A weakness measure for GR(1) formulae

Davide G. Cavezza¹ , Dalal Alrajeh¹, and András György²

¹Imperial College London, London, United Kingdom

²DeepMind, London, United Kingdom

Abstract. When dealing with unrealizable specifications in reactive synthesis, finding the weakest environment assumptions that ensure realizability is often considered a desirable property. However, little effort has been dedicated to defining or evaluating the notion of weakness of assumptions formally. The question of whether one assumption is weaker than another is commonly interpreted by considering the implication relationship between the two or, equivalently, their language inclusion. This interpretation fails to provide any insight into the weakness of the assumptions when implication (or language inclusion) does not hold. To our knowledge, the only measure that is capable of comparing two formulae in this case is entropy, but even it cannot distinguish the weakness of assumptions expressed as fairness properties. In this paper, we propose a refined measure of weakness based on combining entropy with Hausdorff dimension, a concept that captures the notion of size of the ω -language satisfying a linear temporal logic formula. We focus on a special subset of linear temporal logic formulae which is of particular interest in reactive synthesis, called GR(1). We identify the conditions under which this measure is guaranteed to distinguish between weaker and stronger GR(1) formulae, and propose a refined measure to cover cases when two formulae are strictly ordered by implication but have the same entropy and Hausdorff dimension. We prove the consistency between our weakness measure and logical implication, that is, if one formula implies another, the latter is weaker than the former according to our measure. We evaluate our proposed weakness measure in two contexts. The first is in computing GR(1) assumption refinements where our weakness measure is used as a heuristic to drive the refinement search towards weaker solutions. The second is in the context of quantitative model checking where it is used to measure the size of the language of a model violating a linear temporal logic formula.

Keywords: Reactive synthesis, Assumptions refinement, Weakness, Generalized reactivity, Entropy, Hausdorff dimension

1. Introduction

Formal specifications provide significant aid in the analysis of software, supporting tasks such as their verification and implementation. However writing these specifications is difficult and error-prone, often resulting in their incompleteness, inconsistency and/or unrealizability [KHB09]. Hence providing formal and rigorous support for ensuring their highest quality is of key importance [Kup12]. One crucial quality metric for specifications, which this paper focuses on, is that of weakness in the context of reactive synthesis [ADG16, AMT15, CA17, DBSU15].

Reactive synthesis is concerned with finding a system implementation that satisfies a given specification under all possible environments [PR89]. When no such implementation exists, a specification is said to be unrealizable

[CRST08]. Though there may be many reasons for why a specification is unrealizable, common causes include the incompleteness of the given assumptions on the environment [CHJ08] or the restrictiveness of the set of required guarantees the system is expected to achieve simultaneously [BCG⁺14, DBK⁺14].

Recent techniques, such as [AMT13, AMT15, CA17, LDS11], deal with the incompleteness of assumptions by providing algorithms for computing missing assumptions that are necessary to ensure the realizability of a given specification. Their aim is to find the “weakest” assumptions refinements amongst possible alternatives. Other approaches, such as [DBK⁺14, BCG⁺14], focus instead on gradually removing (thus weakening) guarantees until the realizability of the remaining specification is achieved. (The system in this case is said to undergo a process of degradation.) Approaches implementing either tactic, i.e., assumptions refinement or guarantees weakening, consider specifications expressed in a subset of linear temporal logic (LTL), namely generalized reactivity of rank 1 (GR(1) [PPS06]), for which tractable synthesis methods exist. We too focus here on GR(1) specifications.

GR(1) specifications are expressed as a pair consisting of a conjunction of assumptions and of a conjunction of guarantees. We refer to both conjunctions as *GR(1) conjunctions* for short. The *weakness* of a GR(1) conjunction is a property intended to capture the degree of freedom (or permissiveness) a system satisfying the specification has over its behaviours; for instance, in the context of assumptions refinement, weaker assumptions are preferred since they allow for more general solutions to the synthesis problem [CHJ08, Ses15]. Analogously, the *strength* of the guarantees is a dual notion for capturing the restricted set of behaviour that must be permitted by a system—stronger guarantee relaxations are usually preferred in the context of requirements degradation [DBK⁺14].

Existing approaches formalize the weakness relation between specifications through logical implication [AMT13, Ses15], 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 concept of permissiveness [CL08]. 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 (thus we say they are “incomparable” wrt implication).

To overcome this limitation, we propose a quantitative measure for the weakness of GR(1) conjunction—based on their interpretation as an ω -language—and implement a procedure to compute it. The measure builds upon the notion of *entropy* and *Hausdorff dimension* [Sta15], two quantities providing an indication of the size of an ω -language. The basic idea is that the higher the entropy and/or the dimension, the wider the collection of distinct ω -words contained in the ω -language is and thus the more permissive. In our approach, both quantities are computed on automata summarizing all the behaviours satisfying the GR(1) conjunction: while entropy quantifies the entire set of behaviours that can be constructed in the automata, Hausdorff dimension only accounts for the accepting part of these automata, hence providing a more refined way to distinguish between weaker and stronger formulae.

Our contributions are summarized as follows:

- We present an axiomatization of the concept of weakness that extends implication and allows comparison of formulae not implying each other.
- We provide a weakness measure (based on the entropy and the Hausdorff dimension of a formula) that induces a partial order on GR(1) conjunctions consistent with implication. By consistency we mean that if a formula ϕ_1 implies ϕ_2 , then ϕ_1 is weaker than ϕ_2 according to our proposed measure. We identify a class of GR(1) conjunctions, namely *closed*, *strongly connected* ones (which in GR(1) terminology correspond to *strongly connected invariants*), such that our weakness measure is consistent with all axioms.
- For the remaining GR(1) conjunctions, we propose an enhanced weakness measure to allow the strict ordering of formulae that are strictly ordered by implication. The modified measure is based on decomposing the formula into its invariant and its fairness conditions, and measuring the set of behaviors that satisfy the invariant but violate the fairness conditions. In particular we prove the consistency of this enhanced weakness measure with implication for increasingly complex classes of GR(1) conjunctions.
- We demonstrate the applicability of our weakness measure to the problem of GR(1) assumptions refinement [AMT15, CA17] as a way to find weaker refinements. The problem of assumptions refinement usually requires the ranking of several GR(1) formulae. While using implication in this context would require the number of automata to be computed to increase quadratically in the number of formulae, which quickly becomes infeasible, we show that comparing them via a weakness measure only requires a linear number of automata computations.

- We finally show an application of our refined weakness measure to the problem of measuring the language of error as presented in [ABD14a], and show, on the same case study as used in [ABD14a], that our measure is a strict refinement of the measure presented in that work, which is purely based on entropy.

The rest of the paper is organized as follows. Related work is presented in Sect. 2. Notation and background concepts are presented in Sect. 3. Section 4 defines the requirements of a suitable weakness measure in the form of three axioms of weakness. In Sect. 5, we define entropy and Hausdorff dimension, along with algorithms to compute them given a Muller, Büchi, or generalized Büchi automaton; the algorithms will be useful to develop the theory in later sections. In Sect. 6, we introduce the proposed weakness measure first defining it for simpler cases, then for progressively more complex classes of GR(1) conjunctions until encompassing the general case. In the same section, we prove sufficient conditions guaranteeing the consistency between the weakness measure and implication. In Sect. 7 we provide an algorithm to compute the third component of the weakness measure. Section 8 presents several applications of our weakness measure to existing GR(1) benchmarks; an application of the weakness measure as a heuristic for GR(1) assumptions refinement; and an application to quantitative model checking. The experiments data and source code are also available online in [DAV]. Finally, conclusions are drawn in Sect. 9.

2. Related work

Quantitative measures for LTL formulae. The closest notion to our weakness measure is the *entropy* of ω -languages applied by Asarin et al. [ABD14a, ABD⁺14b] in the context quantitative model checking. This quantity measures how diverse the ω -words contained in the language of an LTL formula are. However, it is not sufficient to distinguish between weaker and stronger fairness conditions [ABD14a]. Our measure based on Hausdorff dimension is capable of making this distinction.

The quality of LTL formulae has also been defined in the context of model verification. Henzinger et al. [Hen10, HO13] define 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 this satisfiability region, rather than its distance from a specific model.

The more recent work by Degiovanni et al. [DCA⁺18] uses model counting to measure how many behaviours of a system, described by an LTL formula, lead to a conflict between requirements. Since these behaviours are infinite, they propose to fix a boundary k on the time steps and count all violating finite traces up to the boundary; this provides an estimate of the “likelihood” of reaching a violating state. Our approach takes this notion to the limit for k that grows to infinity, thus replacing the estimate with an exact value.

