Quantitative Assume Guarantee Synthesis[⋆]

Shaull Almagor[1], Orna Kupferman[2], Jan Oliver Ringert[3], and Yaron Velner[2]

[1] Department of Computer Science, Oxford University, UK
[2] School of Computer Science and Engineering, The Hebrew University, Israel
[3] School of Computer Science, Tel Aviv University

Abstract. In *assume-guarantee synthesis*, we are given a specification $\langle A, G\rangle$, describing an assumption on the environment and a guarantee for the system, and we construct a system that interacts with an environment and is guaranteed to satisfy G whenever the environment satisfies A. While assume-guarantee synthesis is 2EXPTIME-complete for specifications in LTL, researchers have identified the $\mathrm{GR}(1)$ fragment of LTL, which supports assume-guarantee reasoning and for which synthesis has an efficient symbolic solution. In recent years we see a transition to *quantitative synthesis*, in which the specification formalism is multi-valued and the goal is to generate high-quality systems, namely ones that maximize the satisfaction value of the specification.

We study quantitative assume-guarantee synthesis. We start with specifications in $\mathrm{LTL}[\mathcal{F}]$, an extension of LTL by quality operators. The satisfaction value of an $\mathrm{LTL}[\mathcal{F}]$ formula is a real value in $[0,1]$, where the higher the value is, the higher is the quality in which the computation satisfies the specification. We define the quantitative extension $\mathrm{GR}(1)[\mathcal{F}]$ of $\mathrm{GR}(1)$. We show that the implication relation, which is at the heart of assume-guarantee reasoning, has two natural semantics in the quantitative setting. Indeed, in addition to $\max\{1-A, G\}$, which is the multi-valued counterpart of Boolean implication, there are settings in which maximizing the ratio G/A is more appropriate. We show that $\mathrm{GR}(1)[\mathcal{F}]$ formulas in both semantics are hard to synthesize. Still, in the implication semantics, we can reduce $\mathrm{GR}(1)[\mathcal{F}]$ synthesis to $\mathrm{GR}(1)$ synthesis and apply its efficient symbolic algorithm. For the ratio semantics, we present a sound approximation, which can also be solved efficiently. Our experimental results show that our approach can successfully synthesize $\mathrm{GR}(1)[\mathcal{F}]$ specifications with over a million of concrete states.

1 Introduction

Synthesis is the automated construction of a system from its specification: given a linear temporal logic (LTL) formula ψ over sets I and O of input and output signals, we synthesize a finite-state system that *realizes* ψ [10, 26]. At each moment in time, the system reads a truth assignment, generated by the environment, to the signals in I, and it generates a truth assignment to the signals in O. Thus, with every sequence of

[⋆] The research leading to these results has received funding from the European Research Council under the European Union's 7th Framework Programme (FP7/2007-2013, ERC grant no 278410). Shaull Almagor is supported by ERC grant AVS-ISS (648701).

inputs, the system associates a sequence of outputs. The system realizes ψ if all the computations that are generated by the interaction satisfy ψ.

In recent years, researchers have considered extensions and variants of the classical setting of synthesis. One class of extensions originates from the *assume-guarantee* [4] approach that is taken in many settings of synthesis. There, the input to the synthesis problem consists of two parts: a behavior A that the environment is assumed to have, and a behavior G that the system is guaranteed to have [6].[5] Both A and G are over $I \cup O$. When A and G are in LTL, synthesis of the assume-guarantee pair $\langle A, G\rangle$ coincides with synthesis of the LTL formula $A \to G$. Still, the assume-guarantee approach brings with it new interesting problems. For example, one may study the weakest A that is required in order to make a given G realizable [6, 21], and dually, the strongest G we can guarantee with a given A [11]. Indeed, the duality between the system and the environment in synthesis is intensified in light of the duality between A and G in assume-guarantee specifications [17]. In the more practical side, there is a challenge of finding expressive specification formalisms for which assume-guarantee synthesis is feasible in practice. Indeed, for LTL, the problem is 2EXPTIME-complete [26]. In [25], the authors introduce the *General Reactivity of Rank 1* fragment of LTL ($\mathrm{GR}(1)$, for short). Essentially, a $\mathrm{GR}(1)$ formula states that if some initial, safety, and fairness environment assumptions hold, then some initial, safety, and fairness system guarantees hold. The synthesis problem for $\mathrm{GR}(1)$ is in EXPTIME. It is shown, however, in [25], that $\mathrm{GR}(1)$ has an efficient symbolic synthesis algorithm, which is polynomial in the number of the concrete states of the specification. $\mathrm{GR}(1)$ synthesis has been used in various application domains and contexts, including robotics [18], scenario-based specifications [24], aspect languages [23], and event-based behavior models [12], to name a few. In addition, it is shown in [22] that almost all common LTL specification patterns can be specified in $\mathrm{GR}(1)$.

Another class of extensions to the classical synthesis problem addresses the quality of synthesized systems. Since LTL is Boolean, synthesized systems are correct, but there is no reference to their quality. This is a crucial drawback, as designers would be willing to give up manual design only if automated-synthesis algorithms return systems of comparable quality. Addressing this challenge, researchers have developed quantitative specification formalisms. For example, in [3], the input to the synthesis problem includes also Mealy machines that grade different realizing systems. In [1], the specification formalism is the multi-valued logic $\mathrm{LTL}[\mathcal{F}]$. The satisfaction value of an $\mathrm{LTL}[\mathcal{F}]$ formula is a real value in $[0, 1]$, where the higher the value is, the higher is the quality in which the computation satisfies the specification. $\mathrm{LTL}[\mathcal{F}]$ is really a family of logics, each parameterized by a set $\mathcal{F} \subseteq \{f : [0, 1]^k \to [0, 1] \mid k \in \mathbb{N}\}$ of functions (of arbitrary arity) over $[0, 1]$. Using the functions in $\mathcal{F}$, a specifier can formally and easily prioritize different ways of satisfaction. For example, as in earlier work

[4] By "assume-guarantee" we refer to the notion of synthesis given environment assumptions and system guarantees, rather than the setting of multi-agent synthesis coined in [7].

[5] We note that an orthogonal line of work adds indirect assumptions about the environment, like *bounded synthesis*, where we assume that there is a bound on the size of the environment [19, 28], or *rational synthesis*, in which the environment has its own objectives [14] and is assumed to behave rationally.

on multi-valued extensions of LTL (c.f., [13]), the set $\mathcal{F}$ may contain the $\min\{x, y\}$, $\max\{x, y\}$, and $1 - x$ functions, which are the standard quantitative analogues of the $\wedge$, $\vee$, and $\neg$ operators. The novelty of $\mathrm{LTL}[\mathcal{F}]$ is the ability to manipulate values by arbitrary functions. For example, $\mathcal{F}$ may contain the binary function $\oplus_\lambda$, for $\lambda \in [0, 1]$. The satisfaction value of the formula $\varphi \oplus_\lambda \psi$ is the weighted (according to λ) average between the satisfaction values of φ and ψ. This enables the quality of the system to be an interpolation of different aspects of it. As an example, consider the $\mathrm{LTL}[\mathcal{F}]$ formula $\varphi = \mathsf{G}(\mathit{req} \to (\mathit{grant} \oplus_{\frac{2}{3}} \mathsf{X}\mathit{grant}))$. The formula specifies the fact that we want requests to be granted immediately and the grant to hold for two steps. When this always holds, the satisfaction value is $\frac{2}{3} + \frac{1}{3} = 1$. We are quite okay with grants that are given immediately and last for only one step, in which case the satisfaction value is $\frac{2}{3}$, and less content when grants arrive with a delay, in which case the satisfaction value is $\frac{1}{3}$.

Using a multi-valued specification formalism, synthesis is upgraded to generate not only correct, but also high-quality systems. In particular, the synthesis algorithm for $\mathrm{LTL}[\mathcal{F}]$ seeks systems of the highest possible satisfaction value. An extension of the Boolean setting to a quantitative one is of special interest in the case of assume-guarantee synthesis. Indeed, when A and G are multi-valued, there are several ways to define the satisfaction value of an assume-guarantee pair $\langle A, G\rangle$. If we adopt the semantics of $\mathrm{LTL}[\mathcal{F}]$ for $\to$, we get that the satisfaction value of $\langle A, G\rangle$ is the maximum between the "violation value" of A (that is, 1 minus its satisfaction value) and the satisfaction value of G. With this semantics we can, for example, synthesize a system that satisfies as many guarantees as possible when all the environment assumptions hold (see Example 2 in Section 5.2).

Sometimes, however, other semantics are more appropriate. Consider, for example, a specification where the environment assumption is the amount of gas in a fuel tank (normalized to $[0, 1]$) and the guarantee is the distance a car can go. An optimal strategy in the $\max\{1 - A, G\}$ semantics can assure a satisfaction value of $1/2$. However, the behavior of the strategy when the tank is more than half full need not be optimal and it could afford to drive only half of the maximal distance. On the other hand, in a *ratio semantics*, where the objective is to maximize G/A, the optimal strategy would strive to maximize the fuel consumption, which is more desirable.

Another interesting issue that arises in the setting of assume-guarantee synthesis and calls for a quantitative view is *cooperative reactive synthesis*, namely the ability of the system to influence the satisfaction value of A. Indeed, recall that both A and G are over $I \cup O$. While a system that causes A to fail does satisfy an $\langle A, G\rangle$ specification, it is very likely that a designer favors behaviors in which G holds over those in which A is violated. In [4], the authors study this issue and present a *hierarchy of cooperation levels* between the system and the environment. They also describe an algorithm that synthesizes systems with the highest possible cooperation level, namely ones that satisfy both A and G. With a quantitative approach to assume-guarantee synthesis, we can incorporate the hierarchy within the specification.

