A Weakness Measure for GR(1) Formulae

Davide G. Cavezza (✉), Dalal Alrajeh, and András György

Imperial College London, London, United Kingdom
{d.cavezza15,dalal.alrajeh,a.gyorgy}@imperial.ac.uk

Abstract. In spite of the theoretical and algorithmic developments for system synthesis in recent years, little effort has been dedicated to quantifying the quality of the specifications used for synthesis. When dealing with unrealizable specifications, finding the weakest environment assumptions that would ensure realizability is typically a desirable property; in such context the weakness of the assumptions is a major quality parameter. The question of whether one assumption is weaker than another is commonly interpreted using implication or, equivalently, language inclusion. However, this interpretation does not provide any further insight into the weakness of assumptions when implication does not hold. To our knowledge, the only measure that is capable of comparing two formulae in this case is entropy, but even it fails to provide a sufficiently refined notion of weakness in case of GR(1) formulae, a subset of linear temporal logic formulae which is of particular interest in controller synthesis. In this paper we propose a more refined measure of weakness based on the Hausdorff dimension, a concept that captures the notion of size of the omega-language satisfying a linear temporal logic formula. We identify the conditions under which this measure is guaranteed to distinguish between weaker and stronger GR(1) formulae. We evaluate our proposed weakness measure in the context of computing GR(1) assumptions refinements.

1 Introduction

Specifications provide significant aid in the formal analysis of software supporting tasks such as their verification and implementation. However writing such specifications is difficult and error-prone, often resulting in their incompleteness, inconsistency and unrealizability [27]. Hence providing formal and rigorous support for ensuring their highest quality is of key importance [28]. One crucial quality metric for specifications, which this paper focuses on, is that of weakness in the context of reactive synthesis [2,5,15,21].

Reactive synthesis is concerned with finding a system implementation that satisfies a given specification under all possible environments [36]. When no such implementation exists, a specification is said to be unrealizable [19]. Though there may be many reasons for why a specification is unrealizable, a common cause is an incomplete set of assumptions over the environment behaviour. Several techniques [4,5,15,30] have been proposed in order to compute refinements for incomplete assumptions so as to ensure the realizability of a specification.

These approaches consider specifications expressed in a subset of linear temporal logic (LTL), namely generalized reactivity of rank 1 (GR(1)) [11,12,13], for which tractable synthesis methods exist. Their aim is to find the "weakest" assumptions amongst possible alternatives.

Assumption weakness [39] is a feature intended to capture the degree of freedom (or permissiveness) an environment satisfying the assumptions has over its behaviours; generally, weaker assumptions are preferred since they allow for more general solutions to the synthesis problem [18,39]. Existing approaches formalize the weakness relation between assumptions through logical implication [4,39], i.e., a formula ϕ_1 is weaker than a formula ϕ_2 if $\phi_2 \rightarrow \phi_1$ is valid. However, this notion does not fully capture the weakness concept as permissiveness [14]. Consider the simple example of a bus arbiter whose environment consists of three devices that can request for bus access. Let r_i be the binary signal meaning "device i requests access". An assumption like "device 1 requests access infinitely often" ($\mathbf{GF}r_1$ in LTL) is intuitively less constraining than "device 2 and 3 request access infinitely often" ($\mathbf{GF}(r_2 \wedge r_3)$). However, since the two assumptions refer to disjoint subsets of variables, no implication relation holds between the two.

To enable comparison between weakness of specifications as in the case above, we propose a quantitative measure for the weakness of GR(1) formulae (Sec. 4)—based on their interpretation as an ω-language—and a procedure to compute it. The measure builds upon the notion of Hausdorff dimension [41], a quantity providing an indication of the size of an ω-language: the higher the dimension, the wider the collection of distinct ω-words contained in the ω-language (Sec. 5). We show that a sufficient condition for assumptions expressed as invariants to be comparable through our measure is the *strong connectedness* of the underlying ω-language (Sec. 5.1). To compare assumptions containing fairness conditions, we identify and measure a language decomposition based on fairness complements (Sec. 5.2-5.3). We finally demonstrate the use of our proposed weakness measure on a set of assumptions refinement benchmarks (Sec. 6). Though we focus on comparing the weakness of assumptions refinements, the applied scope of our weakness metric can be extended to other contexts, e.g., quantitative model checking, in the form of a measure of the set of behaviors violating some given property (see [6]) and specification coverage as in [8,42].

2 Related Work

The closest notion to our measure is the *entropy* of ω-languages applied by Asarin et al. [6,7] to quantitative model checking. This quantity measures how diverse the ω-words contained in the language of an LTL formula are. However, it is not sufficiently fine-grained to distinguish between weaker and stronger fairness conditions [6]. We will show that our metric based on Hausdorff dimension is capable of making this distinction.

Quality of LTL formulae has also been defined in the context of model verification. The work by Henzinger et al. [25,26] defines a similarity measure between models of LTL formulae so as to render the model checking output quantitative:

instead of returning a true/false response, quantitative model checking computes the distance (*stability radius*) of the model from the boundary of the satisfiability region of an LTL property. The scope of our work is different: the measure we propose can be interpreted as the *extension* of such a satisfiability region, which is independent of a specific model to check against.

An alternative way to measure behaviour sets is via probabilities. Probabilistic model checking [24,29] enhances the syntax and semantics of temporal logics (usually *computation tree logic*) with probabilities. This allows for the expressions of properties like "the probability of satisfying a temporal logic formula ϕ by the modelled behaviours is at most p." Further extensions of LTL and/or automata with preference metrics alternative to probabilities have been proposed in [3,10,17,18]. The difference between using such quantities and our proposal is that while all of these measures are additional and depend on arbitrary parameters that may not reflect the true weakness of a logical formula, the measure we propose quantifies a concept of weakness *intrinsic* to the LTL formula itself.

The problem of identifying weakest assumptions appears in the context of assume-guarantee reasoning [20,31,35] for compositional model checking. In order to perform model checking of large systems, those systems are generally broken down to components that can be checked independently for correctness. In this context, one of the challenges is to identify the most general (weakest) assumptions over the environment in which each component operates, such that when they are satisfied, the correctness of the entire system is guaranteed. Assumptions are formalized as transition systems (e.g., modal transition systems) rather than declarative LTL specifications, which is the focus of our work.

3 Preliminaries

Languages and Automata. Let Σ be a finite set of symbols, which we call *alphabet*. A *word* over Σ is a finite sequence of symbols in Σ. An *ω-word* is an infinite sequence of such symbols. A set of words is called a *language*, while a set of ω-words is called an ω-language. A word w is explicitly denoted as a sequence of its symbols $w_1 w_2 \dots w_n$, or with a parenthesis notation $(w_1, w_2, \dots, w_n)$, with the symbols separated by commas; the same notation is used for ω-words. The notation w^j denotes the suffix of w starting with w_j.

Given two words v and w, their concatenation is denoted as $v \cdot w$ or simply vw. The same notation is used for the concatenation of a word v and an ω-word w; the concatenation of an ω-word and a word is not defined. Given a set V of finite-length words and a set W of finite-length words or ω-words over the same alphabet Σ, the set $V \cdot W$ is the set of words obtained by concatenating a word in V with a word in W. *Kleene's star operator* yields the set V^* of finite words obtained by concatenating an arbitrary number of words in V. The *omega operator* applied to V yields the set V^ω of ω-words obtained by concatenating a (countably) infinite number of words in V. Naturally, Σ^* and Σ^ω represent, respectively, the set of all finite words and all ω-words over the alphabet Σ. The

star and omega operators can also be applied to single finite-length words, like in w^* and w^ω.