Quantitative extensions of temporal logics. An alternative way to measure the size of behaviour sets is via probabilities. Probabilistic model checking [HJ94, Kwi07] enhances the syntax and semantics of temporal logics (usually CTL, *computation tree logic*) with probabilities. This allows for expressing 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 other than probabilities have been proposed in [AAK13, BCHJ09, CDF⁺06, CHJ08]. The difference between using probabilities/preference metrics and our proposal is that these measures are parameters provided in input in addition to the logical formulae, or are applicable to stochastic settings where models of behaviour are not purely logical (for instance, Markov chains); these parameters, although conceptually related, may not reflect the true weakness of a logical formula. The measure we propose instead quantifies a concept of weakness *intrinsic* to the LTL formula itself.

Weakness and automata. The problem of identifying weakest assumptions appears in the context of assume-guarantee reasoning [CGP03, LSWW10, NA06] for compositional model checking. In order to perform model checking of large systems, their models are generally broken down to components that can be checked independently for correctness: when checking each component, assume-guarantee reasoning supposes that all other

components satisfy their respective specifications and uses this assumption to prove the correctness of the component undergoing the check. In this setting, 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 in assume-guarantee reasoning are formalized as transition systems (e.g., modal transition systems) rather than declarative LTL specifications, which is where the focus of our work lies.

GR(1) synthesis and assumption refinement. In the context of reactive synthesis from GR(1) formulae, several approaches have been proposed to identify the cause of a violation in the form of an *unrealizable core* [CRST08], and/or in the form of a *counterstrategy* described by an automaton summarizing a set of adversarial environments [KHB09, KMR17].

Existing techniques, such as [LDS11, AMT13, AMT15, CA17, MRS19], use this information to identify a set of assumptions that rule out the adversarial environments and ensure realizability. This is known as *GR(1) assumption refinement*, and is typically formulated as a search over the assumptions space driven by counterstrategies. An important quality measure of assumption refinements is weakness [Ses15]: the generated assumptions are required to be as weak as possible so that the environment is not over-constrained. Weakness in this literature is generally defined via implication and entailment. The weakness measure we propose in this paper can be used as both a quality metric for the computed solutions and as a heuristic to drive the search towards weaker ones.

Assumption refinement is a means to achieve *hard* realizability of a GR(1) specification starting from an unrealizable one. Other techniques [TUS⁺17, DGT18, TN16] are aimed at achieving some notion of *soft* realizability, where some of the guarantees are mandatory and others are “desirable”. The goal of achieving the desirable guarantees is formalized quantitatively as a mean-payoff game or a MAX-SAT problem. As for quantitative model checking techniques, these approaches are based on additional parameters provided in input to the problem, which are orthogonal to the notion of the weakness of the LTL formula.

3. Preliminaries

Languages and automata. Let Σ be a finite set of symbols, which we call the *alphabet*. A *word* over Σ is a finite sequence of symbols in Σ . An ω -word is an infinite sequence of these symbols. A set of words is called a *language*, while a set of ω -words is called an ω -language. Despite this difference between the terms *word* and ω -word, we will sometimes drop the prefix ω from the latter when it is clear from the context that the word is infinite-length.

A word w is explicitly denoted as a sequence of its symbols $\alpha_1\alpha_2\ldots\alpha_n$, or with a parenthesis notation $(\alpha_1, \alpha_2, \ldots, \alpha_n)$, with the symbols separated by commas; the same notation is used for ω -words. The notation $w[i..]$ denotes the suffix of w starting with the symbol α_i . We write $w[i]$ to denote the symbol appearing in the i th position in w , i.e., α_i . $|w|$ denotes the number of symbols appearing in the word w (aka the length of w).

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, where the ω -word comes first, 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 . We also borrow the convention from the literature on languages (such as in [CJS09]) of dropping brackets from singleton sets of words: hence, the set $\{\sigma\}$ containing only the word σ is written as σ , whilst the set $\{\sigma\} \cdot \Sigma^*$ of all finite words starting with a σ is written as $\sigma \Sigma^*$.

Given an ω -language $L \subseteq \Sigma^\omega$, we denote by $\text{pref}_n(L)$ the set of all $w \in \Sigma^*$ of length n such that w is a prefix of some word in L . We also define $\text{pref}(L) = \bigcup_{n \in \mathbb{N}} \text{pref}_n(L)$ the set of all the prefixes of ω -words in L . An ω -language L is *closed* if and only if, for any ω -word w such that $\text{pref}(\{w\}) \subseteq \text{pref}(L)$, $w \in L$. In other words, L is closed if whenever a word w is arbitrarily close to some word w' in L (that is, for every finite prefix of w there is a word w'

in L having the same prefix), 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 . These two concepts come from the topology over Σ^ω defined in [Sta15].

Despite being infinite sets, the ω -languages of interest to us allow a computable 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, q_0 is the initial state, $\delta : Q \times \Sigma \rightarrow Q$ 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 $\forall i \in \mathbb{N} : q_i = \delta(q_{i-1}, w[i])$. To simplify notation, for any finite length word $w^n = w[1], \dots, w[n] \in \Sigma^n$, we sometimes use the shorthand $\delta(q_0, w^n) = q_n$ to denote the final state induced by w^n , which in turn is the n th state q_n induced by w . Let $\text{Inf}(w) \subseteq Q$ be the set of states occurring infinitely many times in $\mathcal{M}(w)$. Then an ω -word is said to be *accepted* by $\mathcal{M}$ if and only if $\text{Inf}(w) \in T$. By extension, the ω -language accepted by $\mathcal{M}$ is the set of ω -words accepted by $\mathcal{M}$.

While DMAs are sufficient to define regular languages, many applications use different accepting conditions, e.g., Büchi automata [SB00, BKvS12, GM03]. 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 $\mathcal{B}$ if and only if $\text{Inf}(w) \cap F \neq \emptyset$. Given a DBA it is always possible to obtain an equivalent DMA by replacing the Büchi acceptance condition. However, there exist DMAs that are not equivalent to any deterministic Büchi automaton [GTW02]. In this case, nondeterministic automata may be used to extend the expressiveness of Büchi automata. We do not consider nondeterministic automata in our work, as the computation of the quantities defined below requires deterministic automata.

Without loss of generality, we assume that every state of any automaton is reachable for some appropriately chosen ω -word.

Graphs and adjacency matrices. Muller and Büchi automata are typically represented by graphs. In particular, a Büchi automaton $\mathcal{B}$ (or a Muller automaton $\mathcal{M}$) is represented as a labelled graph $\mathcal{G} = (Q, E)$, where the set of *nodes* Q is the set of states in $\mathcal{B}$, $E \subseteq Q \times \mathbb{N} \times Q$ is a set of (multi-)edges such that $(q_i, n, q_j) \in E$ if and only if there exist exactly n symbols $\sigma \in \Sigma$ such that $\delta(q_i, \sigma) = q_j$. Throughout the paper, we heavily rely on the properties of the graph underlying an automaton. In particular, we use the adjacency matrix, the notion of a subgraph induced by a subset of states, and strong connectedness.

Given a subset $Q' \subseteq Q$, the *subgraph* induced by Q' on $\mathcal{G}$ is the graph $\mathcal{G}[Q'] = (Q', E')$ such that $(q_i, n, q_j) \in E'$ if and only if $(q_i, n, q_j) \in E$ and $q_i, q_j \in Q'$.

A graph is *strongly connected* if, for any two states $q_i, q_j \in Q$, there exists a path (a sequence of consecutive edges) from q_i to q_j and vice versa. If a graph $\mathcal{G}$ is not strongly connected, it can admit one or more *strongly connected components* (SCCs), which are maximal strongly connected subgraphs of $\mathcal{G}$. Naturally, different SCCs are disjoint.

Let A_{ij} denote an element in position (i, j) of matrix A . A graph can be represented by its *extended adjacency matrix* A , defined as the square matrix of size $|Q| \times |Q|$ such that: $A_{ij} = n$ if and only if $(q_i, n, q_j) \in E$.

Since the elements of A are nonnegative, A is a *nonnegative matrix*. Given a subgraph $\mathcal{G}[Q']$ induced by $Q' \subseteq Q$, we denote its extended adjacency matrix by $A[Q']$; this $|Q'| \times |Q'|$ matrix can be obtained by removing from A the rows and columns corresponding to the states not in Q' . Any matrix obtained by removing from A a set of rows and columns with the same indices is called a *principal submatrix* of A [HJ12]. We say that an adjacency matrix is *irreducible* if it describes a strongly connected graph [HJ12].