In this work we introduce and study *quantitative assume-guarantee synthesis*. The doubly-exponential solution for $\mathrm{LTL}[\mathcal{F}]$ synthesis applies to specifications of the form $A \to G$. We define and study $\mathrm{GR}(1)[\mathcal{F}]$, namely the fragment of $\mathrm{LTL}[\mathcal{F}]$ that is the multi-valued counterpart of $\mathrm{GR}(1)$. Recall that $\mathrm{GR}(1)$ formulas have two Boolean operators: conjunction (between the different components of A and G) and implication (between A and G). We discuss different possible multi-valued semantics to both operators. Our main contributions are as follows.

- We present a theoretical framework for quantitative assume-guarantee synthesis. We identify two natural special cases of interest, namely when implication stands for $\max\{1 - A, G\}$ or G/A (Section 2). For conjunction, we allow all monotonically increasing quantitative functions. We relate quantitative assume-guarantee synthesis with the solution of *quantitative two-player assume-guarantee games*. The winning values in these games correspond to the values with which a $\mathrm{GR}(1)[\mathcal{F}]$ specification can be realized, and winning strategies correspond to transducers that realize the specification in these values.
  - For the $\max\{1 - A, G\}$ semantics, we show an efficient synthesis algorithm for the case the number of fairness assumptions and guarantees is fixed. Further, we show that without this assumption, as well as in the case we allow a quantitative conjunction function that is not monotonically increasing, the corresponding quantitative assume-guarantee games cannot be solved efficiently, provided P≠NP (Section 3).
  - For the G/A semantics, we show that even for a single assumption and guarantee, the corresponding quantitative assume-guarantee games are as hard as (Boolean) parity games. We present a sound approximation that has an efficient solution. Essentially, our approximation replaces the eventuality requirements in the fairness assumptions and guarantees by finitary-fairness ones [8, 20] (Section 4).

  Our algorithms efficiently reduces the $\mathrm{GR}(1)[\mathcal{F}]$ synthesis problem to synthesis of a Boolean $\mathrm{GR}(1)$ specification. Hence, they work also in the symbolic setting.
- Finally, we present a series of experimental results that demonstrates the differences between the different semantics and the scalability of our solution in the symbolic setting (Section 5). Our experimental results also demonstrate the usefulness of the quantitative approach. Indeed, we handle specifications that are not realizable in the Boolean approach but have a high satisfaction value in the quantitative one.

Due to lack of space, in some cases the full proofs were omitted, and can be found in the full version in the authors' webpages.

2 Preliminaries

2.1 The Temporal Logic LTL$[\mathcal{F}]$

The linear temporal logic LTL$[\mathcal{F}]$, introduced in [1], generalizes LTL by replacing the Boolean operators of LTL with arbitrary functions over $[0, 1]$. The logic is actually a family of logics, each parameterized by a set $\mathcal{F}$ of functions.

Syntax Let AP be a set of Boolean atomic propositions, and let $\mathcal{F} \subseteq \{f : [0, 1]^k \to [0, 1] \mid k \in \mathbb{N}\}$ be a set of functions over $[0, 1]$. Note that the functions in $\mathcal{F}$ may have different arities. An LTL$[\mathcal{F}]$ formula is one of the following:

- `True`, `False`, or p, for $p \in AP$.

- $f(\varphi_1, ..., \varphi_k)$, $\mathsf{X}\varphi_1$, or $\varphi_1 \mathsf{U} \varphi_2$, for LTL$[\mathcal{F}]$ formulas $\varphi_1, \ldots, \varphi_k$ and a function $f \in \mathcal{F}$.

We define the description size $|\varphi|$ of an LTL$[\mathcal{F}]$ formula φ to be the number of nodes in the generating tree of φ. Note that the function symbols in $\mathcal{F}$ are treated as constant-length symbols.

Semantics We define the semantics of LTL$[\mathcal{F}]$ formulas with respect to infinite computations over AP. A *computation* is a word $\pi = \pi_0, \pi_1, \ldots \in (2^{AP})^\omega$. We use π^i to denote the suffix $\pi_i, \pi_{i+1}, \ldots$. The semantics maps a computation π and an LTL$[\mathcal{F}]$ formula φ to the *satisfaction value* of φ in π, denoted $[\![\pi, \varphi]\!]$. The satisfaction value is defined inductively as follows.[6]

- $[\![\pi, \texttt{True}]\!] = 1$ and $[\![\pi, \texttt{False}]\!] = 0$.
- For $p \in AP$, we have that $[\![\pi, p]\!] = 1$ if $p \in \pi_0$ and $[\![\pi, p]\!] = 0$ if $p \notin \pi_0$.
- For a function $f \in \mathcal{F}$, we have $[\![\pi, f(\varphi_1, ..., \varphi_k)]\!] = f([\![\pi, \varphi_1]\!], ..., [\![\pi, \varphi_k]\!])$.
- $[\![\pi, \mathsf{X}\varphi_1]\!] = [\![\pi^1, \varphi_1]\!]$.
- $[\![\pi, \varphi_1 \mathsf{U} \varphi_2]\!] = \max\limits_{i \geq 0}\{\min\{[\![\pi^i, \varphi_2]\!], \min\limits_{0 \leq j < i} [\![\pi^j, \varphi_1]\!]\}\}$.

It is not hard to prove, by induction on the structure of the formula, that for every computation π and formula φ, it holds that $[\![\pi, \varphi]\!] \in [0, 1]$. Also, the number of possible satisfaction values of φ is finite and is bounded by $2^{|\varphi|}$.

The logic LTL coincides with the logic LTL$[\mathcal{F}]$ for $\mathcal{F}$ that corresponds to the usual Boolean operators. For simplicity, we use these operators as an abbreviation for the corresponding functions, as described below. In addition, we introduce notations for some useful functions. Let $x, y \in [0, 1]$ be satisfaction values and $\lambda \in [0, 1]$ be a parameter. Then,

- $\neg x = 1 - x$
- $x \vee y = \max\{x, y\}$
- $x \wedge y = \min\{x, y\}$
- $x \to y = \max\{1 - x, y\}$
- $\nabla_\lambda x = \lambda \cdot x$
- $x \oplus_\lambda y = \lambda \cdot x + (1 - \lambda) \cdot y$

Other useful abbreviations are the "eventually" and "always" temporal operators, defined as follows.

- $\mathsf{F}\varphi_1 = \texttt{True}\mathsf{U}\varphi_1$. Thus, $[\![\pi, \mathsf{F}\varphi_1]\!] = \max\limits_{i \geq 0}\{[\![\pi^i, \varphi_1]\!]\}$.
- $\mathsf{G}\varphi_1 = \neg\mathsf{F}\neg\varphi_1$. Thus, $[\![\pi, \mathsf{G}\varphi_1]\!] = \min\limits_{i \geq 0}\{[\![\pi^i, \varphi_1]\!]\}$.

2.2 GR(1) and GR(1)$[\mathcal{F}]$

A *propositional assertion* θ is a Boolean formula over AP, describing a single state in a computation. An *invariant* is an LTL formula φ over AP that uses only the X ("next") operator, and with no nesting of X's. Thus, φ relates a state in a computation and its successor.

[6] The observant reader may be concerned by our use of max and min where sup and inf are in order. It is proven in [1] that there are only finitely many satisfaction values for a formula φ, thus the semantics is well defined.

The *General Reactivity of Rank 1* fragment of LTL ($\mathrm{GR}(1)$, for short), consists of formulas of the form[7]

$$(\theta^e \to \theta^s) \wedge (\theta^e \to \mathsf{G}((\mathsf{H}\varphi^e) \to \varphi^s)) \wedge ((\theta^e \wedge \mathsf{G}\varphi^e) \to (\bigwedge_{1 \le i \le k^e} \mathsf{GF}\psi_i^e \to \bigwedge_{1 \le i \le k^s} \mathsf{GF}\psi_i^s)),$$

for propositional assertions θ^e, θ^s, ψ_i^e, and ψ_i^s, and invariants φ^e and φ^s. We refer to θ^e and θ^s as the *initial assumption* and *initial guarantee*, respectively, refer to $\mathsf{G}\varphi^e$ and $\mathsf{G}\varphi^s$, as the *safety assumption* and *safety guarantee*, respectively, and refer to $\bigwedge_{1 \le i \le k^e} \mathsf{GF}\psi_i^e$ and $\bigwedge_{1 \le i \le k^s} \mathsf{GF}\psi_i^s$ as the *fairness assumption* and *fairness guarantee*, respectively.

The temporal operator H ("Henceforth") is the past variant of G. Thus, a position i in a computation π satisfies $\mathsf{H}\varphi$ if all suffixes π^j, for $j \le i$, satisfy φ.

We proceed to the quantitative counterpart. A *quantitative propositional assertion* θ is an $\mathrm{LTL}[\mathcal{F}]$ propositional formula over AP, assigning a value to a single state in a computation. A *quantitative conjunction* is a monotonically increasing function $\otimes : [0,1]^* \to [0,1]$, which maps a vector of satisfaction values to a new satisfaction value. Formally, for every two vectors $v, u \in [0,1]^n$, if $v \ge u$ (point wise), then $\otimes(v) \ge \otimes(u)$. A typical quantitative conjunction function is $\wedge$, where $x \wedge y = \min\{x, y\}$. A *quantitative implication* is a function $\mapsto$: $[0,1] \times [0,1] \to [0,\infty]$ that is monotonically decreasing in its first parameter and monotonically increasing in its second parameter. A typical quantitative implication function is $\to$, where $x \to y = \max\{1-x, y\}$.

The $\mathrm{GR}(1)[\mathcal{F}]$ fragment of $\mathrm{LTL}[\mathcal{F}]$ consists of formulas of the form
$(\theta^e \to \theta^s) \wedge (\theta^e \to \mathsf{G}((\mathsf{H}\varphi^e) \to \varphi^s)) \wedge ((\theta^e \wedge \mathsf{G}\varphi^e) \to (\otimes_{1 \le i \le k^e} \mathsf{GF}\psi_i^e \mapsto \otimes_{1 \le i \le k^s} \mathsf{GF}\psi_i^s)$, for propositional assertions θ^e and θ^s, invariants φ^e and φ^s, and quantitative propositional assertions ψ_i^e and ψ_i^s.