Given an ω-language $L \subseteq \Sigma^\omega$, we denote by $A_n(L)$ the set of all $w \in \Sigma^*$ such that w is a prefix of a word in L and $|w| = n$. We also define $A(L) = \bigcup_{n\in\mathbb{N}} A_n(L)$ the set of all the prefixes of ω-words in L. It is possible to define a topology on Σ^ω. For more details, we refer the reader to [41]. In this context, we only need the notions of closed ω-languages and of their closure. An ω-language L is *closed* if and only if for any ω-word w such that $A(\{w\}) \subseteq A(L)$, $w \in L$. In other words, L is closed if whenever a word w is arbitrarily close (up to a prefix of arbitrary length) to some word in L, then $w \in L$. The *closure* of an ω-language L, denoted by $\mathcal{C}(L)$, is the smallest closed ω-language that contains L.

The notion of regular ω-languages encompasses ω-languages that allow a finite representation through automata. Formally, we define a *regular ω-language* as an ω-language which is accepted by a deterministic Muller automaton. A *deterministic Muller automaton* (DMA) is defined by the quintuple $\mathcal{M} = \langle Q, \Sigma, q_0, \delta, T\rangle$, where Q is a set of states, Σ is the alphabet of the ω-language, q_0 is the initial state, $\delta : Q \times \Sigma \to Q$ is the transition (partial) function and $T \subseteq 2^Q$ is a set (a table) of accepting state sets. Given an ω-word $w \in \Sigma^\omega$, the *run* induced by w onto $\mathcal{M}$ is a sequence of states $\mathcal{M}(w) = q_0 q_1 \dots$ such that q_0 is the initial state and $q_i = \delta(q_{i-1}, w_i)\ \forall i \in \mathbb{N}$. Let $\mathrm{Inf}(w) \subseteq Q$ be the set of states occurring infinitely many times in $\mathcal{M}(w)$. Then an ω-word is said to be *accepted* by $\mathcal{M}$ iff $\mathrm{Inf}(w) \in T$. By extension, the ω-language accepted by $\mathcal{M}$ is the set of ω-words accepted by $\mathcal{M}$.

A *deterministic Büchi automaton* (DBA) $\mathcal{B}$ is defined in the same way as a DMA except for the acceptance condition, which is stated in terms of a subset of states $F \subseteq Q$. A word w is accepted by $\mathcal{B}$ iff $\mathrm{Inf}(w) \cap F \neq \varnothing$. Given a DBA it is always possible to obtain an equivalent DMA by replacing the Büchi acceptance condition with the table $T = \{Q' \in 2^Q \mid Q' \cap F \neq \varnothing\}$. In Sec. 6 we also refer to nondeterministic automata, where the transition function is replaced by a transition relation and the initial state by a set of initial states.

Linear Temporal Logic and GR(1). *Linear temporal logic* (LTL) [37] is an extension of Boolean logic with temporal operators. It allows for expressing properties of infinite sequences of assignments to a set $\mathcal{V}$ of Boolean variables. Its syntax and semantics are described in the extended version of this paper [16].

In this paper, we deal with a specific subset of LTL, called *Generalized Reactivity (1)* (GR(1)), which is largely employed in controller synthesis [12]. This subset makes use of the operators $\mathbf{G}$ ("always"), which states that its operand formula must hold at each step of a valuation sequence, $\mathbf{F}$ ("eventually"), which requires its operand formula to hold at some point in the sequence, and $\mathbf{X}$ ("next"), which states that the operand formula must hold in the state following the one on which the formula is evaluated.

A GR(1) formula over a set of variables $\mathcal{V}$ has the form $\phi = \phi^{\mathcal{E}} \to \phi^{\mathcal{S}}$, where $\phi^{\mathcal{E}}$ and $\phi^{\mathcal{S}}$ are conjunctions of the following units: (i) an *initial condition*, which is a pure Boolean expression over variables in $\mathcal{V}$, denoted by $B^{init}(\mathcal{V})$; (ii) one or more *invariants*, conditions of the form $\mathbf{G}B^{inv}(\mathcal{V} \cup \mathbf{X}\mathcal{V})$, where $B^{\mathrm{inv}}(\mathcal{V} \cup \mathbf{X}\mathcal{V})$

denotes a pure Boolean expression over the set of variables in $\mathcal{V}$ and the set of atoms obtained by prepending an $\mathbf{X}$ operator to each variable; and (*iii*) one or more *fairness conditions* of the form $\mathbf{GF}B^{fair}(\mathcal{V})$.

The semantics of GR(1), as of LTL, are formalized as ω-*words* over the alphabet $\Sigma = 2^{\mathcal{V}}$. The set of ω-words that satisfy a formula ϕ is a regular ω-language [43] denoted by $L(\phi)$.

4 Problem Statement

In this section, we present an axiomatization of weakness of an LTL formula. Hereafter, we denote the weakness measure of the LTL formula ϕ as $d(\phi)$: the higher this measure, the weaker ϕ is, i.e., ϕ_2 is weaker than ϕ_1 if $d(\phi_1) \leq d(\phi_2)$.

In settings such as [2,4,39], an LTL formula ϕ_2 is *weaker* than ϕ_1 if and only if $\phi_1 \rightarrow \phi_2$ is valid (that is, it is true for any ω-word). Semantically, this translates to language inclusion: namely, ϕ_2 is weaker than ϕ_1 iff $L(\phi_1) \subseteq L(\phi_2)$. This gives us the first axiom of weakness.

Axiom 1 *Given two LTL formulae ϕ_1 and ϕ_2, if $\phi_1 \rightarrow \phi_2$, then $d(\phi_1) \leq d(\phi_2)$.*

Notice that this criterion defines a partial ordering of specifications: if none of the two formulae implies the other, those are incomparable according to this criterion. However, even for the incomparable case it may be useful to define a preference criterion.

Consider the simple case of two invariants over $\mathcal{V} = \{a, b, c\}$, $\phi_1 = \mathbf{G}(a \wedge b)$ and $\phi_2 = \mathbf{G}c$. Even if the two formulae are incomparable according to implication, i.e., neither one implies the other, it is clear that ϕ_1 allows in some sense fewer behaviors than ϕ_2: at each time step, the former allows for 2 distinct valuations of $\mathcal{V}$ while ϕ_2 allows 4 of them.

Consider the formulae $\phi_3 = \mathbf{G}(a \rightarrow \mathbf{X}b)$ and $\phi_4 = \mathbf{G}((a \wedge b) \rightarrow \mathbf{X}c)$ instead. Despite neither implying the other, we note that ϕ_3 is more restrictive than ϕ_4 asymptotically: that is, for a large enough n, the number of finite prefixes of length n that satisfy ϕ_3 is less than the number of finite prefixes of length n satisfying ϕ_4 ($\#(L(\phi_3)) < \#(L(\phi_4))$). This can be easily understood if one considers that ϕ_3 poses a restriction to the next symbol in an ω-word whenever a is true (which holds in 4 out of 8 possible valuations of $\mathcal{V}$), while ϕ_4 poses a similar restriction when $a \wedge b$ holds (in 2 out of the 8 valuations).

