/978-3-319-08867-9_13
22. Finkbeiner, B., Schewe, S.: Bounded synthesis. Int. J. Softw. Tools Technol. Transf. **15**(5), 519–539 (2012). doi:10.1007/s10009-012-0228-z
23. Gastin, P., Oddoux, D.: Fast LTL to Büchi automata translation. In: Berry, G., Comon, H., Finkel, A. (eds.) Computer Aided Verification, Lecture Notes in Computer Science, vol. 2102, pp. 53–65. Springer, Berlin (2001). doi:10.1007/3-540-44585-4_6
24. Hagihara, S., Egawa, N., Shimakawa, M., Yonezaki, N.: Minimal strongly unsatisfiable subsets of reactive system specifications. In: Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering, ASE '14, pp. 629–634. ACM, New York (2014). doi:10.1145/2642937.2642968
25. Hagihara, S., Kitamura, Y., Shimakawa, M., Yonezaki, N.: Extracting environmental constraints to make reactive system specifications realizable. In: 2009 16th Asia-Pacific Software Engineering Conference, pp. 61–68 (2009). doi:10.1109/APSEC.2009.70
26. Hagihara, S., Ueno, A., Tomita, T., Shimakawa, M., Yonezaki, N.: Simple synthesis of reactive systems with tolerance for unexpected environmental behavior. In: Proceedings of the 4th FME Workshop on Formal Methods in Software Engineering, FormaliSE '16, pp. 15–21. ACM, New York (2016). doi:10.1145/2897667.2897672
27. Jobstmann, B., Bloem, R.: Optimizations for LTL synthesis. In: Formal Methods in Computer Aided Design, 2006. FMCAD '06, pp. 117 –124 (2006). doi:10.1109/FMCAD.2006.22
28. Kupferman, O., Vardi, M.: Safraless decision procedures. In: 46th Annual IEEE Symposium on Foundations of Computer Science, 2005. FOCS 2005, pp. 531–540 (2005). doi:10.1109/SFCS.2005.66
29. Li, C.M., Manyà, F.: MaxSAT, Hard and Soft Constraints, Frontiers in Artificial Intelligence and Applications, vol. 185, chap. 19, pp. 613–631. IOS Press, Amsterdam (2009). doi:10.3233/978-1-58603-929-5-613
30. Li, W., Dworkin, L., Seshia, S.A.: Mining assumptions for synthesis. In: 2011 9th IEEE/ACM International Conference on Formal Methods and Models for Codesign (MEMOCODE), pp. 43–50 (2011). doi:10.1109/MEMCOD.2011.5970509
31. Mori, R., Yonezaki, N.: Several realizability concepts in reactive objects. In: Kangassalo, H., Jaakkola, H., Hori, K., Kitahashi, T. (eds.) Information Modelling and Knowledge Bases IV, Frontiers in Artificial Intelligence and Applications, vol. 16, pp. 407–424. IOS Press, Amsterdam (1993)
32. Piterman, N.: From nondeterministic Büchi and Streett automata to deterministic parity automata. In: 21st Annual IEEE Symposium on Logic in Computer Science, 2006, pp. 255–264 (2006). doi:10.1109/LICS.2006.28
33. Pnueli, A.: The temporal logic of programs. In: 18th Annual Symposium on Foundations of Computer Science, 1977, pp. 46 –57 (1977). doi:10.1109/SFCS.1977.32
34. Pnueli, A., Rosner, R.: On the synthesis of a reactive module. In: Proceedings of the 16th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '89, pp. 179–190. ACM, New York (1989). doi:10.1145/75277.75293
35. Rosner, R.: Modular synthesis of reactive systems. Ph.D. thesis, Weizmann Institute of Science (1992)
36. Safra, S.: On the complexity of omega automata. In: Proceedings of the 29th Annual Symposium on Foundations of Computer Science, SFCS '88, pp. 319–327. IEEE Computer Society, Washington, DC (1988). doi:10.1109/SFCS.1988.21948
37. Schewe, S.: Tighter bounds for the determinisation of Büchi automata. In: de Alfaro, L. (ed.) Foundations of Software Science and Computational Structures, Lecture Notes in Computer Science, vol. 5504, pp. 167–181. Springer, Berlin (2009). doi:10.1007/978-3-642-00596-1_13
38. Schewe, S., Varghese, T.: Tight bounds for the determinisation and complementation of generalised Büchi automata. In: Chakraborty, S., Mukund, M. (eds.) Automated Technology for Verification and Analysis,

Lecture Notes in Computer Science, pp. 42–56. Springer, Berlin (2012). doi:10.1007/978-3-642-33386-6_5
39. Tomita, T., Hiura, S., Hagihara, S., Yonezaki, N.: A temporal logic with mean-payoff constraints. In: Aoki, T., Taguchi, K. (eds.) Formal Methods and Software Engineering, Lecture Notes in Computer Science, vol. 7635, pp. 249–265. Springer, Berlin (2012). doi:10.1007/978-3-642-34281-3_19
40. Vardi, M.Y., Wolper, P.: Reasoning about infinite computations. Inf. Comput. **115**, 1–37 (1994)
41. Wolper, P.: The tableau method for temporal logic: an overview. Logique et Analyse 119–136 (1985)
42. Zwick, U., Paterson, M.: The complexity of mean payoff games on graphs. Theor. Comput. Sci. **158**(1–2), 343–359 (1996). doi:10.1016/0304-3975(95)00188-3