Note that the subformulas that refer to the initial and safety assumptions and guarantees are Boolean. Indeed, the assumptions and guarantees are propositional assertions and invariants, their satisfaction values are in $\{0,1\}$, and they are related by $\wedge$ and $\to$. The functions in $\mathcal{F}$ are these used in the quantitative propositional assertions ψ_i^e and ψ_i^s, as well as the functions $\mapsto$ and $\otimes$. It is easy to extend our results to a setting in which the initial and safety assumptions and guarantees are quantitative. We are going to focus on two quantitative implications: $x \to y$, mentioned above, which we are going to term *disjunctive implication*, and y/x, which we are going to term *ratio implication*. Note that the range of the ratio implication is $[0,\infty]$. We may consider variants of ratio implication with which the result is always in $[0,1]$. One possibility is to be fully satisfied whenever $y \ge x$, which corresponds to defining y/x as $\min\{1, y/x\}$. Another possibility, especially given the finite ranges of x and y, is to map the possible values of y/x to $[0,1]$ in a some monotonic way, for example by $1 - 1/(1 + y/x)$.

We may allow a $\mathrm{GR}(1)[\mathcal{F}]$ formula to apply different quantitative conjunctions $\otimes^e$ and $\otimes^s$ to relate the components of the assumption and the guarantee.

[7] In some papers in the literature, $\mathrm{GR}(1)$ formulas have the following weaker form. $(\theta^e \wedge \mathsf{G}\varphi^e \wedge \bigwedge_{1 \le i \le k^e} \mathsf{GF}\psi_i^e) \to (\theta^s \wedge \mathsf{G}\varphi^s \wedge \bigwedge_{1 \le i \le k^s} \mathsf{GF}\psi_i^s)$. That is, if some safety and fairness environment assumptions hold, then some safety and fairness system guarantees hold. The original semantics of [25] as well as the symbolic implementation follow the stronger semantics.

We define the *width* of a quantitative propositional formula ψ, denoted $width(\psi)$, as the number of different satisfaction values that ψ may have. We further denote the width of a $\mathrm{GR}(1)[\mathcal{F}]$ specification by $\max\{\max_{1\leq i\leq k^e} width(\psi^e_i), \max_{1\leq i\leq k^s} width(\psi^s_i)\}$, i.e., the least upper bound of the width of the quantitative propositional assertions appearing in the specification.

2.3 The Synthesis Problem

In the setting of open systems, the set AP of atomic propositions is partitioned into sets I and O of input and output signals. An (I, O)-transducer models the computations generated (deterministically) by a system when it interacts with an environment. The environment assigns values to the signals in I and the systems responds with an assignment to the signals in O. This process repeats forever. Formally, an (I, O)-*transducer* is a tuple $\mathcal{T} = \langle I, O, S, s_0, \rho, L\rangle$, where S is a finite set of states, $s_0 \in S$ is an initial state, $\rho : S \times 2^I \to S$ maps a state and an assignment for the input signals to a successor state, and $L : S \to 2^O$ is a labeling function that maps each state to an assignment for the output signals. Every sequence $i = i_0, i_1, \ldots \in (2^I)^\omega$ of assignments for the input signals induces a single trace $s = s_0, s_1, \ldots$ of $\mathcal{T}$, satisfying $s_{j+1} = \rho(s_j, i_j)$ for all $j \geq 0$, and induces the computation $\pi = \pi_0, \pi_1, \ldots$ over $2^{I\cup O}$ in which $\pi_j = i_j \cup L(s_j)$ for all $j \geq 0$.

In the Boolean setting, the *realizability* problem gets as input an LTL formula over $I \cup O$, and asks for the existence of an (I, O)-transducer all of whose computations satisfy the formula. In the quantitative analogue we seek the generation of high-quality systems. For a transducer $\mathcal{T}$ and an $\mathrm{LTL}[\mathcal{F}]$ formula φ, we define the satisfaction value of φ in $\mathcal{T}$, denoted $[\![\mathcal{T}, \varphi]\!]$, as $\min\{[\![\pi, \varphi]\!] : \pi$ is a computation of $\mathcal{T}\}$. Accordingly, given an $\mathrm{LTL}[\mathcal{F}]$ formula φ over $I\cup O$, the realizability problem is to find $\max\{[\![\mathcal{T}, \varphi]\!] :$ $\mathcal{T}$ is an (I, O)-transducer$\}$. The synthesis problem is then to find a transducer that attains this value.[8] Moving from an optimization to a decision problem, we say, given a specification φ and a threshold $T \in [0, \infty]$, that φ is *realizable with value* T if there is a transducer $\mathcal{T}$ such that $[\![\mathcal{T}, \varphi]\!] \geq T$.

As shown in [1], the synthesis problem for $\mathrm{LTL}[\mathcal{F}]$ is 2EXPTIME-complete. Essentially, as in the Boolean setting, it is possible to construct, given an $\mathrm{LTL}[\mathcal{F}]$ formula φ and a predicate $P \subseteq [0, 1]$, a nondeterministic generalized Büchi automaton $\mathcal{A}_{\varphi,P}$ that accepts exactly all computations π such that $[\![\pi, \varphi]\!] \in P$. This automaton can be used for solving the decision problems that correspond to the optimization problems for $\mathrm{LTL}[\mathcal{F}]$. In particular, in the case of synthesis, we can check the realizability of φ with value above some threshold $T \in [0, 1]$, by generating a game where the objective of the system is to generate only computations that are accepted by $\mathcal{A}_{\varphi,[T,1]}$.

Remark 1. Note that our definition for $[\![T, \varphi]\!]$ considered the worst-case setting, where the goal is to maximize the quality of the computation with the minimal quality. Alternatively, one can take a stochastic approach, where the goal is to generate a transducer

[8] The specification of the problem does not require the transducer to be finite. As we shall show, however, as in the case of LTL, if some transducer that attains the value exists, there is also a finite-state one that does so.

that maximizes the expected quality of a computation, subject to a given distribution of the input signals [2]. As even simple stochastic reachability games are not known to have a polynomial solution [15] we leave it to future work.

Remark 2. In Section 1, we discussed the challenge of *cooperative reactive synthesis* [4], where a hierarchy of cooperation levels is used in order to favor behaviors in which the guarantee holds over these in which the assumption is violated. Using $\text{LTL}[\mathcal{F}]$, the designer can easily specify her priorities in this issue. For example, if the assumption is φ^e and the guarantee is φ^s, then the $\text{LTL}[\mathcal{F}]$ specification $\nabla_{0.9}(\neg\varphi^e)\vee\varphi^s$ has satisfaction value 0.9 in computations that only violate the assumption, and thus its synthesis would prefer transducers in which the guarantee is satisfied. Tuning down our satisfaction with violation of the assumption can also be achieved by taking some power of $(\neg\varphi^e)$, as in $(\neg\varphi^e)^2 \vee \varphi^s$. Dually, $(\neg\varphi^e) \vee \sqrt{\varphi^s}$ tunes up satisfaction of the guarantee. The extend to which we want to tune the assumption down or the guarantee up typically depends on the ability of the system to influence the satisfaction of the assumption. Note that by tuning down the assumptions, we incentivize the system to satisfy the guarantees, rather than to falsify the assumptions. This overcomes a common pitfall of assume-guarantee synthesis.

2.4 Games

A *two-player game* is $\mathcal{G} = \langle V = V_1 \cup V_2, E, v_0, W\rangle$, where V is a set of vertices partitioned to $V_1 \cup V_2$, $E \subseteq V \times V$ is a set of directed edges, and $v_0 \in V$ is an initial vertex, and W is a winning condition, to be defined below. We assume that E is total in its first element. The game is played between Player 1 and Player 2. It starts in v_0. Whenever the current vertex v is in V_i, for $i \in \{1, 2\}$, Player i chooses an edge (v, u) and the game proceeds to u. Note that since E is total, there is always a legal move for the players. Formally, a *strategy* for Player i is a function $\tau_i : V^* \cdot V_i \to V$ such that for all $\pi \cdot v \in V^* \cdot V_i$, we have that $E(v, \tau_i(\pi \cdot v))$. The outcome of strategies τ_1 and τ_2 for the two players is the infinite path $v_0, v_1, v_2, \ldots$ where for all $j \geq 0$, we have that $v_{j+i} = \tau_i(v_0, \ldots, v_j)$, for the player i for which $v_j \in V_i$.

The winning condition W defines a subset of V^ω. The goal of Player 2 is to ensure that the outcome of the game is in W, while the goal of Player 1 is to make sure the outcome is not in W. Several types of winning conditions have been studied. In a *strong-fairness* game, the condition W is given by a formula $\bigwedge_{1\leq i\leq k^e} \mathsf{GF}\psi_i^e \to \bigwedge_{1\leq i\leq k^s} \mathsf{GF}\psi_i^s$, for predicates ψ_i^e and ψ_i^s over V. A path π in the game satisfies W if there is $1 \leq i \leq k^e$ such that π visits vertices that satisfy ψ_i^e only finitely often, or for all $1 \leq i \leq k^s$, it visits vertices that satisfy ψ_i^s infinitely often.