This means that weakness of a formula should be formalized, in addition to Axiom 1, in terms of the number of finite prefixes it allows. Formally:

Axiom 2 *Given two LTL formulae ϕ_1 and ϕ_2, ϕ_2 is said to be weaker than ϕ_1 if there exists some length $\bar{n}$ such that, for every $n > \bar{n}$, the set of prefixes of length n in $L(\phi_2)$ contains more elements than the set of prefixes of the same length in $L(\phi_1)$, i.e., if $\forall n > \bar{n}$, $\#(A_n(L(\phi_2))) \geq \#(A_n(L(\phi_1)))$, then $d(\phi_1) \leq d(\phi_2)$.*

The final desirable property is that a weakness measure be at least as discriminating as implication in case one formula strictly implies the other.

Axiom 3 *Let ϕ_1 and ϕ_2 be such that $\phi_1 \to \phi_2$ is valid and $\phi_2 \to \phi_1$ is not. Then $d(\phi_1) < d(\phi_2)$.*

In the next section, we prove that our proposed weakness measure satisfies Axioms 1 and 2. We then show that, although our weakness measure is not guaranteed to satisfy Axiom 3 in general, we are able to guarantee so for a specific class of formulae.

5 Weakness Measure of GR(1) Formulae

Hausdorff dimension and *Hausdorff measure* are basic concepts in fractal geometry and represent a way to define measures of extension—that is, analogous concepts to length, area, volume from classical geometry—for fractals [34]. Staiger [41] pinpointed a homeomorphism between fractals and regular ω-languages and proposed an analogous interpretation of the two quantities as extension measures of ω-languages. Intuitively, given an ω-language L, its Hausdorff dimension quantifies the growth rate of the number of distinct n-long prefixes of words in the language, over the length n of those prefixes. This makes it a good candidate for quantifying weakness: the less constrained the language is, the more prefixes of a fixed length are contained in it, implying a higher Hausdorff dimension.

The formal definition of Hausdorff dimension is tightly related to the notion of Hausdorff measure. The following definitions are given in [40].

Definition 1 (α-dimensional Hausdorff outer measure). *Given a regular ω-language L over an alphabet Σ with cardinality r, and a nonnegative real value α, the* α-dimensional Hausdorff outer measure *of L is defined as*

$$m_\alpha(L) = \lim_{n\to\infty} \inf_{V\in\mathcal{L}_n} \sum_{v\in V} r^{-\alpha|v|} \tag{1}$$

where $\mathcal{L}_n = \{V \subseteq \Sigma^* \mid V \cdot \Sigma^\omega \supseteq L \text{ and } |v| \geq n \text{ for all } v \in V\}$ is the collection of languages V containing finite words of length at least n and such that every word in L has at least a prefix in V. □

Definition 2 (Hausdorff dimension and measure). *Given an ω-language L, its* Hausdorff dimension*, denoted by* $\dim(L)$*, is the (unique) value $\bar{\alpha}$ such that*

$$m_\alpha(L) = \infty \;\; \alpha < \bar{\alpha}$$
$$m_\alpha(L) = 0 \;\; \alpha > \bar{\alpha}$$

The value $m_{\dim(L)}(L)$ is called the Hausdorff measure *of L.* □

In other words, Hausdorff measure is the limit of the process of approximating the ω-language L by a set V of finite prefixes with length at least n, and weighing each prefix with a quantity $r^{-\alpha|v|}$ that decreases as the prefix length increases. This limit can be finite and positive for at most one value of the α parameter. This value is called *Hausdorff dimension.*

A related concept appearing in the literature is entropy:

Definition 3 (Entropy [34]). *Given an ω-language $L \subseteq \Sigma^\omega$ over an alphabet of size r, the* entropy *of L is $H(L) = \limsup_{n\to\infty} \frac{1}{n} \log_r \#(A_n(L))$.*

It has been proved [34] that the Hausdorff dimension has a close relationship with the notion of entropy: Specifically, we have $\dim(L) \leq H(L)$ in general, and $\dim(L) = H(L)$ if L is a closed ω-language. Details on how entropy is computed are given in [16].

When L is not closed, the general algorithm presented in [40,41] provides a more refined intuition of what is actually quantified by Hausdorff dimension, which distinguishes it from entropy. The algorithm is based on computing a Muller automaton $\mathcal{M}_L$ accepting L with set of accepting state sets T_L. For each accepting set $S' \in T_L$ and for each state $s \in S'$, consider the ω-language $C_{S'}$ consisting of all the infinite paths in $\mathcal{M}_L$ starting from s and visiting no states outside S'. It can be shown that this language is closed and its entropy $H(C_{S'})$ is independent of the choice of s [40]. The Hausdorff dimension of L is then

$$\dim(L) = \max_{S' \in T_L} H(C_{S'}) \,. \tag{2}$$

Hausdorff dimension provides an ordering consistent with the weakness notion defined in Sec. 4. We can interpret it as a measure of the asymptotic degrees of freedom of an ω-language: it quantifies how many different evolutions are allowed to an ω-word once its run remains in an accepting subset of the Muller automaton. The example below shows how it differs from entropy.

Example 1. Consider the LTL formula $\phi_1 = \mathbf{FG}a$ over the variable set $\mathcal{V} = \{a\}$ whose Muller automaton is shown in Fig. 1. The accepting sets to which a state belongs are enclosed in curly braces.

Notice that for any $w \in L(\phi_1)$ both valuations of $\mathcal{V}$ are allowed until w reaches the accepting state, and the satisfaction of $\mathbf{G}a$ may be delayed arbitrarily. Therefore, for any finite n, $\#(A_n(L)) = 2^n$, and thereby $H(L(\phi_1)) = 1$.

In this simple DMA, there is only one accepting singleton $\{s_2\}$. Therefore, there is only one $C_{S'} = \{\{a\}^\omega\}$ which allows only the symbol $\{a\} \in 2^{\mathcal{V}}$. This implies $\#(A_n(C_{S'})) = 1$. The Hausdorff dimension is $\dim(L(\phi_1)) = H(C_{S'}) = 0$. This example demonstrates that the Hausdorff dimension isolates the asymptotic behaviour of $L(\phi_1)$ as it depends only on the condition $\mathbf{G}a$ that is eventually satisfied by any ω-word in the ω-language. □

The following theorem shows that Hausdorff dimension is consistent with implication (hence satisfying Axiom 1).

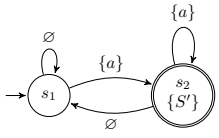

Fig. 1: DMA of $L(\phi_1)$.

Theorem 1. *Given two LTL formulae ϕ_1 and ϕ_2 such that $\phi_1 \to \phi_2$ is valid, then* $\dim(L(\phi_1)) \leq \dim(L(\phi_2))$.

Proof. This follows from the language inclusion $L(\phi_1) \subseteq L(\phi_2)$ and the monotonicity of Hausdorff dimension with respect to language inclusion [34].

Note that Theorem 1 does not exclude the situation where one formula strictly implies another, but the two languages have the same Hausdorff dimension, thus violating Axiom 3. We investigate under which conditions this holds in the context of GR(1) formulae and provide a refined weakness measure that bounds the number of cases in which it can happen.

To this end, in what follows, we introduce a new weakness measure for GR(1) based on Hausdorff dimension. We first analyse the dimension of invariants. We then show that under the condition of strong connectedness, it is possible to distinguish between weaker and stronger invariants, in the implication sense (Sec. 5.1). We show how, under the same condition, this measure fails to capture the impact of conjoining a fairness condition (Sec. 5.2). To overcome this, we define a refined weakness measure for GR(1) formulae that comprises two components: the Hausdorff dimension (i) of the whole formula and (ii) of the difference language between the invariant and the fairness conditions (Sec. 5.3).