Given an automaton's graph, we are interested in the growth rate of the number of paths of a certain length as a function of the length. For large enough lengths, this number grows approximately exponentially, and the growth rate (i.e., the base of this exponential function) is the *spectral radius* of the adjacency matrix A [MS94], which in turn is the Perron-Frobenius eigenvalue of A (a positive real eigenvalue which has the largest absolute value among all eigenvalues), and is denoted by $\rho(A)$ [Sen06] (more details are given in Sect. 5). In what follows, we use a property of the spectral radius that states a sufficient condition for a strict ordering of the spectral radii of two matrices.

Theorem 3.1 (Monotonicity of spectral radius [BP94, Chapter 2 Corollary 1.5]) Let A and B be two nonnegative square matrices of the same size such that $A_{ij} \leq B_{ij}$ for all (i, j) that index positions in A and B , and $A_{ij} < B_{ij}$ for some (i, j) . Suppose $A + B$ is irreducible. Then $\rho(A) < \rho(B)$.

Linear temporal logic and GR(1). *Linear temporal logic* (LTL) [Pnu77] 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. The syntax of LTL is defined by the following grammar:

$$\phi ::= \text{true} \mid \text{false} \mid p \mid \neg \phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \phi \rightarrow \phi \mid \phi \leftrightarrow \phi \mid \mathbf{X}\phi \mid \mathbf{F}\phi \mid \mathbf{G}\phi \mid \phi \mathbf{U}\phi$$

where $p \in \mathcal{V}$.

The following statements describe LTL semantics, that is when an ω -word $w \in (2^{\mathcal{V}})^{\omega}$ is said to satisfy an LTL formula. Hereafter, ϕ and ψ are LTL formulae.

$w \models \text{true}$	always
$w \models \text{false}$	never
$w \models p$	iff $p \in w[1]$
$w \models \neg \phi$	iff $w \not\models \phi$
$w \models \phi \wedge \psi$	iff $w \models \phi$ and $w \models \psi$
$w \models \phi \vee \psi$	iff $w \models \phi$ or $w \models \psi$
$w \models \phi \rightarrow \psi$	iff $w \models \neg \phi \vee \psi$
$w \models \phi \leftrightarrow \psi$	iff $w \models (\phi \wedge \psi) \vee (\neg \phi \wedge \neg \psi)$
$w \models \mathbf{X}\phi$	iff $w^2 \models \phi$
$w \models \mathbf{F}\phi$	iff $\exists j \in \mathbb{N}$ s. t. $w^j \models \phi$
$w \models \mathbf{G}\phi$	iff $\forall j \in \mathbb{N} \ w^j \models \phi$
$w \models \phi \mathbf{U}\psi$	iff $\exists j \in \mathbb{N}$ s. t. $w^j \models \psi$ and $\forall i < j \ w^i \models \phi$

In other words, **F** can be read as “eventually”, **G** as “always”, **X** as “next” and **U** as “until”. We say that an LTL formula ϕ is *valid* if $w \models \phi$ for every $w \in (2^{\mathcal{V}})^{\omega}$.

LTL formulae are evaluated over ω -words over the alphabet $\Sigma = 2^{\mathcal{V}}$: this alphabet corresponds to all the assignments of truth values to the variables in $\mathcal{V}$, and consequently the ω -words are infinite sequences of assignments to those variables. The set of ω -words that satisfy a formula ϕ is a regular ω -language [Var96] denoted by $L(\phi)$.

In this paper, we deal with a specific subset of LTL, called *Generalized Reactivity (1)* (GR(1)), which is largely used in controller synthesis [KHB09, DBPU10, BCG⁺10, BJP⁺12, BDP⁺13, MR15]. This subset makes use of the operators **G**, which states that its operand formula must hold at each step of a valuation sequence, **F**, which requires its operand formula to hold at some point in the sequence, and **X**, 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 $\phi = \phi^{\mathcal{E}} \rightarrow \phi^{\mathcal{S}}$, 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 $B^{\text{init}}(\mathcal{V})$; (ii) one or more *invariants*, conditions of the form $\mathbf{G}B^{\text{inv}}(\mathcal{V} \cup \mathbf{X}\mathcal{V})$, where $B^{\text{inv}}(\mathcal{V} \cup \mathbf{X}\mathcal{V})$ denotes a pure Boolean expression over the set of variables in $\mathcal{V}$ and the set of atoms obtained by prepending an **X** operator to each variable; and (iii) one or more *fairness conditions* of the form $\mathbf{GFB}^{\text{fair}}(\mathcal{V})$. We will refer to the formulae $\phi^{\mathcal{E}}$ and $\phi^{\mathcal{S}}$ as *GR(1) conjunctions*.

4. Weakness of LTL formulae

In this section, we present an axiomatization of the weakness of an LTL formula. Hereafter, we denote the weakness measure of an LTL formula as a function $d : \Phi \rightarrow P$, where Φ is the set of LTL formulae and P is a totally ordered set (such as the set of n -tuples $\mathbb{R}^n$ with lexicographic order). The weakness measure of a formula $\phi \in \Phi$, denoted by $d(\phi)$, will be interpreted as follows: the higher this measure is, the weaker ϕ is, that is, ϕ_2 is weaker than ϕ_1 if $d(\phi_1) \leq d(\phi_2)$.

In settings such as [ADG16, AMT13, Ses15], 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 4.1 (Consistency with implication) Given two LTL formulae ϕ_1 and ϕ_2 , if $\phi_1 \rightarrow \phi_2$, then $d(\phi_1) \leq d(\phi_2)$.

In addition to this, it is desirable that a weakness measure performs at least as well as implication in discriminating pairs of formulae where one strictly implies the other.

Axiom 4.2 (Consistency with strict implication) Given two LTL formulae ϕ_1 and ϕ_2 such that $\phi_1 \rightarrow \phi_2$ is valid and $\phi_2 \rightarrow \phi_1$ is not. Then $d(\phi_1) < d(\phi_2)$.

Notice that this criterion defines a partial ordering of specifications: if none of the two formulae implies the other, then they are *incomparable wrt implication* according to this criterion.

The above two axioms are for formulae that are comparable via implication. However, even for the incomparable case, it may be useful to define some ordering between formulae based on their “permissiveness”.

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 wrt implication, it is clear that ϕ_1 allows in some sense fewer behaviors than ϕ_2 (and hence more permissive): at each time step, the former allows for 2 distinct valuations of $\mathcal{V}$ while ϕ_2 allows 4 of them.

Furthermore, 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).

The number of prefixes admitted by an LTL formula up to a time step k is already used as a measure of the permissiveness of a formula (for instance in model counting [DCA⁺18]). This notion provides an intuitive criterion to impose a linear order when implication does not hold, captured by the following axiom:

Axiom 4.3 (Consistency with permissiveness) Given two LTL formulae ϕ_1 and ϕ_2 , $d(\phi_1) \leq d(\phi_2)$ if there exists some length $\bar{n}$ such that, for every $n > \bar{n}$, the set of prefixes of length n in $L(\phi_1)$ contains fewer elements than the set of prefixes of the same length in $L(\phi_2)$; i.e., if $\exists \bar{n}, \forall n > \bar{n}, |pref_n(L(\phi_1))| \leq |pref_n(L(\phi_2))|$, then $d(\phi_1) \leq d(\phi_2)$.

In this paper, we focus on defining a weakness measure for GR(1) conjunctions satisfying Axiom 4.1. We also identify a class of formulae in which the weakness measure satisfies 4.3. We then show that, although our weakness measure is not guaranteed to satisfy Axioms 4.2 in general, our proposed extensions to the measure aim at augmenting its resolution, increasing the set of formulae that can be strictly ordered via the measure when strict implication holds; we also identify two classes of formulae in which Axiom 4.2 is provably satisfied.

5. Entropy and Hausdorff dimension

Our proposal for a weakness measure is based on two quantities, *entropy* and *Hausdorff dimension*, defined in the context of ω -automata. In Sect. 5.1, we provide their definitions and the algorithm devised in [Sta98] to compute Hausdorff dimension using Muller automata. Then we extend this algorithm in Sect. 5.2 to Büchi and generalized Büchi automata.

5.1. Definitions

Hausdorff dimension and *Hausdorff measure* are basic concepts in fractal geometry and represent a way of defining measures—that is, analogous concepts to length, area, volume from classical geometry—for fractals [Fal04]. Staiger [Sta15] pointed out a homeomorphism between fractals and regular ω -languages and proposed an analogous interpretation of the Hausdorff dimension and measure for 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 as a function of n . 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 taken from [Sta98].

Definition 5.1 (γ -dimensional Hausdorff outer measure) Given a regular ω -language L over an alphabet Σ with cardinality r , and a nonnegative real value γ , the γ -dimensional Hausdorff outer measure of L is defined as

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

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 such that every word in L has at least one prefix in V .