A *weighted game* augments $\mathcal{G}$ with a (multidimensional) *weight function* $w : V \to [0, 1]^k$, for some $k \in \mathbb{N}$. The *width* of a dimension $1 \leq i \leq k$ is $|\{w(v)[i] : v \in V\}|$, namely the number of different values that w may assign in the i-th dimension. Then, the width of w is the least upper bound on the widths of all dimensions. A weighted *strong-fairness* game is parameterized by quantitative conjunction and implication functions $\otimes$ and $\mapsto$. The winning condition is of the form $W = \otimes_{1\leq i\leq k^e} \mathsf{GF}w[i] \mapsto \otimes_{1\leq i\leq k^s} \mathsf{GF}w[k^e + i]$, for k^e and k^s such that $k = k^e + k^s$. The value of a path π is the evaluation of W in the path, where the value $w[i]$, for $1 \leq i \leq k$, in a vertex v, is

$w(v)[i]$. Thus, the first k^e dimensions in $w(v)$ are associated with environment assumptions, and then k^s dimensions are associated with systems guarantees. Accordingly, we use $e[i]$, for $1 \leq i \leq k^e$, to denote $w[i]$, and use $s[i]$, for $1 \leq i \leq k^s$, to denote $w[k^e + i]$. For a threshold T and a weighted game $\mathcal{G}$ with winning condition W, we say that Player 2 wins $\mathcal{G}$ with value T iff Player 2 has a strategy to force the game into paths with value at least T.

For sets I and O of input and output signals, we say that a game $\mathcal{G}$ is an (I, O)-game if, intuitively, the moves of Player 1 (the environment) correspond to assignments to the signals in I and these of Player 2 (the system) correspond to assignments to the signals in O. Formally, there is a finite set S such that $V = 2^{I \cup O} \times S$, moves of Player 1 change only the 2^I component of a vertex, and then moves of Player 2 change only the 2^O and S components. It is not hard to see that a strategy of Player 2 in an (I, O)-game induces an (I, O)-transducer with state space S.

In the Boolean setting, LTL and $\mathrm{GR}(1)$ synthesis is reduced to the solution of a two-player game. For LTL, the construction of the game involves a translation of the specification to an automaton. The special structure of $\mathrm{GR}(1)$ formulas circumvents the need to construct an automaton. Instead, the initial and safety conditions determine the initial vertex of the game as well as the allowed transitions, and the fairness conditions induce the winning condition. In the full version we describe a similar construction from $\mathrm{GR}(1)[\mathcal{F}]$ formulas to weighted strong-fairness games. Formally, we prove the following.

Theorem 1. *Consider a* $\mathrm{GR}(1)[\mathcal{F}]$ *formula* $\varphi = \varphi_{init} \wedge \varphi_{safe} \wedge ((\theta^e \wedge \mathsf{G}\varphi^e) \rightarrow (\otimes_{1 \leq i \leq k^e} \mathsf{GF}\psi^e_i \mapsto \otimes_{1 \leq i \leq k^s} \mathsf{GF}\psi^s_i)$ *over* $I \cup O$*. We can construct a weighted strong-fairness* (I, O)*-game* $\mathcal{G}$ *with weight function* $w : V \rightarrow [0, 1]^{k^e + k^s}$ *and winning condition of the form* $\otimes_{1 \leq i \leq k^e} \mathsf{GF}e[i] \mapsto \otimes_{1 \leq i \leq k^s} \mathsf{GF}s[i]$*, such that the state space of* $\mathcal{G}$ *is contained in* $2^{I \cup O} \times \{1, 2\}$*, the width of* w *is equal to the width of* φ*, and for every* $T \in [0, \infty]$*, we have that* φ *is realizable with value* T *iff Player* 2 *wins* $\mathcal{G}$ *with value* T*.*

3 Weighted Games with Disjunctive Implication

In this section we study weighted games with disjunctive implication, namely these induced by $\mathrm{GR}(1)[\mathcal{F}]$ formulas in which the satisfaction value of $x \mapsto y$ is $\max\{1 - x, y\}$.

3.1 Upper bound

We start with good news and show that we can translate weighted strong-fairness games to Boolean ones. In Section 5, we describe a symbolic implementation of this translation. Then, combining it with a symbolic algorith for Boolean $\mathrm{GR}(1)$ synthesis, we obtain a symbolic synthesis algorithm for $\mathrm{GR}(1)[\mathcal{F}]$.

Theorem 2. *Consider a weighted strong-fairness game* $\mathcal{G}$ *with* n *vertices, weight function* $w : V \rightarrow [0, 1]^{k^e + k^s}$ *of width* m*, and winning condition* $\otimes_{1 \leq i \leq k^e} \mathsf{GF}e[i] \mapsto \otimes_{1 \leq i \leq k^s} \mathsf{GF}s[i]$*. Given a threshold* T*, we can construct a Boolean strong-fairness game* $\mathcal{G}'$ *with* $O(n \cdot m^{k^e + k^s})$ *vertices, such that Player* 2 *wins* $\mathcal{G}$ *with value at least* T *iff he wins* $\mathcal{G}'$*.*

Proof. Intuitively, at each step of $\mathcal{G}'$ we record the maximal values that were attained for $e[i]$ during a certain segment. Once $\otimes_{1\leq i\leq k^e}\mathsf{GF}e[i] \geq 1-T$, we record that the assumptions have been fulfilled, and the environment visits a winning vertex and resets its record. Similarly, once $\otimes_{1\leq i\leq k^s}\mathsf{GF}s[i] \geq T$, we record that the guarantees have been fulfilled, so the system visits a winning vertex and resets its record. Then, the goal of the system is to generate only paths such that if the environment visits a winning vertex infinitely often, then so does the system.

We now turn to formalize this. Let $k = k^e + k^s$ and $\mathcal{G} = \langle V = V_1 \cup V_2, E, v_0, w, W\rangle$, with $w : V \to [0,1]^k$. We define $\mathcal{G}' = \langle S = S_1 \cup S_2, E', s_0, W'\rangle$ as follows. The vertices are (a finite subset of) $S = V \times [0,1]^k \times \{0,1,2\}$, with S_1 and S_2 determined by V_1 and V_2, respectively, in the first component, and the initial vertex is $s_0 = (v_0, r, 0)$ with $r \equiv 0$. Consider such a vertex $(v, r, b) \in S$. Intuitively, the game is played "mostly" on the $b = 0$ component, with visits to vertices with $b = 1$ whenever the environment assumptions hold, and to vertices with $b = 2$ whenever the system guarantees hold. We refer to r as a tuple $r = (r_1, \ldots, r_k)$.

We turn to define the edges. Consider vertices $s = (v, r, b)$ and $s' = (v', r', b')$. Then, $(s, s') \in E'$ if the following hold. First, if $b \in \{1, 2\}$, then we only reset the respective components of r. Thus, $v' = v, b' = 0$, and r' is obtained from r as follows: if $b = 1$ then $r'_i = 0$ for $1 \leq i \leq k^e$ and $r'_i = r_i$ for $k^e + 1 \leq i \leq k$, and similarly, if $b = 2$ then $r'_i = r_i$ for $1 \leq i \leq k^e$ and $r'_i = 0$ for $k^e + 1 \leq i \leq k$.

Next, for $b = 0$, the edges are induced by E. That is, v' is such that $(v, v') \in E$. In addition, we update the record by setting $r'_i = \max\{r_i, w(v')[i]\}$. Thus, r'_i records the maximal value seen by w in the i-th component since the last reset. Finally, for every $v \in V$, if $\otimes_{k^e+1\leq i\leq k} r_i \geq T$, we remove all outgoing edges from s, and set the only edge to $(v, r, 2)$. Otherwise, if $\otimes_{1\leq i\leq k^e} r_i \geq 1-T$, we remove all outgoing edges from s, and set the only edge to $(v, r, 1)$. Note that we give a priority to the guarantee, thus we go to a vertex with $b = 2$ whenever both $\otimes_{k^e+1\leq i\leq k} r_i \geq T$ and $\otimes_{1\leq i\leq k^e} r_i \geq 1-T$.

Observe that since the edges in the $b = 0$ component are determined by E, it is easy to draw a correspondence between paths in $\mathcal{G}$ and paths in $\mathcal{G}'$. Indeed, the only non-triviality in the correspondence is the reset operation. Since, however, resets do not change the first component of the vertex, the correspondence is maintained.

The winning condition in $\mathcal{G}'$ asserts that if vertices with $b = 1$ are visited infinitely often, then vertices with $b = 2$ should be visited infinitely often. That is, denoting $V \times [0,1]^k \times \{j\}$ by V^j, we have $W' = \mathsf{GF}(V^1) \to \mathsf{GF}(V^2)$.

The correctness of the construction follows from the next argument: For every path ρ in $\mathcal{G}$ and a corresponding path ρ' in $\mathcal{G}'$.

- $[\![\rho, \otimes_{1\leq i\leq k^e}\mathsf{GF}e[i]]\!] \geq 1-T$ iff ρ' satisfies $\mathsf{GF}(V^1)$.
- $[\![\rho, \otimes_{1\leq i\leq k^s}\mathsf{GF}s[i]]\!] \geq T$ iff ρ' satisfies $\mathsf{GF}(V^2)$.

Finally, we analyze the size of $\mathcal{G}'$. Consider a reachable vertex $(v, r, b) \in S$. Then, for every $1 \leq i \leq k$, we have that r_i is a value of $w[j]$ for some j. Accordingly, $|S| = O(|V| \cdot m^k \cdot |V|)$. In particular, if k is fixed, this is a polynomial blow-up with respect to $\mathcal{G}$. □

By composing Theorems 1 and 2, we can conclude with the following.