5.1 Dimension of Invariants

Consider the formula $\phi^{inv} = \mathbf{G}B(\mathcal{V} \cup \mathbf{X}\mathcal{V})$. The ω-language $L(\phi^{inv})$ is closed. Hence, the Hausdorff dimension of $L(\phi^{inv})$ coincides with its entropy $H(L(\phi^{inv}))$ and can be computed as the maximum eigenvalue of the adjacency matrix of its Büchi automaton (see [16]). From this equivalence and Definition 3, it is easy to see that in this case Hausdorff dimension satisfies Axiom 2. In general, Theorem 1 may hold for invariants where one is strictly weaker than the other and both have equal dimensions as demonstrated in the following.

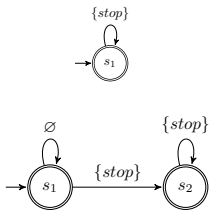

Fig. 2: DBAs of ϕ_1^{inv} (top) and ϕ_2^{inv} (bottom)

Example 2. Consider the variable set $\mathcal{V} = \{stop\}$ and the formulae $\phi_1^{inv} = \mathbf{G}stop$ and $\phi_2^{inv} = \mathbf{G}(stop \rightarrow \mathbf{X}stop)$. Their Büchi automata are shown in Fig. 2. Clearly $\phi_1^{inv} \rightarrow \phi_2^{inv}$ strictly, however the two languages have the same Hausdorff dimension $\dim\left(L(\phi_1^{inv})\right) = \dim\left(L(\phi_2^{inv})\right) = 0$.

There exists, however, a subclass of invariants for which the dimension is strictly monotonic with respect to implication. This subclass is characterized through the concept of *strong connectedness* of an ω-language. Hereafter, given a word $w \in A(L)$, we denote by $\mathrm{S}_w(L)$ the ω-language formed by the ω-words v such that $wv \in L$ (that is, the suffixes allowed in L after reading w).

Definition 4 (Strongly connected ω-language [34]). *An ω-language L is* strongly connected *if for every prefix $w \in A(L)$ there exists a finite word $v \in \Sigma^*$ such that $\mathrm{S}_{wv}(L) = L$.*

In other words, an ω-language is strongly connected if and only if there exists a strongly connected finite-state automaton which represents it [34], i.e., an automaton such that given any pair of states, each of them is reachable from the

other. Using this notion, in the next theorem we provide a sufficient condition over invariants for Axiom 3 to be satisfied (the proof is relegated to [16]):

Theorem 2. *Let* $\phi_1^{inv} = \mathbf{G}B_1(\mathcal{V} \cup \mathbf{X}\mathcal{V})$ *and* $\phi_2^{inv} = \mathbf{G}B_2(\mathcal{V} \cup \mathbf{X}\mathcal{V})$ *be two non-empty invariants such that* $\phi_1^{inv} \rightarrow \phi_2^{inv}$ *is valid,* $\phi_2^{inv} \rightarrow \phi_1^{inv}$ *is not valid and* ϕ_2^{inv} *is strongly connected. Then* $\dim\left(L(\phi_1^{inv})\right) < \dim\left(L(\phi_2^{inv})\right)$.

An interesting kind of invariant that falls in this class is the *one-state invariant*, one that does not use the $\mathbf{X}$ operator: $\phi_s^{inv} = \mathbf{G}B(\mathcal{V})$ whose DBA is shown in Fig. 3. (For succinctness, the set of valuations that label a transition between the same states is denoted by the Boolean expression characterizing it.) In this case, the Hausdorff dimension has a closed form:

Fig. 3: DBA of a one-state invariant.

$$\dim\left(\phi_s^{inv}\right) = \log_r \#(B(\mathcal{V}))$$

where $r = 2^{\#(\mathcal{V})}$ is the number of valuations of $\mathcal{V}$ and $\#(B(\mathcal{V}))$ is the number of valuations that satisfy $B(\mathcal{V})$. Invariants of this type are clearly strongly connected and satisfy Theorem 2.

Remark 1. Typical examples of GR(1) specifications manually produced, like those of device communication protocols, make use of strongly connected environment assumptions. It is indeed natural to allow environments to be reset to their initial state after some steps. However, when specifications contain "until" operators or response patterns, the procedure to convert them into GR(1) [33] may yield assumptions which are no longer strongly connected. In those cases, a problem similar to that of Example 2 may arise. □

5.2 Fairness and Fairness Complements

Consider the generic fairness condition $\phi^{fair} = \mathbf{GF}B(\mathcal{V})$ whose DBA is shown in Fig. 4. This language is not closed: take a symbol $x \in \Sigma$ that does not satisfy $B(\mathcal{V})$ and the ω-word x^ω consisting of infinite repetitions of this symbol. It is clear that $A(\{x^\omega\}) \subseteq A(L(\phi^{fair}))$, but $x^\omega \notin L(\phi^{fair})$. We apply the algorithm in Sec. 5 (cf. equation 2) for non-closed languages. A DMA for $L(\phi^{fair})$ can be obtained from the top DBA in Fig. 4: the accepting sets are $S'_1 = \{q_1, q_2\}$ and $S'_2 = \{q_2\}$. It is easy to see that $H(C_{S'_1}) = 1$ and $H(C_{S'_2}) = \log_r \#(B(\mathcal{V})) \leq 1$. Therefore, $\dim\left(L(\phi^{fair})\right) = 1$, independently of $B(\mathcal{V})$. We conclude that fairness conditions are indistinguishable from the *true* constant, which also has dimension 1. To allow for a distinction to be made, we characterize the negation of such formula. We call an LTL formula of the kind $\phi^{cfair} = \mathbf{FG}\neg B(\mathcal{V})$ a *fairness complement*. The

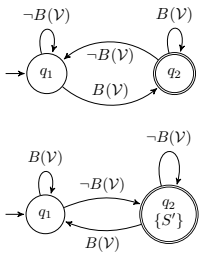

Fig. 4: DBA of $L(\phi^{fair})$ (top) and DMA of $L(\phi^{cfair})$ (bottom).

DMA of $L(\phi^{cfair})$ is shown in the bottom of Fig. 4. The only accepting set is $S' = \{q_2\}$. (Notice that unlike the top one, this automaton accepts only words that stay forever in q_2 from a certain step on.) The language $C_{S'}$ (see Sec. 5) has an entropy of $\log_r \#(\neg B(\mathcal{V}))$. Hence

$$\dim\left(L(\phi^{cfair})\right) = \log_r \#(\neg B(\mathcal{V}))$$

where $r = 2^{\#(\mathcal{V})}$. Notice that $C_{S'}$ is the language of the formula $\mathbf{G}\neg B(\mathcal{V})$, which is an "asymptotic" condition of ϕ^{cfair}. As observed previously, Hausdorff dimension is strictly monotonic for one-state invariants. Therefore, the weakness of fairness complements can be ranked in terms of the Hausdorff dimension, allowing to compare fairness conditions as follows:

Theorem 3. *Let ϕ_1^{fair} and ϕ_2^{fair} be two fairness conditions such that $\phi_1^{fair} \rightarrow \phi_2^{fair}$ is valid and $\phi_2^{fair} \rightarrow \phi_1^{fair}$ is not. Then* $\dim\left(L(\neg\phi_1^{fair})\right) > \dim\left(L(\neg\phi_2^{fair})\right)$.

In other words, the stronger a fairness formula is, the weaker its complement and thereby the higher its dimension.