Definition 5.2 (Hausdorff dimension and measure) Given an ω -language L , its Hausdorff dimension, denoted by $\dim(L)$, is the (unique) value $\bar{\gamma}$ such that

$$\begin{aligned} m_\gamma(L) &= \infty & \gamma < \bar{\gamma} \\ m_\gamma(L) &= 0 & \gamma > \bar{\gamma} \end{aligned}$$

The value $m_{\dim(L)}(L)$ is called the Hausdorff measure of L , and is denoted by $m(L)$.

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 weighting each prefix w with a quantity $r^{-\gamma|w|}$ that decreases exponentially as the prefix length increases. This limit can be finite and positive for at most one value of the γ parameter. This value is called Hausdorff dimension.

We illustrate the notions above in the following examples.

Example 5.3 Consider the set of variables $\mathcal{V} = \{a, b\}$, and the associated alphabet $\Sigma = \{\emptyset, \{a\}, \{b\}, \{a, b\}\}$, with number of symbols $r = 4$; we denote the symbols in Σ as $\alpha_0, \alpha_1, \alpha_2$, and α_3 respectively. We want to determine the γ -dimensional Hausdorff measure of the language L of the formula $\mathbf{Ga}$, i.e., the set of all ω -words containing only symbols α_1 and α_3 :

$$L = \{\alpha_1, \alpha_3\}^\omega.$$

To achieve this, we determine a $V_n \in \mathcal{L}_n$ which achieves the limit of the infimum in (1), in particular, for $\gamma < \dim(L)$, minimizes the sum $s_V = \sum_{v \in V} 4^{-\gamma|v|}$ over $\mathcal{L}_n$ (that is over the set of languages V containing only words of length at least n such that every ω -word in L has at least one prefix in V).

Next we show that $V_n = \{\alpha_1, \alpha_3\}^n$, which trivially belongs to $\mathcal{L}_n$, is an appropriate choice. Since V_n contains 2^n words of length n ,

$$s_{V_n}(n) = \sum_{v \in V_n} 4^{-\gamma n} = 2^n 4^{-\gamma n} = 2^{(1-2\gamma)n} = \begin{cases} 2^{(1-2\gamma)n} \xrightarrow{n \rightarrow \infty} 0 & \text{for } \gamma > 1/2; \\ 1 & \text{for } \gamma = 1/2; \\ 2^{(1-2\gamma)n} \xrightarrow{n \rightarrow \infty} \infty & \text{for } \gamma < 1/2. \end{cases}$$

Thus, for $\gamma > 1/2$,

$$0 \leq m_\gamma(L) = \lim_{n \rightarrow \infty} \inf_{V \in \mathcal{L}_n} s_V \leq \lim_{n \rightarrow \infty} s_{V_n} = 0.$$

Hence $m_\gamma(L) = 0$ for $\gamma > 1/2$.

For $\gamma \leq 1/2$ we show that $s_{V_n} = \inf_{V \in \mathcal{L}_n} s_V$, that is V_n achieves the infimum over $\mathcal{L}_n$. Intuitively, what happens is that if we take any $V \in \mathcal{L}_n$ and replace a word $v \in V$ with $v\alpha_1$ and $v\alpha_3$, then the corresponding sum s_V increases, since the difference is $2 \cdot 4^{-\gamma(|v|+1)} - 4^{-\gamma|v|} = 4^{-\gamma|v|}(2 \cdot 4^{-\gamma} - 1) \geq 0$. Reversing the argument, starting from any finite V , we can get to V_n by reducing (non-increasing) s_V in every step. Unfortunately this argument does not work when V is infinite. To address this case rigorously, we take a different route.¹ Consider first the case of $\gamma = 1/2$, and let $V \in \mathcal{L}_1$ (note that $\mathcal{L}_{i+1} \subset \mathcal{L}_i$ for any i , hence $V \in \mathcal{L}_n$ for all n). Introducing the notation $\alpha_1 = 0$ and $\alpha_3 = 1$, for any $v \in V$ of length $k = |v|$, let r_v denote the real number with binary expansion $0.v_1 v_2 \dots v_k$. Then the interval $I_v = [v, v + 1/2^k]$ is the set of real numbers whose binary expansion begins with $0.v_1 v_2 \dots v_k$ (where $v + 1/2^k$ is interpreted here as $0.v_1 v_2 \dots v_k 111 \dots$). Since L contains all infinite

¹ The argument presented is based on the ideas used in the proof of the extended Kraft inequality in information theory [CT06].

binary strings, and each of them have a prefix in V , $\bigcup_{v \in V} I_v$ contains the whole unit interval $[0, 1]$. Therefore, denoting by $|I|$ the length of interval I ,

$$s_V = \sum_{v \in V} 4^{-\frac{1}{2}|v|} = \sum_{v \in V} 2^{-|v|} = \sum_{v \in V} |I_v| \geq \left| \bigcup_{v \in V} I_v \right| \geq |[0, 1]| = 1,$$

which readily shows that V_n is minimal for $\gamma = 1/2$ and $m_{1/2}(L) = 1$. Using the above inequality, for $\gamma < 1/2$ and $V \in \mathcal{L}_n$,

$$s_V = \sum_{v \in V} 4^{-\gamma|v|} = \sum_{v \in V} 2^{-|v|} 4^{(1/2-\gamma)|v|} \geq 4^{(1/2-\gamma)n} \sum_{v \in V} 2^{-|v|} \geq 4^{(1/2-\gamma)n}$$

showing the minimality of V_n again. Therefore, $m_\gamma(L) = \infty$ for $\gamma < 1/2$. In summary, we have

$$\dim(L) = 1/2 \quad \text{and} \quad m(L) = 1.$$

The Hausdorff dimension properly captures the notion that when generating a word from L , every time half of the possibilities (relative to Σ) are excluded.

Example 5.4 Consider the set of variables $\mathcal{V} = \{a\}$ and the associated alphabet $\Sigma = \{\emptyset, \{a\}\}$, with number of symbols $r = 2$. We denote the symbol $\{a\}$ by α for short. Let the language L be the language of the formula **FGa**, which is the set of all ω -words ending in an infinite sequence of α s,

$$L = \Sigma^* \alpha^\omega.$$

We want to determine the γ -dimensional outer measure of L , and—from it—its Hausdorff dimension and measure.

Definition 5.1 requires the identification of a set $V \in \mathcal{L}_n$, for a fixed n , such that $s(n) = \sum_{v \in V} r^{-\gamma|v|}$ is as small as possible. This sum s is obtained by attaching a measure $r^{-\gamma|v|}$ to each word $v \in V$. This quantity is non-negative and decreasing in $|v|$, thereby the sum can be minimized by choosing a V containing a few short words and several long words.

Consider a set

$$V_1 = \alpha^n \cup \emptyset \alpha^n \cup \Sigma \emptyset \alpha^{n+1} \cup \Sigma^2 \emptyset \alpha^{n+2} \cup \dots = \alpha^n \cup \bigcup_{i=0}^{\infty} \Sigma^i \emptyset \alpha^{n+i}.$$

This is constructed such that any word having i symbols before the last occurrence of $\emptyset$ has also i α s at the end. Notice that any ω -word of L has at least one prefix in V_1 , and all elements of V_1 have length at least n , hence $V_1 \in \mathcal{L}_n$. The set V_1 contains one word of length n and 2^i words of length $n + 2i + 1$; hence the sum is

$$s_1(n) = 2^{-\gamma n} + \sum_{i=0}^{\infty} 2^i 2^{-\gamma(n+2i+1)} = \begin{cases} 2^{-\gamma n} \left(1 + \frac{2^{-\gamma}}{1-2^{1-2\gamma}}\right) & \text{for } \gamma > 1/2 \\ \infty & \text{for } \gamma \leq 1/2 \end{cases}.$$

For any fixed γ , we can find another set which has a lower sum. Consider the set

$$V_2 = \alpha^n \cup \emptyset \alpha^n \cup \Sigma \emptyset \alpha^{n+2} \cup \Sigma^2 \emptyset \alpha^{n+4} \cup \dots = \alpha^n \cup \bigcup_{i=0}^{\infty} \Sigma^i \emptyset \alpha^{n+2i}.$$

That is, a word $v \in V_2$ ends in twice as many α s as symbols it has before its last $\emptyset$. For the same reasons as V_1 , also V_2 belongs to $\mathcal{L}_n$. V_2 contains one word of length n and 2^i words of length $n + 3i + 1$. Its corresponding sum is

$$s_2(n) = 2^{-\gamma n} + \sum_{i=0}^{\infty} 2^i 2^{-\gamma(n+3i+1)} = \begin{cases} 2^{-\gamma n} \left(1 + \frac{2^{-\gamma}}{1-2^{1-3\gamma}}\right) & \text{for } \gamma > 1/3 \\ \infty & \text{for } \gamma \leq 1/3 \end{cases}.$$