Theorem 3. *Consider a* $\mathrm{GR}(1)[\mathcal{F}]$ *formula* φ *over* $I \cup O$ *with* k^e *assumptions,* k^s *guarantees, and width* m. *Given a threshold* $T \in [0, \infty]$, *we can construct a Boolean strong-fairness game* $\mathcal{G}_\varphi$ *whose winning condition has a single assumption and a single guarantee, such that Player* 2 *wins in* $\mathcal{G}_\varphi$ *iff* φ *is realizable with value* T. *Moreover,* $\mathcal{G}_\varphi$ *has* $O(2^{|I \cup O|} m^{k^e + k^s})$ *vertices.*

3.2 Lower bounds

Theorem 3 reduces $\mathrm{GR}(1)[\mathcal{F}]$ realizability to the solution of strong-fairness games. The reduction relies on the monotonicity of the quantitative conjunctive operator $\otimes$. In addition, The obtained game is polynomial in $2^{|I \cup O|}$ whenever the number of assumptions and guarantees in the $\mathrm{GR}(1)[\mathcal{F}]$ formula is fixed. In this section, we show that if we drop either of the assumptions, then the corresponding weighted game becomes hard to solve. We start by dropping the monotonicity assumption.

Theorem 4. *Solving weighted strong-fairness games is NP-hard for non-monotonic* $\otimes$ *functions, even when there are no environment assumptions, and only two guarantees; i.e., when* $k^e = 0$ *and* $k^s = 2$.

Proof. We show a polynomial reduction from the problem of solving *two-dimensional parity games*. A two-dimensional parity game is $\mathcal{P} = \langle V = V_1 \cup V_2, E, v_0, p \rangle$, where V, E and v_0 describe a game graph, and $p : V \to \{1, ..., k\}^2$ is a priority function, assigning to every $v \in V$ two priorities $p(v) = (p_1(v), p_2(v))$. An infinite path is winning for Player 2 if the minimal priority that is visited infinitely often in each dimension is even. In Lemma 1 of [9], it is shown that solving such games is NP-hard.

Given a two-dimensional parity game $\mathcal{P}$, we construct a weighted strong-fairness game $\mathcal{G} = \langle V = V_1 \cup V_2, E, v_0, w, W \rangle$, where $w : V \to [0, 1]^2$ is of width k, and the winning condition W is of the form $\otimes(\mathsf{GF}w[1], \mathsf{GF}w[2])$, such that Player 2 wins $\mathcal{P}$ iff he wins $\mathcal{G}$ with value 1. Note that W has no environment assumption, and its system guarantee includes two conjuncts. For all $v \in V$, we define $w(v)[i] = \frac{1}{p_i(v)}$. The quantitative (non-monotonic) conjunction $\otimes$ is defined by $\otimes(x, y) = 1$ if $\frac{1}{x}$ and $\frac{1}{y}$ are even integers, and $\otimes(x, y) = 0$ otherwise.

We observe that for $i \in \{1, 2\}$, the satisfaction value of $\mathsf{GF}e[i]$ in a computation is $\frac{1}{x}$, where x is the minimal rank that occurs infinitely often in the computation in component i. Thus, a path has value 1 according to W iff it satisfies the parity condition. □

Next, we show that dropping the assumption about the number of assumptions and guarantees being fixed yields co-NP-hardness, even for a monotonic $\otimes$ function. Specifically, the function we consider is the average function.

Theorem 5. *Solving weighted strong-fairness games is co-NP-hard.*

Proof. We show that the complement problem is NP-hard, by showing a polynomial reduction from the SET-COVER problem, which was shown to be NP-hard in [16]. In the SET-COVER problem, we are given a set $U = \{1, \ldots, m\}$, a collection of subsets $S \subseteq 2^U$ and a number $k \in \mathbb{N}$. The problem is to decide whether there exists a collection $T \subseteq S$ with $|T| = k$ such that $\bigcup_{s \in T} s = U$. The collection T is called a *cover* of U. We note that the problem is NP-hard also for the special case where $k = \frac{|S|}{2}$. Given a SET-COVER instance as above, we construct a weighted strong-fairness game $\mathcal{G} = \langle V = V_1 \cup V_2, E, v_0, w, W \rangle$, where $w : V \to [0,1]^{|U|+|S|}$ is of width 3, $V_1 = S$, $V_2 = \emptyset$, and $E = V_1 \times V_1$. That is, Player 1 controls all the vertices of the graph, which is a clique of size $|S|$. Intuitively, Player 1 chooses a cover, i.e., subsets from S, and he wins iff the collection is of size at most $\frac{|S|}{2}$ and covers all the elements of U. We now formally define the weight function and the winning condition. The assumptions are the number of elements from U that are covered. Hence, for every $u \in U$, we add a dimension to the function e, and for every vertex $v \in V$ (recall that $V = S$), we define $e(v)[u] = 0.5$ if $u \in v$ and otherwise $e(v)[u] = 0$. The guarantees are the number of elements from S that are used in the cover. Hence, for every $r \in S$, we add a dimension to the function s, and for every vertex $v \in V$, we define $s(v)[r] = 1$ if $v = r$ and $s(v)[r] = 0$ otherwise. Finally, we set $\otimes$ to be the average function. A set cover of size at most $\frac{|S|}{2}$ exists iff Player 1 can violate the winning condition $\max(1 - \otimes(\mathsf{GF}e[1], \ldots, \mathsf{GF}e[|U|]), \otimes(\mathsf{GF}s[1], \ldots, \mathsf{GF}s[|S|])) > \frac{1}{2}$, and we are done.

□

4 Weighted Games with Ratio Implication

In this section we study weighted games with ratio implication, namely these induced by $\mathrm{GR}(1)[\mathcal{F}]$ formulas in which the satisfaction value of $x \mapsto y$ is y/x. For this purpose we define $x/0 = \infty$ and allow quantitative values in $[0, \infty]$.

4.1 Lower Bound

We first show that deciding weighted games with ratio implication is hard even for the simple winning condition $\mathsf{GF}e[1] \mapsto \mathsf{GF}s[1]$, namely when the fairness assumption and guarantee consists of a single quantitative propositional assertion. For simplicity, we refer to $e[1]$ and $s[1]$ by e and s, respectively. Thus, each vertex in the graph is labeled by two weights, e and s with values in $[0,1]$, and the value of a path π is the ratio $\frac{[\![\pi, \mathsf{GF}s]\!]}{[\![\pi, \mathsf{GF}e]\!]}$. We call weighted strong-fairness games with such a winning condition 1-*ratio games*. We show that deciding 1-ratio games is as hard as deciding parity games.

Theorem 6. *1-ratio games are polynomial-time inter-reducible with parity games.*

Proof. We first show a reduction from parity games to 1-ratio games. Let $\mathcal{G} = (V, E, p : V \to \{1, \ldots, n\})$ be a parity game. Consider weight functions $e, s : V \to \{0, 1, \ldots, n\}$, where $e(v) = p(v)$ if $p(v)$ is odd, and $e(v) = 0$ otherwise, and $s(v) = p(v)$ if $p(v)$ is even, and $s(v) = 0$ otherwise. For every infinite path, the maximal priority that is visited infinitely often is even if and only if $\frac{\mathsf{GF}s}{\mathsf{GF}e} \geq 1$.

We now show a reduction in the converse direction. W.l.o.g we consider a 1-ratio game with threshold 1 and with integer weights. A general threshold T can be simulated simply by multiplying environment weights by T. Rational weights can be transformed to integer weights by multiplying environment and system weights by the least common multiplier of the weights. Given a 1-ratio game $\mathcal{G} = \langle V, E, v_0, e, s \rangle$, consider the following priority function: If $s(v) \geq e(v)$, then $p(v) = 2s(v) + 2$, and otherwise $p(v) = 2e(v) + 1$. For every infinite path, the maximal priority that is visited infinitely often is even if and only if $\frac{\mathsf{GF}s}{\mathsf{GF}e} \geq 1$.

4.2 Upper bound in a finitary semantics

Theorem 6 motivates an approximated solution for the case of $\mathrm{GR}(1)[\mathcal{F}]$ formulas with ratio implication. Inspired by finitary parity games [8, 20], we strengthen the winning condition in order to have a polynomial algorithm. Intuitively, the specification $\frac{\mathsf{GF}s}{\mathsf{GF}e} \geq T$ requires that whenever a computation visits a vertex v, where the value of the assumption is $e(v)$, then eventually it would visit also a vertex u in which the value of the guarantee is T times bigger than $e(v)$, i.e., $s(u) \geq T \cdot e(v)$. The finitary condition requires the existence of a bound b, such that whenever a vertex v is visited, then a vertex u with $s(u) \geq T \cdot e(v)$ is visited within at most b moves.

Chatterjee et al., showed that finitary parity games have a polynomial solution. Hence, by Theorem 6, synthesis over the finitary version of $\frac{\mathsf{GF}s}{\mathsf{GF}e} \geq T$ is also polynomial. Here, we present an alternative solution that has two advantages: (i) it involves a reduction to Boolean strong-fairness games (while the solution in [8] involves repeated iterations of winning region computation for a so called *weak parity* objective), and thus allow us to use existing tools for $\mathrm{GR}(1)$ symbolic synthesis; (ii) it naturally scales to winning conditions that involve a conjunction of objectives.

In Section 5, we describe a symbolic implementation of the $\mathrm{GR}(1)[\mathcal{F}]$ synthesis algorithm that follows from our solution.

From finitary 1-ratio games to Boolean games We first formally define the finitary winning condition. A path $\pi = \pi_0, \pi_1, \ldots$ satisfies a a *finitary* 1-*ratio winning condition* $\frac{\mathsf{GF}s}{\mathsf{GF}e} \geq T$ if there is a bound $b \in \mathbb{N}$ such that for all $i \geq 0$, there is $0 \leq j_i \leq b$ such that $s(\pi_{i+j_i}) \geq e(\pi_i) \cdot T$. That is, whenever a vertex v is visited, a vertex u with $s(u) \geq e(v) \cdot T$ is visited within the next b rounds.

In order to obtain a reduction to Boolean strong-fairness games, we first consider a modified winning condition. Intuitively, the modified winning condition allows Player 2 to respond with a required guaranteed value within an unbounded number of rounds, yet he has to declare when he gives up and no longer tries to present a high guaranteed value, which is ok to do finitely often.