5.3 Dimension Pairs for GR(1) Formulae

Consider a generic GR(1) formula $\phi = \phi^{init} \wedge \phi^{inv} \wedge \bigwedge_{i=1}^{m} \phi_i^{fair}$. We show through an example that even when ϕ^{inv} is strongly connected, Hausdorff dimension may not distinguish between weaker and stronger fairness conditions in the implication sense (as also pointed out in [6]).

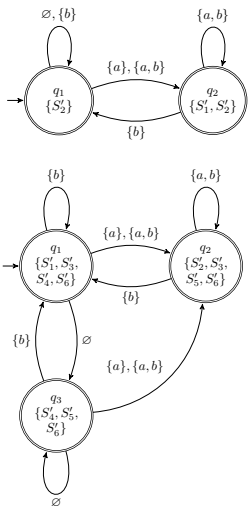

Fig. 5: DMAs of ϕ_1 (top) and ϕ_2 (bottom) of Example 3.

Example 3. Consider the two formulae over the variables $\mathcal{V} = \{a, b\}$: $\phi_1 = \mathbf{G}(a \rightarrow \mathbf{X}b) \wedge \mathbf{GF}a$ and $\phi_2 = \mathbf{G}(a \rightarrow \mathbf{X}b) \wedge \mathbf{GF}b$. The same invariant appears in both, and thereby have the same Hausdorff dimension, but the fairness condition in ϕ_2 is always satisfied when the fairness condition of ϕ_1 is satisfied, by virtue of the invariant itself. However, the ω-word $\{b\}^\omega$ satisfies ϕ_2 but not ϕ_1. So, ϕ_1 implies ϕ_2 but not vice versa.

The language of both formulae is not closed. The Muller automata of ϕ_1 and ϕ_2 are shown at the top and bottom, respectively, in Fig. 5. In both automata, there is an accepting set that covers the entire state space (S'_2 in $L(\phi_1)$ and S'_6 in $L(\phi_2)$). It is possible to show that the maximum $H(C_{S'})$ of equation (2) is achieved exactly for these accepting sets [9,34]. The ω-languages $C_{S'_2}$ in $L(\phi_1)$ and $C_{S'_6}$ in $L(\phi_2)$ both coincide with the language of the invariant alone. Therefore,

$$\dim(\phi_1) = \dim(\phi_2) = \dim\left(L(\mathbf{G}(a \rightarrow \mathbf{X}b))\right) .$$

To distinguish between the two formulae, we exploit the fact that the complement of a fairness condition is a formula of the kind $\mathbf{FG}B(\mathcal{V})$ which can be compared through Hausdorff dimension. Therefore, we propose a weakness measure which consists of two components: one relating to the whole formula and one measuring the ω-language excluded from the invariant by the addition of the fairness conditions.

Definition 5 (Weakness). *The* weakness *of a GR(1) formula* $\phi = (\phi^{init} \wedge \phi^{inv} \bigwedge_{i=1}^{m} \phi_i^{fair})$*, denoted by* $d(\phi)$*, is the pair* $(d_1(\phi), d_2(\phi))$ *such that* $d_1(\phi)$ *is the Hausdorff dimension of* $L(\phi)$*; and* $d_2(\phi)$ *is the Hausdorff dimension of* $L(\phi^c) = L(\phi^{init} \wedge \phi^{inv} \wedge \bigvee_{i=1}^{m} \phi_i^{cfair})$*, where* $\phi_i^{cfair} = \neg\phi_i^{fair}$*. The following partial ordering is defined based on the weakness measure: If* $d^i = (d_1^i, d_2^i)$*, with* $i \in 1, 2$ *are weakness measures for two GR(1) formulae, then* $d^1 < d^2$ *if* $d_1^1 < d_1^2$ *or* $d_1^1 = d_1^2$ *and* $d_2^1 > d_2^2$*.*

We apply below this weakness measure to the formulae in Example 3.

Example 4. To compute d_2, let us define $\phi_1^c = \mathbf{G}(a \rightarrow \mathbf{X}b) \wedge \mathbf{FG}\neg a$ and $\phi_2^c = \mathbf{G}(a \rightarrow \mathbf{X}b) \wedge \mathbf{FG}\neg b$. The DMAs of the resulting languages are shown respectively in Fig. 6. Each of them has just one accepting singleton, so the computation of the Hausdorff dimension is straightforward: $\dim(\phi_1^c) = \frac{1}{2}$ and $\dim(\phi_2^c) = 0$. In summary, since ϕ_1 is more restrictive than ϕ_2, the Hausdorff dimension of the ω-language cut out by $\mathbf{GF}a$ is higher than the Hausdorff dimension of the behaviours excluded by $\mathbf{GF}b$.

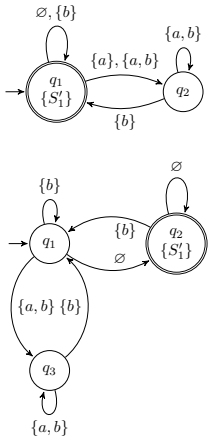

Fig. 6: DMAs of ϕ_1^c (top) and ϕ_2^c (bottom) of Example 4.

The following Theorem justifies the use of this dimension pair for weakness quantification when the formulae have the same invariant.

Theorem 4. *Let* $\phi_1 = \phi^{inv} \wedge \bigwedge_{i=1}^{m} \phi_{1,i}^{fair}$ *and* $\phi_2 = \phi^{inv} \wedge \bigwedge_{j=1}^{l} \phi_{2,j}^{fair}$*, such that* $\phi_1 \rightarrow \phi_2$ *is valid. Then* $d_1(\phi_1) = d_1(\phi_2)$ *and* $d_2(\phi_1) \geq d_2(\phi_2)$*.*

Proof. Since ϕ_1 implies ϕ_2, $L(\phi_1) \subseteq L(\phi_2)$. Furthermore, for $i = 1, 2$, $L(\phi_i) = L(\phi^{inv}) \cap L(\bigwedge_{j=1}^{m} \phi_{i,j}^{fair})$. Hence, $L(\phi^{inv}) \backslash L(\bigwedge_{j=1}^{m} \phi_{1,j}^{fair}) \supseteq L(\phi^{inv}) \backslash L(\bigwedge_{j=1}^{l} \phi_{2,j}^{fair})$, i.e., $L(\phi_1^c) \supseteq L(\phi_2^c)$. Then, by monotonicity, $\dim(\phi_1^c) \geq \dim(\phi_2^c)$, finishing the proof. □

Therefore, given two formulae with the same invariant, we deem the formula with lower d_2 weaker.

Regarding formulae with the same d_1 and different invariants, we justify heuristically the same order relation. We first note that the Hausdorff dimension

of a countable union of ω-languages, as noted in [41], is

$$\dim\left(\bigcup_i L_i\right) = \sup_i \dim\left(L_i\right).$$

This property is known as the *countable stability* of Hausdorff dimension. This implies that for any formula ϕ, if $d_2(\phi) \leq d_1(\phi)$ then

$$\dim\left(L(\phi^{inv})\right) = \dim\left(L(\phi) \cup L(\phi^c)\right) = \dim\left(L(\phi)\right) .$$

So, if for two formulae, ϕ_1 and ϕ_2, we have $d_1(\phi_1) = d_1(\phi_2) > d_2(\phi_1) > d_2(\phi_2)$, then this can be interpreted as the two invariants having the same dimension and the fairness condition of ϕ_1 removing more behaviours than the fairness condition of ϕ_2. In this sense, ϕ_2 is weaker than ϕ_1. This justifies intuitively our weakness definition and the associated partial ordering. In Sec. 6, we illustrate applications of this order relation for comparing GR(1) assumptions.