A sequence of sets $V_j \in \mathcal{L}_n$ can be constructed by following the same rule:

$$V_j = \alpha^n \cup \bigcup_{i=0}^{\infty} \Sigma^i \emptyset \alpha^{n+j \cdot i}$$

with sums

$$s_j(n) = 2^{-\gamma n} + \sum_{i=0}^{\infty} 2^i 2^{-\gamma(n+(j+1) \cdot i+1)} = \begin{cases} 2^{-\gamma n} \left(1 + \frac{2^{-\gamma}}{1-2^{1-(j+1)\gamma}}\right) & \text{for } \gamma > 1/(j+1) \\ \infty & \text{for } \gamma \leq 1/(j+1) \end{cases}.$$

Of course $\inf_{V \in \mathcal{L}_n} s(n) \leq s_j(n)$ for every $j \in \mathbb{N}$. Taking the limit over n on both sides, we obtain that regardless of the parameter $\gamma > 0$

$$m_\gamma(L) = \lim_{n \rightarrow \infty} \inf_{V \in \mathcal{L}_n} s(n) \leq \lim_{n \rightarrow \infty} s_j(n) = 0 \text{ for every } j \text{ and } \gamma > 0.$$

Since the γ -dimensional Hausdorff measure is nonnegative, equality holds in the expression above. From this we conclude the Hausdorff dimension of L is

$$\dim(L) = 0$$

and its Hausdorff measure is

$$m_{\dim(L)}(L) = \infty.$$

This example shows how the Hausdorff dimension is mainly affected by the infinite suffixes allowed in L . If all $w \in L$ have a restriction on the allowed suffixes, it is possible to reduce $s(n)$ by extending the words in $V \in \mathcal{L}_n$ with longer suffixes fulfilling these restrictions. $\square$

A related concept appearing in the literature which we will make use of in our weakness measure is entropy:

Definition 5.5 (Entropy [MS94]) Given an ω -language $L \subseteq \Sigma^\omega$ over an alphabet of size r , the *entropy* of L is $H(L) = \limsup_{n \rightarrow \infty} \frac{1}{n} \log_r |pref_n(L)|$, where $pref_n(L)$ is the set of all prefixes of length n of the ω -words in L .

In other words, the entropy quantifies the growth rate of the number of n -long prefixes of the ω -words in L as n grows. Since the number of prefixes grows exponentially in L (its maximum value is r^n , in case L allows any symbol at any position in the ω -word), its logarithm is considered.

Example 5.6 Consider the language L of the formula **FGa** over the variables $\mathcal{V} = \{a\}$ as in Example 5.4, $L = \Sigma^* \{a\}^\omega$. The set of prefixes of length n is

$$pref_n(L) = \Sigma^n.$$

Applying Definition 5.5, the entropy can be calculated as

$$H(L) = \limsup_{n \rightarrow \infty} \frac{1}{n} \log_r |pref_n(L)| = \limsup_{n \rightarrow \infty} \frac{1}{n} \log_2 2^n = 1.$$

Notice that the entropy is greater than the Hausdorff dimension, since it is affected by the number of prefixes of L rather than by its suffixes. In contrast, the entropy for Example 5.3 is $1/2$, which is equal to the Hausdorff dimension.

It has been proved [MS94] that $\dim(L) \leq H(L)$ in general, and $\dim(L) = H(L)$ if L is a closed ω -language.

The entropy of a closed language can be computed on its Büchi automaton $\mathcal{B}$ if all states are accepting [MS94]. Given a Büchi automaton with all states accepting and its interpretation as a graph $\mathcal{G}$, [MS94] presents the following algorithm to compute the entropy:

1. Compute all the strongly connected components $\{SCC_i\}$ and their adjacency matrices $\{A_i\}$;
2. For every A_i compute the Perron-Frobenius eigenvalue $\rho(A_i)$;²
3. Return the maximum $\log_r \rho(A_i)$ (over all SCCs).

² Since A_i is the adjacency matrix of a strongly connected component, it is an irreducible non-negative matrix. Therefore, it has a real eigenvalue $\rho(A_i)$ (of multiplicity 1) which is at least as large as the absolute value of any other eigenvalue of A_i . This eigenvalue is called the Perron-Frobenius eigenvalue (see, e.g., [Sen06, Theorem 1.5]).

To give an intuition of the relationship between the entropy and the Perron-Frobenius eigenvalues of the A_i , we recall that the (s, s') entry of the power matrix A_i^n is the number of paths of length n from state s to state s' . Since A_i is the adjacency matrix of a strongly connected component, it is irreducible, and hence for any $s, s' \in SCC_i$, $\limsup_{n \rightarrow \infty} \frac{1}{n} \log_r A_i^n(s, s') = \log_r \rho(A_i)$ by the Perron-Frobenius theory [Sen06, Section 1.4]. Therefore, for every SCC_i ,

$$\limsup_{n \rightarrow \infty} \frac{1}{n} \log_r |pref_n(L)| = \limsup_{n \rightarrow \infty} \frac{1}{n} \log_r (\rho(A_i))^n = \log_r \rho(A_i). \quad (2)$$

The correctness of the algorithm then follows by noticing that the growth rate of the total number of prefixes over all SCCs is dominated by the largest growth rate over the components.

For the case when L is not closed, a general algorithm is presented in [Sta98, Sta15], which highlights the difference between the entropy and the Hausdorff dimension. The method, shown in Algorithm 1, is based on computing a Muller automaton $\mathcal{M}_L$ accepting L whose table of accepting state sets is T . For each accepting set $S' \in T$ 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 the starting state s [Sta98]. The Hausdorff dimension of L is then

$$\dim(L) = \max_{S' \in T} H(C_{S'}), \quad (3)$$

where the $H(C_{S'})$ can be computed as the Perron-Frobenius eigenvalue of the adjacency matrix of the subgraph induced by S' .

Algorithm 1: HAUSDORFFDMA

Input: A DMA $\mathcal{M} = \langle Q, \Sigma, q_0, \delta, T \rangle$, with adjacency matrix M

Output: Hausdorff dimension dim

```

1 foreach  $S' \in T$  do
2    $\rho_{S'} \leftarrow$  Perron-Frobenius eigenvalue of  $M[S']$ ;
3 end
4  $\rho \leftarrow \max_{S'} \rho_{S'}$ ;
5  $dim \leftarrow \log_r \rho$ ;
6 return  $dim$ 

```

As a final remark, we note that the language $C_{S'}$ plays a central role in the definition of our weakness measure (see Sect. 6).

5.2. Hausdorff dimension of Büchi and generalized Büchi automata

Algorithm HAUSDORFFDMA requires the Muller automaton of the ω -language. However, in applications to formal methods, other types of automata are usually employed, typically Büchi automata or generalized Büchi automata [BKvS12, GL02, KGFP09]. In this section, we extend the original algorithm to these ω -automata.

The algorithm is based on finding the DMA equivalent to a given DBA and on algebraic results on nonnegative matrices. First we assume the DBA is strongly connected; later we show how the algorithm changes when this assumption is relaxed.

Strongly connected Büchi automata. Let $\mathcal{B} = (Q, \Sigma, q_0, \delta, F_B)$ be a strongly connected DBA and $\mathcal{M} = (Q, \Sigma, q_0, \delta, F_M)$ be a DMA where $F_M = \{S'_i \in 2^Q \mid S'_i \cap F_B \neq \emptyset\}$. That is, $\mathcal{M}$ has the same states and transition function as $\mathcal{B}$, and the collection of accepting states F_M consists of all the subsets of Q that have nonempty intersection with the accepting set of F_B . The next proposition shows that the two automata are equivalent:

Proposition 5.7 The DBA $\mathcal{B}$ and the DMA $\mathcal{M}$ defined above accept the same ω -language.

Proof Let w be a word accepted by $\mathcal{B}$. Then $\text{Inf}_{\mathcal{B}}(w) \cap F_B \neq \emptyset$, and therefore $\text{Inf}_{\mathcal{B}}(w) \in F_M$. Moreover, since the two automata have the same set of states and transition function, the sequences of states generated by the

two automata are the same, that is, $\mathcal{B}(w) = \mathcal{M}(w)$; in other words, w induces the same computation on the two automata. Hence $\text{Inf}_{\mathcal{M}}(w) = \text{Inf}_{\mathcal{B}}(w) \in F_{\mathcal{M}}$, and by definition of Muller acceptance also $\mathcal{M}$ accepts w . Showing that $\mathcal{B}$ accepts every word accepted by $\mathcal{M}$ can be done similarly. $\square$

Algorithm HAUSDORFFDMA computes the Hausdorff dimension, operating directly on $\mathcal{M}$. However, the number of accepting sets in $F_{\mathcal{M}}$ grows exponentially with the size of Q , which makes the execution of the loop in Algorithm 1 intractable. The computational complexity of the algorithm can be reduced by excluding the S'_i that do not affect the overall dimension. Those can be identified via the analysis of algebraic properties of nonnegative matrices.