Formally, given a game $\mathcal{G}$ with a finitary 1-ratio winning condition with labels s and e, we define the game $\mathcal{G}'$ as follows:

- The vertices and edges are as in $\mathcal{G}$, except that Player 2 can always make a "give up" declaration when he takes a move.
- A Player 1 request is opened whenever a vertex is visited. A request can be either satisfied or closed.
  - A request of vertex v is satisfied when a vertex u with $s(u) \geq e(v) \cdot T$ is visited.
  - A request is closed when Player 2 gives up (and then, all requests are closed).
- A path satisfies the winning condition if Player 2 gives up only finitely many times and every request along the path is eventually satisfied or closed.

Clearly, if Player 2 wins $\mathcal{G}$, then the same strategy used there would be winning in $\mathcal{G}'$. In addition, taking b to be the size of the memory in a finite-memory winning strategy for Player 2 in $\mathcal{G}'$, we can prove that this strategy is winning also in $\mathcal{G}$. Formally, we have the following.

Lemma 1. *Player* 2 *wins* $\mathcal{G}$ *iff he wins* $\mathcal{G}'$.

We show that solving $\mathcal{G}'$, and in fact generating a winning strategy for Player 2, can be done in polynomial time. We do so by reducing $\mathcal{G}'$ to a Boolean strong-fairness game $\mathcal{G}''$. The latter games can be decided using Boolean $\mathrm{GR}(1)$ synthesis.

Given $\mathcal{G}'$, we label its vertices by 3 priorities. Indeed, the reduction is really to a parity game with 3 priorities (1, 2, and 3), which we can further translate to a strong-fairness winning condition. In order to label the vertices of $\mathcal{G}'$, the game $\mathcal{G}''$ keeps track of the maximal open request. This involves an $O(m)$ blow-up, for the width m of $\mathsf{GF}e \mapsto \mathsf{GF}s$. When the maximal request is satisfied, the vertex is labeled by priority 2. When the maximal request is closed, the vertex is labeled by priority 3. All other vertices are labeled by 1. Hence, if Player 2 gives up infinitely often or fails to eventually satisfy a request, then the outcome of the play is either 3 or 1, and the Player 2 loses. Otherwise, the outcome is 2 and Player 2 wins.

Lemma 2. *Player* 2 *wins* $\mathcal{G}'$ *iff he wins* $\mathcal{G}''$.

Lemmas 1 and 2 together imply that finitely 1-ratio games are polynomial time reducible to parity games with priority set $\{1, 2, 3\}$. It is not hard to see that winning in such games amounts to violating a strong-fairness condition, and thus can be specified as the negation of the $\mathrm{GR}(1)$ formula $\mathsf{GFV}^2 \to \mathsf{GF}V^3$, where V^j stands for vertices with priority j. Since synthesis tools for $\mathrm{GR}(1)$ specification generate also counter-strategies, namely, strategies for the environment in case the specification is not realizable, we have reduced finitely 1-ratio games to Boolean $\mathrm{GR}(1)$ synthesis.

From finitely (k^e, k^s)-ratio games to Boolean games In this section we extend the results above to conjunctions of objectives. Consider $\otimes$ functions that are monotonically increasing and the objective $\frac{\otimes_{1\leq i\leq k^s}\mathsf{GF}s[i]}{\otimes_{1\leq i\leq k^e}\mathsf{GF}e[i]} \geq T$. We first define a corresponding finitary condition. Let π be an infinite path in a graph. We say that a quantitative conjunction $\otimes_{1\leq i\leq k}\mathsf{GF}w[i]$ gets value x in position r if along the segment between the previous time that $\otimes_{1\leq i\leq k}\mathsf{GF}w[i]$ got a value x (or since the beginning of the path if it never got value x) and r, the path visited vertices $\{v_1, \ldots, v_k\}$ such that $\otimes(w(v_1)[1], \ldots, w(v_k)[k]) = x$. We note that in this segment the path may visit also other vertices other then $\{v_1, \ldots, v_k\}$, and the order of visits does not matter.

Winning a game $\mathcal{G}$ with finitely winning condition $W = \frac{\otimes_{1\leq i\leq k^s}\mathsf{GF}s[i]}{\otimes_{1\leq i\leq k^e}\mathsf{GF}e[i]} \geq T$, requires the existence of a bound $b \in \mathbb{N}$ such that a computation satisfies W if whenever $\otimes_{1\leq i\leq k^e}\mathsf{GF}e[i]$ gets value x, then $\otimes_{1\leq i\leq k^s}\mathsf{GF}s[i]$ gets value at least $x \cdot T$ at least once within the next b positions. We refer to W as a finitary (k^e, k^s)-ratio game.

It is not hard to see that the finitary ratio condition is a sound approximation of the ratio condition. Indeed, a winning strategy for the finitary version of W is also winning for its non-finitary version. In Section 5.2, we show an example where the approximation is not complete.

We now adjust the construction of $\mathcal{G}'$ in the 1-ratio case to finitary (k^e, k^s)-ratio games. The idea is similar, except that opening and closing of requests is now required for all values obtained along the computation.

- The vertices and edges are as in $\mathcal{G}$, except that Player 2 can always make a "give up" declaration when he takes a move.
- A Player 1 request for value x is opened whenever $\otimes_{1 \leq i \leq k^e} \mathsf{GF} e[i]$ gets value x. A request can be either satisfied or closed.
  - A request for value x is satisfied when $\otimes_{1 \leq i \leq k^s} \mathsf{GF} s[i]$ gets value greater or equal to $x \cdot T$.
  - A request is closed when Player 2 gives up (and then, all requests are closed).
- A path satisfies the winning condition if Player 2 gives up only finitely many times and every request along the path is eventually satisfied or closed.

By similar arguments as in Lemmas 1 and 2, Player 2 wins $\mathcal{G}'$ if and only if he wins $\mathcal{G}$. Moreover, a construction of $\mathcal{G}'$ and the reduction to $\mathrm{GR}(1)$ synthesis follows by the same arguments as in the proof of Lemma 2. As in the reduction in Theorem 2, the game $\mathcal{G}'$ needs to maintain of the values that $\otimes_{1 \leq i \leq k^s} \mathsf{GF} s[i]$ and $\otimes_{1 \leq i \leq k^e} \mathsf{GF} e[i]$ get. For this purpose we need to keep track of whether a request for value x was opened for every possible value of $\otimes_{1 \leq i \leq k^e} \mathsf{GF} e[i]$, i.e., we have to maintain a separate maximal record for every value of $\otimes_{1 \leq i \leq k^e}$. The reduction is explicitly described in Section 5.1. Let $m(\otimes^e)$ denote the number of different values that $\otimes_{1 \leq i \leq k^e} \mathsf{GF} e[i]$ can have. Note that $m(\otimes^e) \leq m^{k^e}$. By the above, the state blow-up required for maintaning the values is $(m^{k^s + k^e})^{m(\otimes^e)}$. Thus, when k^e, k^s, and $m(\otimes^e)$ are fixed, we get only a polynomial blowup.

5 Symbolic Solution

In this section we describe a symbolic implementation (section 5.1) and experimental results (sections 5.2, 5.2) for the $\mathrm{GR}(1)[\mathcal{F}]$ synthesis algorithm. In Section 3.1, we described a reduction from $\mathrm{GR}(1)[\mathcal{F}]$ synthesis to $\mathrm{GR}(1)$ synthesis. Our algorithm is based on combining a symbolic implementation of the reduction with the known symbolic algorithm for $\mathrm{GR}(1)$ synthesis. For synthesis we used the implementation of $\mathrm{GR}(1)$ from [5] based on JTLV [27] with CUDD 3.0 64Bit as a BDD engine. We ran the algorithms with Java 1.8 64Bit on a Windows 7 64Bit desktop computer with 16GB and an Intel 3.2GHz CPU. All the specifications are available in the supplementary material from `http://tinyurl.com/m5s4hsn`.

5.1 Symbolic encoding

Our goal is to synthesize a reactive system that interacts with an environment that generates truth assignments to ℓ Boolean input signals (variables), thus $I = \{x_1, \ldots, x_\ell\}$, and generates assignments to ℓ Boolean output signals, thus $O = \{y_1, \ldots, y_\ell\}$. We use $\overline{x}$ to denote $x_1, \ldots, x_\ell$, and similarly for $\overline{y}$. Each state in the system is an assignment $(\overline{x}, \overline{y}) \in \{0, 1\}^{I \cup O}$ to the signals. A computation of the system is an infinite sequence of assignments to the signals. When a time t is known from the context we denote by x the value of a variable x in time t and by x' the value of x in time $t+1$.

Adjusting the basic notions to the symbolic setting, we get that an *invariant* is a propositional formula $\varphi(\overline{x}, \overline{y}, \overline{x}', \overline{y}')$, relating the current and the next values of the variables. Also, a *propositional quality function* is $\psi : \{0,1\}^{I \cup O} \to [0,1]$, mapping each assignment to the variables (that is, each state) to a value in $[0,1]$. Let $\varphi = (\theta^e \to \theta^s) \wedge (\theta^e \to \mathsf{G}((\mathsf{H}\varphi^e) \to \varphi^s)) \wedge ((\theta^e \wedge \mathsf{G}\varphi^e) \to (\otimes_{1 \leq i \leq k^e} \mathsf{GF}\psi_i^e \mapsto \otimes_{1 \leq i \leq k^s} \mathsf{GF}\psi_i^s)$. Let $m(\psi)$ be the width of a quantitative propositional assertion ψ. Recall that $m(\psi) \leq 2^{\min\{|\psi|, |I \cup O|\}}$. We encode each quantitative propositional assertion $\psi \in \{\psi_1^e, \ldots, \psi_{k^e}^e, \psi_1^s, \ldots, \psi_{k^s}^s\}$ by $m(\psi)$ Boolean functions $\psi^1, \ldots, \psi^{m(\psi)}$, where $\psi^j(\overline{x}, \overline{y})$ holds iff $\psi(\overline{x}, \overline{y}) = j$.