The computation of $d_2(\phi)$ for a generic ϕ with m fairness conditions can be reduced to the case of a single fairness condition. Based on the countable stability of Hausdorff dimension, we have

$$d_2(\phi) = \sup_{i=1,\ldots,m} d_2(\phi^{init} \wedge \phi^{inv} \wedge \phi_i^{cfair}) .$$

Furthermore, the case of a single fairness condition can be further reduced to computing the Hausdorff dimension of an invariant by the following theorem.

Theorem 5. *Given a formula* $\phi^c = \mathbf{G}B^{inv}(\mathcal{V} \cup \mathbf{X}\mathcal{V}) \wedge \mathbf{FG}\neg B^{fair}(\mathcal{V})$ *we have*

$$\dim\left(L(\phi^c)\right) = \dim\left(L(\mathbf{G}(B^{inv} \wedge \neg B^{fair}))\right) .$$

P*roof sketch (full proof is presented in [16]).* Since $L(\phi^c)$ is not closed, the Hausdorff dimension must be computed from a DMA. The proof (given in [16]) consists in showing that the DMA's accepting subsets correspond to the automaton of an ω-language where both B^{inv} and $\neg B^{fair}$ are satisfied at every step. This property is a generalization of the observation made in Sec. 5.2 about the Hausdorff dimension of fairness complements. □

5.4 Initial Conditions

Consider $\phi^{init} = B(\mathcal{V})$. An expression of this form constrains only the first symbol of the ω-words in $L(\phi^{init})$. For the same reason as ϕ^{fair} in Sec. 5.2, $L(\phi^{init})$ is closed, and therefore its dimension can be computed via its entropy. By applying the definition of entropy, it is easy to see that, similarly to the unconstrained language $L(true)$, $\dim\left(L(\phi^{init})\right) = 1$.

Consider now a formula $\phi = \phi^{init} \wedge \phi^{inv}$. A DBA $\mathcal{B}$ for $L(\phi)$ can be computed from a DBA $\mathcal{B}_{inv}$ of $L(\phi^{inv})$ by removing all transitions starting from its initial state whose labels do not satisfy $B(\mathcal{V})$. The resulting automaton may leave out

parts of $\mathcal{B}_{inv}$ that are no longer reachable from the initial state. This does not happen if $L(\phi^{inv})$ is strongly connected, as in that case any non-initial state in $\mathcal{B}_{inv}$ is reachable from any other state. In this case

$$\dim(\phi) = \dim\left(\phi^{inv}\right) .$$

This implies that the initial conditions do not affect the Hausdorff dimension and hence cannot be always ordered by our weakness measure. This is acceptable since typically, in applications like assumptions refinement, the focus is in assessing invariants or fairness conditions rather than initial conditions [30].

6 Evaluation

We evaluate here our proposed weakness measure through applications to benchmarks within the assumptions refinement domain, demonstrating its usefulness in distinguishing weakness of different formulae, and discussing the computation time bottlenecks. In [16] we report on our evaluation within another application domain, namely quantitative model checking.

To this aim, we implemented the weakness measure computation for GR(1) specifications in Python 2.7 and made it publicly available in [1]. Our implementation makes use of the Spot tool [22] for LTL-to-automata conversion. We integrated the weakness computation algorithm within two state-of-the-art counterstrategy-guided assumptions refinement approaches [4,15] (the implementations are available in [1]). The outcome of such approaches is a *refinement tree*, a tree structure where each node is associated with a GR(1) formula consisting of a conjunction of environment assumptions; if we denote by ϕ a formula associated with a node, the node's children are of the form $\phi \wedge \psi$, where ψ is a single initial condition, invariant, or fairness condition. Since the goal of such procedures is identifying weakest formulae that describe an environment, our weakness measure can be used to provide a preference ranking of the tree nodes.

We conducted experiments on two benchmarks for GR(1) assumptions refinement, namely the specifications of a lift controller and of the AMBA-AHB protocol for device communications in its versions for two, four and eight master devices [4,12,30]. The lift controller example specifies a controller for a lift with three floors: the Boolean variable b_i denotes the state of the button on floor i; the Boolean variable f_i is true iff the lift is at floor i. For more details on the initial assumptions $\phi^{\mathcal{E}}$ see [4]. The AMBA-AHB protocol provides signals for requesting access to a bus ($hbusreq_i$), for granting access ($hgrant_i$), for signalling the termination of a communication (*hready*), and for identifying the current owner of the bus (*hmaster*). Other signals are detailed in [12]. To our knowledge, the AMBA08 specification is one of the biggest benchmarks available in the field.

In the followings we focus on examples taken from [4,15], and discuss three cases highlighting features of our weakness measure: (*i*) in the first example, we demonstrate the relationship between weakness and implication; (*ii*) second, we consider cases when two formulae are not comparable by implication but can be

ranked with our measure; and (*iii*) we discuss the case of formulae equally constraining the environment, which have equal ranking according to our measure. We refer the reader to [1] for the complete results.

Relation between weakness and implication. Consider the lift controller example. Two refinements computed by the automated approach in [15] are: $\phi_1 = \mathbf{G}((\neg b_1 \wedge \neg b_2 \wedge \neg b_3) \rightarrow \mathbf{X}(b_1 \vee b_2 \vee b_3))$; and $\phi_2 = \mathbf{GF}(b_1 \vee b_2 \vee b_3)$. The first forces one of the buttons to be pressed at least every second step in a behaviour. The second forces one of the buttons to be pressed infinitely often in a behaviour. It is clear that ϕ_1 implies ϕ_2. We compare the assumptions obtained by refining the original assumptions with the first one and with the second one: $d(\phi^{\mathcal{E}} \wedge \phi_1) = (0.7746, 0)$ and $d(\phi^{\mathcal{E}} \wedge \phi_2) = (0.7925, 0.5)$. Notice that $d_1(\phi^{\mathcal{E}} \wedge \phi_1) < d_1(\phi^{\mathcal{E}} \wedge \phi_2)$ and this is consistent with the fact that ϕ_1 is stronger than ϕ_2. Consider now the two fairness refinements: $\phi_2 = \mathbf{GF}(b_1 \vee b_2 \vee b_3)$; and $\phi_3 = \mathbf{GF}b_1$. We have $d(\phi^{\mathcal{E}} \wedge \phi_2) = (0.7925, 0.5)$ and $d(\phi^{\mathcal{E}} \wedge \phi_3) = (0.7925, 0.695)$. Here, d_1 is equal for both formulae and $d_2(\phi^{\mathcal{E}} \wedge \phi_2) < d_2(\phi^{\mathcal{E}} \wedge \phi_3)$; this is consistent with the fact that ϕ_2 is weaker than ϕ_3.

Formulae incomparable via implication. Consider ϕ_3 above and $\phi_4 = \mathbf{GF}(b_2 \vee b_3)$. Neither implies the other. However, it is reasonable to argue that ϕ_4 is less restrictive than ϕ_3: while ϕ_3 constrains exactly one button to be pressed infinitely often, ϕ_4 allows the extra choice of which one (out of two) . This intuition is indeed reflected by our computed weakness metric: $d(\phi^{\mathcal{E}} \wedge \phi_3) = (0.7925, 0.695)$ and $d(\phi^{\mathcal{E}} \wedge \phi_4) = (0.7925, 0.5975)$. This expresses the notion that ϕ_4 removes less behaviours from $\phi^{\mathcal{E}}$ than ϕ_3.