Next we show that the accepting sets S'_i that affect the Hausdorff dimension are the maximal accepting sets, defined as follows:

Definition 5.8 (Maximal accepting sets) We call a set $S_j \in F_{\mathcal{M}}$ a *maximal accepting set* if there is no $S_i \in F_{\mathcal{M}}$ such that $S_j \subset S_i$.

Then the following proposition holds:

Proposition 5.9 Consider a language L represented by a deterministic Muller automaton with adjacency matrix M , and let T_m denote the table of maximal accepting state sets. Then $\dim(L) = \max_{S \in T_m} \log_r \rho(M[S])$, where $\rho(A)$ denotes the spectral radius of a matrix A . Accordingly, in Algorithm 1, if the loop in lines 1-3 is modified to iterate over maximal accepting sets only, the output is still the Hausdorff dimension of $L(\mathcal{M})$.

Proof Given two sets of states S'_i and S'_j such that $S'_i \subseteq S'_j$, the matrices $M[S'_i]$ and $M[S'_j]$ are principal submatrices of M by definition. More specifically, $M[S'_i]$ is a principal submatrix of $M[S'_j]$. By Corollary 8.1.20 in [HJ12], we know that $\rho(M[S'_i]) \leq \rho(M[S'_j])$. This, combined with (3), implies that $\dim(L) = \max_{S \in T_m} \log_r \rho(M[S])$. The statement on Algorithm 1 then follows trivially. $\square$

By construction, Q is an accepting set of $\mathcal{M}$, and since it contains all states, it is the unique maximal accepting set (note that the construction is possible because we assumed that the Büchi automaton we start with is strongly connected). Therefore, the Hausdorff dimension of $L(\mathcal{M})$ is just the spectral radius of the adjacency matrix of $\mathcal{M}$. This in turn is the same adjacency matrix as that of the original Büchi automaton $\mathcal{B}$, resulting in the following theorem:

Theorem 5.10 (Hausdorff dimension of strongly connected Büchi automata) Given a strongly connected Büchi automaton $\mathcal{B}$ with adjacency matrix M , its Hausdorff dimension is $\dim(L(\mathcal{B})) = \log_r \rho(M)$.

Büchi automata. Extending the above simplification in computing the Hausdorff dimension to non-strongly connected Büchi automata is straightforward. Consider a Büchi automaton $\mathcal{B} = (Q, \Sigma, q_0, \delta, F_{\mathcal{B}})$ such that its strongly connected components are $C_1, \dots, C_k \subseteq Q$. Let $\mathcal{M} = (Q, \Sigma, q_0, \delta, F_{\mathcal{M}})$, where $F_{\mathcal{M}} = \{S_i \in Q \mid S_i \subseteq C_j \text{ for some } j \wedge S_i \cap F_{\mathcal{B}} \neq \emptyset\}$. It is easy to verify that $\mathcal{B}$ and $\mathcal{M}$ accept the same language.

In this case, the maximal accepting subsets in $F_{\mathcal{M}}$ are all the SCCs C_j having nonempty intersection with $F_{\mathcal{B}}$; we call these *accepting SCCs*. Therefore, similarly to the previous section, the algorithm only needs to compute the spectral radii of $M[C_j]$ for any such C_j ; the Hausdorff dimension of $L(\mathcal{B})$ is the maximum of these spectral radii. This proves the following theorem:

Theorem 5.11 Consider a Büchi automaton $\mathcal{B}$ with accepting SCCs $C_1, \dots, C_k$ and their adjacency matrices $M[C_1], \dots, M[C_k]$ respectively. Then $\dim(L(\mathcal{B})) = \max_{i \in \{1, \dots, k\}} \log_r \rho(M[C_i])$.

The formula for computing the Hausdorff dimension of an ω -language given its Büchi automaton in the above theorem is implemented in Algorithm 2. The procedure STRONGLYCONNECTEDCOMPONENTS, which returns the strongly connected components of a graph defined by a set of states Q and a transition function δ , can be implemented by using Tarjan's algorithm [Tar72], which runs in $O(|Q| + |E|)$ time, linear with the sum of the number of nodes and edges in the graph.

Generalized Büchi automata. Generalized Büchi automata allow for more than one accepting set of states. It is defined as a tuple $\mathcal{G} = \{Q, \Sigma, q_0, \delta, F_{\mathcal{G}}\}$, where $F_{\mathcal{G}} = \{F_{\mathcal{G},1} \dots F_{\mathcal{G},k}\}$. An ω -word w is accepted by $\mathcal{G}$ if $\text{Inf}(w) \cap F_{\mathcal{G},i} \neq \emptyset$ for every $i \in [k]$. The corresponding Muller automaton has the same set of states and transition function, and the accepting condition is defined in terms of $F_{\mathcal{M}} = \{Q_j \in Q \mid \forall i \in [k], Q_j \cap F_i \neq \emptyset\}$.

Algorithm 2: HAUSDORFFDBA

Input: A DBA $\mathcal{B} = (Q, \Sigma, q_0, \delta, F_B)$, with adjacency matrix M
Output: Hausdorff dimension $\dim$ of $L(\mathcal{B})$

```

1  $C \leftarrow \text{STRONGLYCONNECTEDCOMPONENTS}(Q, \delta);$ 
2  $i \leftarrow 0;$ 
3 foreach  $C_j \in C$  do
4   if  $C_j \cap F_B \neq \emptyset$  then
5      $\lambda_i \leftarrow \text{Perron-Frobenius eigenvalue of } M[C_j];$ 
6      $i \leftarrow i + 1;$ 
7   end
8 end
9  $\lambda \leftarrow \max_i \lambda_i;$ 
10  $\dim \leftarrow \log_r \lambda;$ 
11 return  $\dim$ 

```

Fig. 1. DMA of $L(\phi_1)$. In this and the following DMAs, the collection of accepting sets to which a state belongs is enclosed in curly braces inside the state.

The maximal accepting subsets of this Muller automaton correspond to the strongly connected component that have nonempty intersection with each F_i . The algorithm to compute Hausdorff dimension is the same as Algorithm 2 where the **if** condition is replaced with $C_j \cap F_{G,1} \neq \emptyset \wedge \dots \wedge C_j \cap F_{G,k} \neq \emptyset$.

6. Weakness measure of GR(1) conjunctions

In this section, we first discuss the use of entropy and Hausdorff dimension in measuring the weakness of GR(1) conjunctions. We then demonstrate that these measures individually do not guarantee that the axioms in Sect. 4 hold for the general class of GR(1) conjunctions. To alleviate this problem, we introduce restrictions on GR(1) conjunctions for which these axioms hold. Then we gradually lift these restrictions, providing a more refined weakness measure satisfying the axioms for larger subclasses of GR(1) conjunctions.

6.1. Entropy and Hausdorff dimension as weakness measures

Entropy and Hausdorff dimension, defined in Sect. 5.1, provide an ordering consistent with the weakness notion defined in Sect. 4. While entropy is, by definition, the growth rate of the number of prefixes in the language as the length of the prefixes increases (and has been used in the context of quantitative model checking, see Sect. 2), we can interpret Hausdorff dimension as a measure of the asymptotic degrees of freedom of an ω -language: it quantifies how many different continuations are allowed from a prefix of an ω -word given that the corresponding Muller automaton does not leave an accepting subset.

It was already demonstrated in Examples 5.4 and 5.6 that the entropy and Hausdorff dimension can be different. In what follows, we further explore the behavior of the formula $\phi_1 = \mathbf{FG}a$ from those examples.

Example 6.1 Consider the LTL formula $\phi_1 = \mathbf{FG}a$ over the variable set $\mathcal{V} = \{a\}$ whose Muller automaton is shown in Fig. 1. Notice that, for any $w \in L(\phi_1)$, both valuations of $\mathcal{V}$ are allowed until w reaches the accepting state, and satisfying $\mathbf{G}a$ may be delayed arbitrarily. Therefore, for any finite n , $|\text{pref}_n(L)| = 2^n$, and thereby $H(L(\phi_1)) = 1 = H(L(\text{true}))$ (see the definition of entropy in Sect. 5.1). In this simple DMA, there is only a single accepting state set $S' = \{s_2\}$, with corresponding ω -language $C_{S'} = \{\{a\}^\omega\}$.