In the presence of a threshold T, the user can encode the $\otimes$ operator with two formulas $\chi_{\otimes \geq T}$ and $\chi_{\otimes \geq 1-T}$ that define when the value of a conjunction is greater or equal to T and when it is greater or equal to $1-T$.

The symbolic solution is the reduction from sections 3 and 4. The maximal values record is constructed by automatically adding deterministic monitors to the $\mathrm{GR}(1)$ specification, similar to the temporal testers described in [5, Sect. 5.2]. These monitors add auxiliary variables and safety guarantees. In the reduction to Boolean $\mathrm{GR}(1)$, the fairness assumptions are determined according to the $\otimes$ function over the maximal values record.

Encoding disjunctive implication The reduction of disjunctive implication from section 3.1 generates $\mathrm{GR}(1)$ specifications with a single assumption and a single guarantee. Given φ as above, the reduction generates the $\mathrm{GR}(1)$ formula $(\hat{\theta}^e \to \hat{\theta}^s) \wedge (\hat{\theta}^e \to \mathsf{G}((\mathsf{H}\hat{\varphi}^e) \to \hat{\varphi}^s)) \wedge ((\hat{\theta}^e \wedge \mathsf{G}\hat{\varphi}^e) \to (\mathsf{GF}\hat{\psi}^e \to \mathsf{GF}\hat{\psi}^s))$, over the signals $\hat{I}$ and $\hat{O}$, where

- Let Aux be a set of auxiliary variables used for encoding maximal records. Thus, $\overline{a}^e = (a_1^e, \ldots, a_{k^e}^e)$ encodes the maximal record for $1 \leq i \leq k^e$, and $\overline{a}^s = (a_1^s, \ldots, a_{k^s}^s)$ encodes the maximal record for $1 \leq i \leq k^s$. Then, $\hat{I} = I$ and $\hat{O} = O \cup Aux$.
- $\hat{\theta}^e = \theta^e$ and $\hat{\theta}^s = \theta^s \wedge \bigwedge_{1 \leq i \leq k^e} a_i^e = 0 \wedge \bigwedge_{1 \leq i \leq k^s} a_i^s = 0$.
- $\hat{\varphi}^e = \varphi^e$ and $\hat{\varphi}^s = \varphi^s \wedge$
  $\bigwedge_{1 \leq i \leq k^e}(\text{if } \varphi_{\otimes \geq 1-T}(\overline{a}^e) \text{ then } a_i^{e\prime} = 0 \text{ else } a_i^{e\prime} = \max(a_i^e, \psi_i^e(\overline{x}', \overline{y}'))) \wedge$
  $\bigwedge_{1 \leq i \leq k^s}(\text{if } \varphi_{\otimes \geq T}(\overline{a}^s) \text{ then } a_i^{s\prime} = 0 \text{ else } a_i^{s\prime} = \max(a_i^s, \psi_i^s(\overline{x}', \overline{y}')))$.
- $\hat{\psi}^e = \varphi_{\otimes \geq 1-T}(\overline{a}^e)$ and $\hat{\psi}^s = \varphi_{\otimes \geq T}(\overline{a}^s)$.

The number of added Boolean auxiliary variables is $|Aux| = (k^e + k^s) \cdot log_2(m)$.

Encoding Ratio Objective Recall that the reduction for the ratio implication from section 4 leads to the negation of a $\mathrm{GR}(1)$ formula. Thus, in our experiments we used a variant of a $\mathrm{GR}(1)$ counter-strategy synthesis algorithm (see e.g., [17]). We denote the range of $\otimes_{1 \leq i \leq k^e}(\psi_i^e)$ by $range(\otimes^e)$. Given φ as above, the reduction generates a negated $\mathrm{GR}(1)$ specification $(\hat{\theta}^e \to \hat{\theta}^s) \wedge (\hat{\theta}^e \to \mathsf{G}((\mathsf{H}\hat{\varphi}^e) \to \hat{\varphi}^s)) \wedge ((\hat{\theta}^e \wedge \mathsf{G}\hat{\varphi}^e) \to (\bigwedge_{r \in range(\otimes^e)} \mathsf{GF}\hat{\psi}_r^s \wedge \mathsf{FG}\neg giveup))$, over the signals $\hat{I}$ and $\hat{O}$, where

- Let $giveup$ be a variable for the system to declare giving up and Aux be a set of auxiliary variables used for encoding maximal records for every $r \in range(\otimes^e)$.

Thus, $\overline{a}_r^e = (a_{1,r}^e, \ldots, a_{k^e,r}^e)$ encodes the maximal record for $1 \leq i \leq k^e$, and $\overline{a}_r^s = (a_{1,r}^s, \ldots, a_{k^s,r}^s)$ encodes the maximal record for $1 \leq i \leq k^s$. Then, $\hat{I} = I$ and $\hat{O} = O \cup \{giveup\} \cup Aux$.

- $\hat{\theta}^e = \theta^e$ and $\hat{\theta}^s = \theta^s \wedge \bigwedge_{r \in range(\otimes^e)} (\bigwedge_{1 \leq i \leq k^e} a_{i,r}^e = 0 \wedge \bigwedge_{1 \leq i \leq k^s} a_{i,r}^s = 0)$.
- $\hat{\varphi}^e = \varphi^e$ and $\hat{\varphi}^s = \varphi^s \wedge$
  $\bigwedge_{r \in range(\otimes^e)}$ if $\varphi_{\otimes \geq rT}(\overline{a}_r^s) \vee giveup$ then
  $\bigwedge_{1 \leq i \leq k^e} (a_{i,r}^{e\prime} = 0)$ else $a_{i,r}^{e\prime} = \max(a_{i,r}^e, \psi_i^e(\overline{x}', \overline{y}'))$
  $\bigwedge_{r \in range(\otimes^e)}$ if $\varphi_{\otimes \geq rT}(\overline{a}_r^s) \vee giveup$ then
  $\bigwedge_{1 \leq i \leq k^s} (a_{i,r}^{s\prime} = 0)$ else $a_{i,r}^{s\prime} = \max(a_{i,r}^s, \psi_i^s(\overline{x}', \overline{y}'))$.
- $\hat{\psi}_r^s = \varphi_{\otimes < r}(\overline{a}_r^e) \vee \varphi_{\otimes \geq rT}(\overline{a}_r^s))$.

The number of added Boolean auxiliary variables is $|Aux| = |range(\otimes^e)| \cdot (k^e + k^s) \cdot log_2(m)$.

5.2 Experimental results

Beyond the feasibility of our algorithms, the examples below demonstrate the usefulness of the quantitative approach in assume-guarantee synthesis. Indeed, it involves specifications that are not realizable in the Boolean approach, but have high satisfaction values in the quantitative approach.

Example 1: paint robot Consider a paint robot with two arms that paints parts of manufactured pieces (see Fig. 1). Each arm can paint using different colors. Colors can be changed, one at a time, when the environment (a human operator) supports the change. The goal of the robot is to always eventually paint pieces in a set of different color configurations expressed in its specification. A $\mathrm{GR}(1)$ specification of the robot controller is shown in List. 1.1. Essentially, the specification states that when the environment enables color change in the two arms, then the robot should produce all four combinations of colors. The colors used by each robot arm (`color[0]` and `color[1]`) are system controlled (output) and the respective supported color changes (`chg[0]` and `chg[1]`) are modeled as environment variables (input). The safety guarantees to not change colors unless supported are expressed in l. 13-14. The safety assumption that a change of both colors does not occur at the same time is expressed in l. 11. Finally, the fairness assumptions are to always eventually support color changes for each arm (l. 16) and the fairness guarantees are that the system always eventually colors pieces in color combinations `c1` to `c4` (l. 18).

The $\mathrm{GR}(1)$ specification of the robot is realizable. Notice that the $\mathrm{GR}(1)$ specification is unrealizable if one of the fairness assumptions was omitted; i.e., if one of the arms could have a constant color.

We obtain $\mathrm{GR}(1)[\mathcal{F}]$ specifications $\otimes^e(\psi_1^e, \psi_2^e) \rightarrow \otimes^s(\psi_1^s, \psi_2^s, \psi_3^s, \psi_4^s)$ with different semantics from the $\mathrm{GR}(1)$ specification, where for every fairness assumption and guarantee $\hat{\psi}_i$ we define a quantitative proposition ψ_i with value 1 if $\hat{\psi}_i$ is satisfied and 0 otherwise. The specification has 2 environment variables and 16 system variables (Boolean variables). The reductions use 6 auxiliary variables for the disjunctive

```
module PaintJobRobot
// Robot with arms that color pieces.
out Int(0..255)[2] color; // colors of robot
in boolean[2] chg; // color change allowed
define // different colorings
  c1 := color[0] < 128 & color[1] < 128;
  c2 := color[0] < 128 & color[1] >= 128;
  c3 := color[0] >= 128 & color[1] < 128;
  c4 := color[0] >= 128 & color[1] >= 128;
// change support does not appear at same
     time
asm G !(chg[0] & chg[1]);
// no change support implies same color
gar G !chg[0] -> next(color[0])=color[0];
gar G !chg[1] -> next(color[1])=color[1];
// always eventually support change
asm GF chg[0]; asm GF chg[1];
// always eventually produce coloring
gar GF c1; gar GF c2; gar GF c3; gar GF c4;
```

Listing 1.1. $\mathrm{GR}(1)$ specification with winning condition $(\mathsf{GF}\mathtt{chg[0]} \wedge \mathsf{GF}\mathtt{chg[1]}) \rightarrow (\mathsf{GF}\mathtt{c1} \wedge \mathsf{GF}\mathtt{c2} \wedge \mathsf{GF}\mathtt{c3} \wedge \mathsf{GF}\mathtt{c4})$

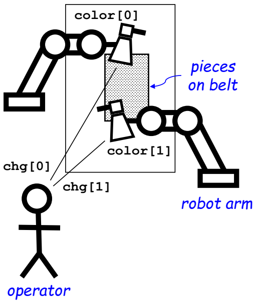

Fig. 1. Sketch of paint robot with two arms, different colors, and a human operator to support color changes.

implication semantics and 12 ($\otimes^e = \mathit{average}$) or 6 ($\otimes^e = \min$) auxiliary variables for the ratio implication. Table 1 shows maximal satisfaction values T for realizing the specifications, and the running times of realizability checks.