Our weakness measure can help in spotting asymmetries between assumptions that are syntactically equal but constrain semantically different variables. Consider an extended version of the lift controller example given in [16], including the input variable *alarm* and the output variable *stop*: whenever *alarm* is set to high, the lift enters a *stop* state where it does not move from the floor it is at. Computing the weakness of the two refinements $\phi_5 = \mathbf{G}\neg b_1$ and $\phi_6 = \mathbf{G}\neg alarm$ yields $d(\phi^{\mathcal{E}} \wedge \phi^{\mathcal{S}} \wedge \phi_5) = (0.3694, 0.3207)$ and $d(\phi^{\mathcal{E}} \wedge \phi^{\mathcal{S}} \wedge \phi_6) = (0.3746, 0.3346)$. This is consistent with the intuition that the former assumption excludes a part of the desirable system behaviors (all the ones that allow it to reach floor 1), while the latter excludes only the traces ending in the *stop* state, being then a weaker restriction on the combined behaviors of the controller and the environment.

The following two assumptions refinements are computed for the AMBA-AHB case study with two masters: $\psi_1 = \mathbf{G}(\neg hbusreq_1 \vee \mathbf{X}(hready \vee \neg hbusreq_1))$; and $\psi_2 = \mathbf{G}((\neg hgrant_1 \wedge hready \wedge hbusreq_1) \rightarrow \mathbf{X}(\neg hready \vee \neg hbusreq_1))$. As in the case of the lift example, neither formula implies the other. The weakness of the resulting assumptions is: $d(\psi^{\mathcal{E}} \wedge \psi_1) = (0.9503, 0.9068)$ and $d(\psi^{\mathcal{E}} \wedge \psi_2) = (0.9607, 0.9172)$. The refinement ψ_2 is weaker than ψ_1. Such insight into their weakness could be used to guide the refinement approach (e.g., [4,15]) in choosing to only refine those assumptions that may lead to weaker specifications, for instance further refining ψ_2 rather than ψ_1.

Consistency between equally constraining formulae. Let us consider the AMBA-AHB protocol with eight masters and the two alternative refinements:

$\theta_1 = \mathbf{GF}(hmaster_0 \vee \neg hbusreq_1)$; and $\theta_2 = \mathbf{GF}(hmaster_1 \vee \neg hbusreq_2)$. Clearly the two alternatives express the same kind of constraint on different masters. Since the two masters do not have priorities over each other, expectedly the two refinements have the same weakness: $d(\theta^{\mathcal{E}} \wedge \theta_1) = d(\theta^{\mathcal{E}} \wedge \theta_2) = (0.9396, 0.9214)$.

Performance. In order to compare the discriminative power of the weakness measure and implication, we perform an experiment where every pair of refinements from the trees in [15] is compared via both methods. An implication check for the pair of formulae ϕ_1 and ϕ_2 is performed by computing the nondeterministic transition-based generalized Büchi automata (TGBA) [32] of the formulae $\phi_1 \wedge \neg\phi_2$ and $\phi_2 \wedge \neg\phi_1$, and checking whether any of them is empty [38].

We compare the proportion of formulae pairs that have different weakness measure (and thereby can be discriminated via our proposed metric) and the proportion of formulae pairs where one formula strictly implies the other (that can be discriminated via logical implication). Table 1 shows the results: the columns show the total number of nodes in the refinement tree (**#Nodes**), the corresponding number of pairs (**#Pairs**), the percentage of pairs that can be discriminated via implication (**%Impl**) and via weakness (**%Weak**). The table shows that, despite weakness does not capture implication in all cases, it still allows for the discrimination of a larger set of assumptions, by virtue of Axiom 2.

Table 1: Discriminative power of implication and weakness

Case study	**#Nodes (k)**	**#Pairs**	**%Impl**	**%Weak**
AMBA02	9	36	63.9	88.9
AMBA04	17	136	69.1	79.4

The time taken to compute the weakness measure for each refinement (computed via the approach in [15]) was consistently less than 1 minute for the lift controller, AMBA02, and AMBA04 case studies. The time needed on a representative subset of refinements from the AMBA08 example is shown in Fig. 7 as a function of the number of GR(1) conjuncts in the assumptions. The subset comprises a path from the root of the refinement tree (initial assumptions) to one of the 80 leaves. We observed that 79 of the 80 leaves showed similar performance as the one reported in figure; one of them, instead, took around 5200s. Notice that over 99% of the time is spent on DMA computation, and the remaining time is employed on eigenvalue computation.

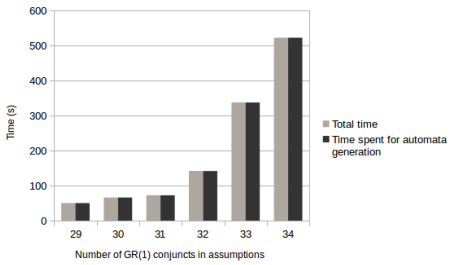

Fig. 7: Execution time of weakness computation for AMBA08

In general implication checks require an $O(k^2)$ number of automata computations. On the other hand, for a set of formulae containing at most m fairness

conditions, our weakness measure requires $m + 1$ DMA computations, yielding $O(mk)$ automata for comparing k formulae. In this respect, the advantage of our weakness measure resides in the reduced number of DMA computations with respect to implication.

The price to pay lies in the complexity of the needed automata: while weakness requires deterministic automata, implication can be checked via nondeterministic ones, which are typically faster to compute [23]. However, in the AMBA08 case we observed that the quadratic growth of implication checks prevailed over the lesser complexity of nondeterministic automata: the value of k for this case study is 158; while computing all weakness values for the refinement tree required a total time of 15 hours, in the same amount of time only a small fraction of the 12,403 formulae pairs could be checked for implication.

7 Conclusion

In this paper we proposed a new measure for assessing the weakness of GR(1) formulae quantitatively and demonstrated its application in the context of weakest assumptions refinement for GR(1) controller synthesis. We showed that strong connectedness of invariants is a sufficient requirement to guarantee that our measure distinguishes between stronger and weaker formulae in the implication sense. We introduced a component to the measure which allows one to compare formulae with the same dimension based on the weakness of their fairness conditions. The major limitation of the approach is the need for deterministic automata to be produced, which induces high computation time because of the determinization process [23].

As part of our future work, we plan to explore the possibility of refining the weakness relation by including Hausdorff measure in the definition, since Hausdorff measure can distinguish between stronger and weaker ω-languages in case they are not strongly connected [34]. We also intend to investigate algorithms for computing—or approximating at a controlled accuracy—Hausdorff dimension on nondeterministic automata.

Acknowledgments The support of the EPSRC HiPEDS Centre for Doctoral Training (EP/L016796/1) is gratefully acknowledged. We also thank our reviewers for their insightful comments and suggestions.