Fig. 2. DBAs of ϕ_a^{inv} (left) and ϕ_b^{inv} (right)

This implies $|pref_n(C_{S'})| = 1$, and so 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 **G** a that is eventually satisfied by any ω -word in the ω -language. This allows the strict ordering of the two formulae (ϕ_1 and $true$); one is strictly stronger than the other in the sense of implication, but has the same entropy.

The following proposition shows that both entropy and Hausdorff dimension are consistent with implication, hence satisfy Axiom 4.1.

Proposition 6.2 Given two LTL formulae ϕ_1 and ϕ_2 such that $\phi_1 \rightarrow \phi_2$ is valid, $H(L(\phi_1)) \leq H(L(\phi_2))$ and $\dim(L(\phi_1)) \leq \dim(L(\phi_2))$.

Proof If $L(\phi_1) \subseteq L(\phi_2)$, for every $n \in \mathbb{N}$ $pref_n(L(\phi_1)) \subseteq pref_n(L(\phi_2))$. Then $|pref_n(L(\phi_1))| \leq |pref_n(L(\phi_2))|$ and the relation between the entropies follows immediately.

The relation between the Hausdorff dimensions follows from the language inclusion $L(\phi_1) \subseteq L(\phi_2)$ and the monotonicity of Hausdorff dimension with respect to language inclusion [MS94]. $\square$

In addition, it follows immediately from the definition that entropy is consistent with Axiom 4.3:

Proposition 6.3 Given two LTL formulae ϕ_1 and ϕ_2 such that there exists an $\bar{n}$ such that $|pref_n(L(\phi_1))| \leq |pref_n(L(\phi_2))|$ for all $n > \bar{n}$, then $H(L(\phi_1)) \leq H(L(\phi_2))$.

Therefore, entropy is a good candidate for the weakness measure. However, it does not necessarily satisfy Axiom 4.2: there are pairs of formulae with the same entropy such that one strictly implies the other. While this was already demonstrated in Example 6.1 (for the non-GR(1) formula **FG** a and the constant $true$), next we provide two more informative examples involving GR(1) formulae only, highlighting the two reasons behind this issue: the lack of strong connectedness (Example 6.4) and the presence of fairness conditions (Example 6.5).

Example 6.4 Consider the variable set $\mathcal{V} = \{stop\}$ and the formulae $\phi_a^{inv} = \mathbf{G}stop$ and $\phi_b^{inv} = \mathbf{G}(stop \rightarrow \mathbf{X}stop)$. Their Büchi automata are shown in Fig. 2. Clearly $\phi_a^{inv} \rightarrow \phi_b^{inv}$ in the strict sense. However the two languages have the same entropy $H(L(\phi_a^{inv})) = H(L(\phi_b^{inv})) = 0$.

Example 6.5 Consider the two formulae over the variables $\mathcal{V} = \{a, b\}$: $\phi_a = \mathbf{G}(a \rightarrow \mathbf{X}b) \wedge \mathbf{GF}a$ and $\phi_b = \mathbf{G}(a \rightarrow \mathbf{X}b) \wedge \mathbf{GF}b$. The same strongly connected invariant appears in both, and therefore the two formulae have the same entropy and Hausdorff dimension, but the fairness condition in ϕ_b (i.e., **GF** b) is always satisfied when the fairness condition of ϕ_a (i.e., **GF** a) is satisfied, by virtue of the invariant itself. However, the ω -word $\{b\}^\omega$ satisfies ϕ_b but not ϕ_a . So, ϕ_a implies ϕ_b but not vice versa.

The language of both formulae, shown in Fig. 7, is not closed, but is strongly connected. Therefore, like in the case of fairness conditions, the entropy and the Hausdorff dimension both coincide with the spectral radius of the only SCC of the automaton; this SCC coincides with the automaton of the common invariant alone. Therefore,

$$H(L(\phi_a)) = H(L(\phi_b)) = \dim(L(\phi_a)) = \dim(L(\phi_b)) = \dim(L(\mathbf{G}(a \rightarrow \mathbf{X}b))) .$$

$\square$

In cases where the Hausdorff dimension is not equal to entropy, the Hausdorff dimension can help to discriminate between two formulae. In other cases, the Hausdorff dimension may not be sufficient. In such cases, we propose an additional component to the weakness measure which can help distinguish between two formulae.

Table 1. Summary of findings

Class of formulae	Weakness components	Axioms satisfied	Theorem involved
Strongly connected invariants	$H(L(\phi))$	Axiom 4.1	Proposition 6.2
		Axiom 4.2	Theorem 6.7
		Axiom 4.3	Proposition 6.3
Fairness conditions	$\dim(L(\neg \phi^{fair}))$	Axiom 4.1	Theorem 6.13
		Axiom 4.2	Theorem 6.13
		Axiom 4.3	Trivial (fairness does not affect prefixes)
Common, strongly connected invariant	1. $H(L(\phi))$ 2. $\dim(L(\phi^c)) = \dim(L(\phi^{inv} \wedge \neg \phi^{fair}))$	Axiom 4.1	Theorem 6.17
Strongly connected invariant	1. $H(L(\phi))$ 2. $\dim(L(\phi^c)) = \dim(L(\phi^{inv} \wedge \neg \phi^{fair}))$	Axiom 4.1	Corollary 6.21, Theorem 6.17
GR(1) conjunctions	1. $H(L(\phi))$ 2. $\dim(L(\phi))$ 3. $\dim(L^c) = \dim(L(C_S \wedge \neg \phi^{fair}))$	Axiom 4.1	Theorem 6.29 (under some conditions on the structure of the automata)

In the following subsections, we first identify a class of formulae where entropy alone is guaranteed to satisfy all three axioms, namely the class of *closed, strongly connected* ω -languages. For ω -languages outside of this class, we take a best-effort approach. Despite not guaranteeing Axiom 4.2, we augment the weakness measure with additional components that in some cases induce on these ω -languages a strict order consistent with strict implication.

To this end, we show how Hausdorff dimension helps in case of non-strongly connected ω -languages. Finally, for non-closed ω -languages, we identify a language decomposition based on the negation of fairness conditions such that the Hausdorff dimension of the sublanguage we define (which we call *fairness complement*, $L(\phi^{cfair})$) can discriminate between formulae with the same entropy and Hausdorff dimension. We use these three components to define a weakness measure $d(\phi)$ of a formula ϕ : the first component is the entropy $H(L(\phi))$, the second is its Hausdorff dimension $\dim(L(\phi))$, and the third is the entropy of the fairness complement $\dim(L(\phi^{cfair}))$; d induces a lexicographic ordering of GR(1) formulae using the three components. In the following, we prove that this ordering is consistent with implication for classes of ω -languages of increasing complexity: (i) ω -languages of formulae with a fixed invariant; (ii) ω -languages of formulae with strongly connected invariants; and (iii) ω -languages of GR(1) formulae with the same number of accepting SCCs. Although d does not always comply with Axiom 4.2, in several cases it induces a strict order consistent with strict implication. Table 1 lists the classes of ω -languages along with the components of the weakness measure introduced to order them and the axioms that are guaranteed to hold.

Notice that when the entropies of two formulae are equal, the order induced by d may violate Axiom 4.3. As we will show, this is due to the fact that neither the entropy nor the Hausdorff dimension take into account the strongly connected components that have lower spectral radius than the maximum (see Sect. 5.2). However, in our work we choose to prioritize the consistency with implication since we are driven by the application of GR(1) assumption refinement. Ensuring that the order relation is consistent with permissiveness in all cases is a subject for future work.

In our analysis, we do not take into account initial conditions, which in general may or may not affect the weakness measure. This is because both the entropy and the Hausdorff dimension depend only on the reachable SCCs in the formula's automaton (see Sect. 5.2). The weakness measure of $\phi^{init} \wedge \phi^{inv}$ is the same as that of ϕ^{inv} , provided that all the SCCs of an automaton of ϕ^{inv} are still reachable in the automaton of $\phi^{init} \wedge \phi^{inv}$.

6.2. Ordering invariants and strongly connected invariants

Consider the formula $\phi^{inv} = \mathbf{GB}(\mathcal{V} \cup \mathbf{X}\mathcal{V})$. The ω -language $L(\phi^{inv})$ is closed. Hence, the entropy $H(L(\phi^{inv}))$ of $L(\phi^{inv})$ coincides with its Hausdorff dimension and is computed as the Perron-Frobenius eigenvalue of the adjacency matrix of its Büchi automaton (see Sect. 5.2). From this equivalence and Definition 5.5, it is easy to

see that in this case both quantities satisfy Axiom 4.3. Therefore they can be used as a measure of weakness interchangeably.

In general, Proposition 6.2 allows one invariant to be strictly weaker than another according to implication despite both having equal entropies. Examples 6.4 and 6.5 suggest when this can happen: that is, when comparing two formulae containing fairness conditions, or when one of the formulae does not have a strongly connected automaton.