	$\otimes^e = \mathit{average}$	$\otimes^e = \min$
disjunctive implication	1/2 (40 ms)	1 (50 ms)
ratio implication	3/2 (121 ms)	3/1 (60 ms)

Table 1. Maximal value T and running time of $\mathrm{GR}(1)[\mathcal{F}]$ realizability check for $\otimes^s = \mathit{average}$, different $\otimes^e$, and different implication semantics (all specifications in supplementary material).

For $\otimes^e = \mathit{average}$, the maximal value of the disjunctive implication is $1/2$, as in the worst case if one assumption is violated the robot can only paint two colors, albeit the robot cannot commit on two specific colors. Note that even when both assumptions are satisfied, an optimal strategy need not paint more than two colors. For the ratio objective, the maximal value is $3/2$, therefore an optimal strategy paints 2 colors when one assumption holds and paints 3 colors when both assumptions hold. Thus, the optimal strategy for the ratio implication is more desirable in this case.[9]

Intuitively, for $\otimes^e = \min$, the environment has to satisfy all assumptions and thus the system should be able to paint all four colors. Indeed, in the disjunctive implication semantics we get value 1. However, in the ratio semantics, the formed optimal strategy only paints 3 colors because of the finitary overapproximation metric. The finitary condition dictates a bound over the response time of the system and in this case the immediate consequence of two changes is only three different colors.

[9] In the full version we explain why ratio 2 is impossible to obtain.

Spec	$\|\psi^e\|$	$\|\psi^s\|$	$\|I\| + \|O\|$	t in ms	$\|Aux\|$	max T	$\hat{t}$ in ms	$\hat{t}/t$
amba_ahb_wgf_1	2	4	5 + 11	11	7	3 / 4	305	28
amba_ahb_wgf_2	2	6	7 + 15	72	9	5 / 6	6,337	88
amba_ahb_wgf_3	2	8	9 + 19	410	12	7 / 8	499,630	1,219
amba_ahb_wgt_1	2	3	5 + 11	10	5	0 / 3	9	1
amba_ahb_wgt_2	2	5	7 + 15	51	8	0 / 5	1,032	20
amba_ahb_wgt_3	2	7	9 + 19	92	10	0 / 7	4,653	51
amba_ahb_woaf_1	1	3	5 + 11	35	5	2 / 3	257	7
amba_ahb_woaf_2	1	5	7 + 15	175	8	4 / 5	2,058	12
amba_ahb_woaf_3	1	7	9 + 19	1,132	10	5 / 7	75,945	67

Table 2. Unrealizable specifications from [11], the maximal average of satisfiable fairness guarantees, and running times of computing the winning states.

Example 2: maximal realizability One interesting application of $\mathrm{GR}(1)[\mathcal{F}]$ synthesis is to compute *maximal realizability* of $\mathrm{GR}(1)$ specifications, i.e., the maximal number of guarantees that can be satisfied when all assumptions hold. This is naturally captured in a quantitative setting where the quantitative value of a Boolean assumption or guarantee is 1 if it is satisfied and 0 otherwise. Maximal realizability is expressed by the $\mathrm{GR}(1)[\mathcal{F}]$ specification $\max(1 - \min(\psi_i^e), \mathit{average}(\psi_i^s))$ (note that average is the normalized sum).

For example, for a specification with environment variable $x \in I$ and guarantees $\mathsf{GF}x \wedge \mathsf{GF}\neg x$, the maximal realizability is $1/2$. Note that a computation of realizable subsets would perform worse and yield result 0.

We have checked maximal realizability for unrealizable specifications of the AMBA case study from [11]. For AMBA variants of different sizes Table 2 shows the number of fairness assumptions $|\psi^e|$ and guarantees $|\psi^s|$, the number of system and environment variables $|I| + |O|$, and the running times of the $\mathrm{GR}(1)$ algorithm for checking realizability on the original problem in milli-seconds. We selected three different sizes (prefix 1 to 3) for each variant (wgf: added fairness guarantee, wgt: added safety guarantee, and woaf: removed fairness assumption) of AMBA provided by [11]. Table 2 also reports on the auxiliary Boolean variables $|Aux|$ our reduction adds (here $|Aux| = |\psi^s| + \log_2(|\psi^s|)$, see the full version), the optimal T the system can guarantee, the time $\hat{t}$ of checking realizability of the reduced $\mathrm{GR}(1)$ game, and the ratio between t and $\hat{t}$.

Table 2 shows that for many unrealizable AMBA specifications, the maximal realizable satisfaction value T is high. The times for computing all winning states of the $\mathrm{GR}(1)[\mathcal{F}]$ specification show an expected increase with growing specification size.

References

1. S. Almagor, U. Boker, and O. Kupferman. Formalizing and reasoning about quality. *Journal of the ACM*, 63(3), 2016.
2. S. Almagor and O. Kupferman. High-quality synthesis against stochastic environments. In *CSL*, volume 62 of *LIPIcs*, pages 28:1–28:17, 2016.
3. R. Bloem, K. Chatterjee, T. Henzinger, and B. Jobstmann. Better quality in synthesis through quantitative objectives. In *CAV*, pages 140–156, 2009.

4. R. Bloem, R. Ehlers, and R. Könighofer. Cooperative reactive synthesis. In *ATVA*, pages 394–410, 2015.
5. R. Bloem, B. Jobstmann, N. Piterman, A. Pnueli, and Y. Sa'ar. Synthesis of Reactive(1) Designs. *J. Comput. Syst. Sci.*, 78(3):911–938, 2012.
6. K. Chatterjee, T. Henzinger, and B. Jobstmann. Environment assumptions for synthesis. In *CONCUR*, volume 5201 of *LNCS*, pages 147–161. Springer, 2008.
7. K. Chatterjee and T.A. Henzinger. Assume-guarantee synthesis. In *TACAS*, number 4424 in LNCS, pages 261–275. Springer, 2007.
8. K. Chatterjee, T.A. Henzinger, and F. Horn. Finitary winning in omega-regular games. *ACM Trans. Comput. Log.*, 11(1), 2009.
9. K. Chatterjee, T.A. Henzinger, and N. Piterman. Generalized parity games. In *FOSSACS*, pages 153–167, 2007.
10. A. Church. Logic, arithmetics, and automata. In *Proc. Int. Congress of Mathematicians, 1962*, pages 23–35. Institut Mittag-Leffler, 1963.
11. A. Cimatti, M. Roveri, V. Schuppan, and A. Tchaltsev. Diagnostic information for realizability. In *VMCAI*, volume 4905 of *LNCS*, pages 52–67. Springer, 2008.
12. N. D'Ippolito, V.A. Braberman, N. Piterman, and S. Uchitel. Synthesizing nonanomalous event-based controllers for liveness goals. *ACM Trans. Softw. Eng. Methodol.*, 22(1):9, 2013.
13. M. Faella, A. Legay, and M. Stoelinga. Model checking quantitative linear time logic. *Electr. Notes Theor. Comput. Sci.*, 220(3):61–77, 2008.
14. D. Fisman, O. Kupferman, and Y. Lustig. Rational synthesis. In *TACAS*, volume 6015 of *LNCS*, pages 190–204. Springer, 2010.
15. H. Gimbert and F. Horn. Solving simple stochastic games with few random vertices. 5(2), 2009.
16. R.M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103, 1972.
17. R. Könighofer, G. Hofferek, and R. Bloem. Debugging formal specifications: a practical approach using model-based diagnosis and counterstrategies. *STTT*, 15(5-6):563–583, 2013.
18. H. Kress-Gazit, G.E. Fainekos, and G.J. Pappas. Temporal-logic-based reactive mission and motion planning. *IEEE Trans. Robotics*, 25(6):1370–1381, 2009.
19. O. Kupferman, Y. Lustig, M.Y. Vardi, and M. Yannakakis. Temporal synthesis for bounded systems and environments. In *STACS*, pages 615–626, 2011.
20. O. Kupferman, N. Piterman, and M.Y. Vardi. From liveness to promptness. In *CAV*, volume 4590 of *LNCS*, pages 406–419. Springer, 2007.
21. W. Li, L. Dworkin, and S. A. Seshia. Mining assumptions for synthesis. In *MEMOCODE*, pages 43–50, 2011.
22. S. Maoz and J.O. Ringert. GR(1) synthesis for LTL specification patterns. In *ESEC/FSE*, pages 96–106, 2015.
23. S. Maoz and Y. Sa'ar. AspectLTL: an aspect language for LTL specifications. In *AOSD*, pages 19–30, 2011.
24. S. Maoz and Y. Sa'ar. Assume-guarantee scenarios: Semantics and synthesis. In *MODELS*, pages 335–351, 2012.
25. N. Piterman, A. Pnueli, and Y. Saar. Synthesis of reactive(1) designs. In *VMCAI*, volume 3855 of *LNCS*, pages 364–380. Springer, 2006.
26. A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *POPL*, pages 179–190, 1989.
27. A. Pnueli, Y. Sa'ar, and L.D. Zuck. JTLV: A framework for developing verification algorithms. In *CAV*, volume 6174 of *LNCS*, pages 171–174. Springer, 2010.
28. S. Schewe and B. Finkbeiner. Bounded synthesis. In *ATVA*, volume 4762 of *LNCS*, pages 474–488. Springer, 2007.