References

1. `https://gitlab.doc.ic.ac.uk/dgc14/WeakestAssumptions`
2. Albarghouthi, A., Dillig, I., Gurfinkel, A.: Maximal specification synthesis. ACM SIGPLAN Notices 51(1), 789–801 (2016)
3. Almagor, S., Avni, G., Kupferman, O.: Automatic Generation of Quality Specifications. In: Computer Aided Verification. pp. 479–494 (2013)
4. Alur, R., Moarref, S., Topcu, U.: Counter-strategy guided refinement of GR(1) temporal logic specifications. In: Formal Methods in Computer-Aided Design. pp. 26–33 (2013)

5. Alur, R., Moarref, S., Topcu, U.: Pattern-Based Refinement of Assume-Guarantee Specifications in Reactive Synthesis. In: Tools and Algorithms for the Construction and Analysis of Systems. pp. 501–516 (2015)
6. Asarin, E., Blockelet, M., Degorre, A.: Entropy model checking. In: 12th Workshop on Quantitative Aspects of Programming Languages - Joint with European Joint Conference On Theory and Practice of Software (2014)
7. Asarin, E., Blockelet, M., Degorre, A., Dima, C., Mu, C.: Asymptotic behaviour in temporal logic. In: Joint Meeting CSL/LICS. pp. 1–9. ACM Press (2014)
8. Barnat, J., Bauch, P., Beneš, N., Brim, L., Beran, J., Kratochvíla, T.: Analysing sanity of requirements for avionics systems. Form. Asp. Comput. 28(1), 45–63 (2016)
9. Berman, A., Plemmons, R.: Nonnegative Matrices in the Mathematical Sciences. Society for Industrial and Applied Mathematics (1994)
10. Bloem, R., Chatterjee, K., Henzinger, T.A., Jobstmann, B.: Better Quality in Synthesis through Quantitative Objectives. In: Computer Aided Verification, pp. 140–156. Springer, Berlin, Heidelberg (2009)
11. Bloem, R., Galler, S., Jobstmann, B., Piterman, N., Pnueli, A., Weiglhofer, M.: Specify, Compile, Run: Hardware from PSL. Electronic Notes in Theoretical Computer Science 190(4), 3–16 (2007)
12. Bloem, R., Jobstmann, B., Piterman, N., Pnueli, A., Sa'Ar, Y.: Synthesis of Reactive(1) designs. Journal of Computer and System Sciences 78(3), 911–938 (2012)
13. Braberman, V., D'Ippolito, N., Piterman, N., Sykes, D., Uchitel, S.: Controller synthesis: From modelling to enactment. In: International Conference on Software Engineering. pp. 1347–1350. IEEE (2013)
14. Cassandras, C.G., Lafortune, S.: Introduction to Discrete Event Systems. Springer (2008)
15. Cavezza, D.G., Alrajeh, D.: Interpolation-Based GR(1) Assumptions Refinement. In: Tools and Algorithms for the Construction and Analysis of Systems. pp. 281–297 (2017)
16. Cavezza, D.G., Alrajeh, D., György, A.: A Weakness Measure for GR(1) Formulae. CoRR abs/1805.03151 (2018), `http://arxiv.org/abs/1805.03151`
17. Chatterjee, K., De Alfaro, L., Faella, M., Henzinger, T.A., Majumdar, R., Stoelinga, M.: Compositional quantitative reasoning. In: International Conference on the Quantitative Evaluation of Systems. pp. 179–188 (2006)
18. Chatterjee, K., Henzinger, T.A., Jobstmann, B.: Environment Assumptions for Synthesis. In: International Conference on Concurrency Theory. pp. 147–161 (2008)
19. Cimatti, A., Roveri, M., Schuppan, V., Tchaltsev, A.: Diagnostic Information for Realizability. In: International Conference on Verification, Model Checking, and Abstract Interpretation. pp. 52–67 (2008)
20. Cobleigh, J.M., Giannakopoulou, D., Păsăreanu, C.S.: Learning Assumptions for Compositional Verification. In: Tools and Algorithms for the Construction and Analysis of Systems. pp. 331–346 (2003)
21. D'Ippolito, N., Braberman, V., Sykes, D., Uchitel, S.: Robust degradation and enhancement of robot mission behaviour in unpredictable environments. In: Proceedings of the 1st International Workshop on Control Theory for Software Engineering. pp. 26–33 (2015)
22. Duret-Lutz, A., Lewkowicz, A., Fauchille, A., Michaud, T., Renault, E., Xu, L.: Spot 2.0 — a framework for LTL and ω-automata manipulation. In: Automated Technology for Verification and Analysis. vol. 9938, pp. 122–129. Springer (2016)
23. Esparza, J., Ketínský, J., Sickert, S.: From LTL to deterministic automata. Formal Methods in System Design 49(3), 219–271 (2016)

24. Hansson, H., Jonsson, B.: A logic for reasoning about time and reliability. Formal Aspects of Computing 6(5), 512–535 (1994)
25. Henzinger, T.: From Boolean to quantitative notions of correctness. ACM SIGPLAN Notices 45(1), 157 (2010)
26. Henzinger, T.A., Otop, J.: From Model Checking to Model Measuring. In: International Conference on Concurrency Theory, pp. 273–287. Springer (2013)
27. Konighofer, R., Hofferek, G., Bloem, R.: Debugging formal specifications using simple counterstrategies. In: Formal Methods in Computer-Aided Design. pp. 152–159 (2009)
28. Kupferman, O.: Recent challenges and ideas in temporal synthesis. In: Proceedings of the 38th International Conference on Current Trends in Theory and Practice of Computer Science. pp. 88–98 (2012)
29. Kwiatkowska, M.: Quantitative verification: Models, Techniques and Tools. In: Joint Meeting on Foundations of Software Engineering - ESEC/FSE 2015. p. 449. ACM Press (2007)
30. Li, W., Dworkin, L., Seshia, S.A.: Mining assumptions for synthesis. In: International Conference on Formal Methods and Models for Codesign. pp. 43–50 (2011)
31. Lomuscio, A., Strulo, B., Walker, N., Wu, P.: Assume-guarantee reasoning with local specifications. International Conference on Formal Engineering Methods pp. 204–219 (2010)
32. Lutz, A.D.: LTL translation improvements in Spot 1.0. International Journal of Critical Computer-Based Systems 5(1/2), 31 (2014)
33. Maoz, S., Ringert, J.O.: GR(1) synthesis for LTL specification patterns. In: Joint Meeting on Foundations of Software Engineering - ESEC/FSE 2015. pp. 96–106. ACM Press (2015)
34. Merzenich, W., Staiger, L.: Fractals, dimension, and formal languages. Informatique théorique et applications 28(3-4), 361–386 (1994)
35. Nam, W., Alur, R.: Learning-based symbolic assume-guarantee reasoning with automatic decomposition. Automated Technology for Verification and Analysis pp. 170–185 (2006)
36. Pnueli, A., Rosner, R.: On the synthesis of a reactive module. In: Principles of Programming Languages. pp. 179–190 (1989)
37. Pnueli, A.: The temporal logic of programs. In: Annual Symposium on Foundations of Computer Science. pp. 46–57 (1977)
38. Renault, E., Duret-Lutz, A., Kordon, F., Poitrenaud, D.: Three SCC-based emptiness checks for generalized Büchi automata. In: International Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR). pp. 668–682 (2013)
39. Seshia, S.A.: Combining Induction, Deduction, and Structure for Verification and Synthesis. IEEE 103(11), 2036–2051 (2015)
40. Staiger, L.: The Hausdorff Measure of Regular ω-languages is Computable. Tech. Rep. August, Martin-Luther-Universität (1998)
41. Staiger, L.: On the Hausdorff measure of regular omega-languages in Cantor space. Tech. Rep. 1, Martin-Luther-Universität Halle-Wittenberg (2015)
42. Tan, L., Sokolsky, O., Lee, I.: Specification-based testing with linear temporal logic. In: Proceedings of the IEEE International Conference on Information Reuse and Integration. pp. 493–498 (2004)
43. Vardi, M.Y.: An automata-theoretic approach to linear temporal logic. Logics for concurrency pp. 238 – 266 (1996)