In this section, we prove the existence of a subclass of invariants for which entropy and Hausdorff dimension are strictly monotonic with respect to implication. This subclass is characterized through the concept of *strong connectedness* of an ω -language. Hereafter, given a word $w \in \text{pref}(L)$, we denote by $S_w(L)$ 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 6.6 (Strongly connected ω -language [MS94]) An ω -language L is *strongly connected* if, for every prefix $w \in \text{pref}(L)$, there exists a finite word $v \in \Sigma^*$ such that $S_{wv}(L) = L$.

In other words, an ω -language is strongly connected if and only if there exists a strongly connected automaton that represents it [MS94], that is, an automaton such that given any pair of states, each of them is reachable from the other. The DBA in Fig. 3 accepts a strongly connected ω -language, while the automaton in Fig. 2 (right) accepts a non-strongly connected one.

Using this notion, we provide in the next theorem a sufficient condition over invariants for Axiom 4.2 to be satisfied:

Theorem 6.7 Let $\phi_a^{\text{inv}} = \mathbf{G}B_a(\mathcal{V} \cup \mathbf{X}\mathcal{V})$ and $\phi_b^{\text{inv}} = \mathbf{G}B_b(\mathcal{V} \cup \mathbf{X}\mathcal{V})$ be two non-empty invariants such that $\phi_a^{\text{inv}} \rightarrow \phi_b^{\text{inv}}$ is valid, $\phi_b^{\text{inv}} \rightarrow \phi_a^{\text{inv}}$ is not valid (that is, ϕ_a^{inv} implies ϕ_b^{inv} strictly) and ϕ_b^{inv} is strongly connected. Then $H(L(\phi_a^{\text{inv}})) < H(L(\phi_b^{\text{inv}}))$ and $\dim(L(\phi_a^{\text{inv}})) < \dim(L(\phi_b^{\text{inv}}))$.

Proof The proof consists of three steps.

1. First we construct, for any invariant ϕ^{inv} , a strongly connected automaton $\mathcal{B}$ accepting $L(\phi^{\text{inv}})$ where each state keeps memory of the latest symbol read by the automaton; this construction is similar to the one in [KE12], although, unlike them, we do not need to label states with LTL subformulae.
2. Then we show that a Büchi automaton of $L(\phi_a^{\text{inv}})$ can be constructed by removing transitions and possibly states from the automaton of $L(\phi_b^{\text{inv}})$ obtained using the construction in step 1.
3. Finally, using some algebraic properties of the adjacency matrices of the two Büchi automata, we prove that the spectral radius of the automaton of $L(\phi_a^{\text{inv}})$ is less than the spectral radius of the automaton of $L(\phi_b^{\text{inv}})$. Hence $H(L(\phi_a^{\text{inv}})) < H(L(\phi_b^{\text{inv}}))$ and $\dim(L(\phi_a^{\text{inv}})) < \dim(L(\phi_b^{\text{inv}}))$ (since both entropy and Hausdorff dimension are equal to the spectral radius when the corresponding automaton is strongly connected, see Eq. 2 and Theorem 5.10).

The construction of $\mathcal{B}$ for $L(\phi^{\text{inv}})$ uses a set of states Q which are labelled by a one-to-one function $\lambda : Q \setminus \{q_0\} \rightarrow \Sigma$. The transition function is built such that

1. $\delta(q, \sigma)$ is defined if and only if every ω -word in $\lambda(q)\sigma \cdot \Sigma^\omega$ satisfies $B(\mathcal{V} \cup \mathbf{X}\mathcal{V})$;
2. in case $\delta(q, \sigma)$ is defined, $\lambda(\delta(q, \sigma)) = \sigma$.

The initial state q_0 satisfies the following properties:

1. for every $\sigma \in \Sigma$, there exists $\delta(q_0, \sigma)$ if and only if there exists $\tau \in \Sigma$ such that $\sigma\tau \cdot \Sigma^\omega \models B_b$;
2. for every $q \in Q \setminus \{q_0\}$, if $q = \delta(q_0, \sigma)$ then $\lambda(q) = \sigma$.

An example is depicted in Fig. 3.

We prove that an ω -word w satisfies ϕ^{inv} if and only if it corresponds to an infinite path on $\mathcal{B}$. Suppose $w \models \phi^{\text{inv}}$. Then we prove by induction that it induces an infinite path on $\mathcal{B}$. For every $i \in \mathbb{N}$, $w[i..] \models B(\mathcal{V} \cup \mathbf{X}\mathcal{V})$. Since $B(\mathcal{V} \cup \mathbf{X}\mathcal{V})$ constrains the first two symbols of w only, any ω -word in $w[i]w[i+1] \cdot \Sigma^\omega$ satisfies B . By construction, the automaton $\mathcal{B}$ has a transition $\delta(q_i, w[i+1]) = q_{i+1}$ such that $\lambda(q_i) = w[i]$ and $\lambda(q_{i+1}) = w[i+1]$. Therefore, if there exists a path from q_0 to q_i induced by the prefix $w[1] \dots w[i]$, there exists a path from q_0 to q_{i+1} . As a base case of the induction, consider that $w[1]w[2] \cdot \Sigma^\omega \models B$, and thereby there exists a path from q_0 to q_1 in $\mathcal{B}$.

Conversely, suppose that w is an infinite sequence of transition labels such that $\mathcal{B}(w) = q_0q_1\dots$. Then $\delta(q_i, w[i+1])$ exists for every $i \in \mathbb{N} \cup \{0\}$ and $\lambda(q_i) = w[i]$ for all $i \in \mathbb{N}$.

Fig. 3. Büchi automaton of $\mathbf{G}(x \rightarrow \mathbf{X}y)$. The state labels $\lambda(q)$ are shown inside the nodes. The initial state (not shown) has transitions towards all the states in the figure, since the first symbol is unconstrained.

By the construction of $\mathcal{B}$ this means that for every $i \in \mathbb{N}$ $w[i]w[i+1] \cdot \Sigma^\omega \models B$, which implies $w[i..] \models \phi^{inv}$. Then we can conclude $w \models \phi^{inv}$. This allows us to say that $\mathcal{B}$, so constructed, is a Büchi automaton of the formula ϕ^{inv} .

If $L(\phi)$ is strongly connected, then $\mathcal{B}$ is also strongly connected, except for the initial state. To prove this, let $q_n = \delta(q_0, w) = \delta(\delta(\delta(\dots \delta(q_0, w[1]), \dots), w[n-1]), w[n])$ be the state reached by $\mathcal{B}$ after reading the prefix $w \in \text{pref}_n(L(\phi))$, and $q_m = \delta(q_0, v)$ for $v \in \text{pref}_m(L(\phi))$. By construction, $\lambda(q_n) = w[n]$ and $\lambda(q_m) = v[m]$. Since $L(\phi)$ is strongly connected, there exist $v' \in \Sigma^*$ such that $S_{vv'} = L(\phi)$. Therefore $vv'w \in \text{pref}(L(\phi))$ and $\lambda(q_0, vv'w) = w[n]$, so $\delta(q_0, vv'w) = q_n$. So, there exists a path in $\mathcal{B}$ from q_m to q_n . Symmetrically, there exists a path from q_n to q_m . We have proved that for any pair of states reachable from q_0 there is a path between them in both directions, that is the graph induced by non-initial states is strongly connected.

Now consider the automata $\mathcal{B}_a$ and $\mathcal{B}_b$ of $L(\phi_a^{inv})$ and $L(\phi_b^{inv})$ respectively. Since the two formulae are pure invariants, $\phi_a^{inv} \rightarrow \phi_b^{inv}$ if and only if $B_a(\mathcal{V} \cup \mathbf{X}\mathcal{V}) \rightarrow B_b(\mathcal{V} \cup \mathbf{X}\mathcal{V})$. So, given the hypothesis that ϕ_a^{inv} is strictly stronger than ϕ_b^{inv} there must exist a pair of valuations σ, τ such that $\sigma\tau \cdot \Sigma^\omega \models \phi_b^{inv}$ but $\sigma\tau \cdot \Sigma^\omega \not\models \phi_a^{inv}$. By construction, this corresponds to at least one transition $\delta(q, \tau)$ that exists in $\mathcal{B}_b$ and does not in $\mathcal{B}_a$. Consequently, we can conclude that $\mathcal{B}_a$ is a proper subgraph of $\mathcal{B}_b$.

The next step is to construct the adjacency matrices corresponding to $\mathcal{B}_a$ and $\mathcal{B}_b$ excluding the respective initial states. Let $Q_a \setminus \{q_{0,a}\}$ and $Q_b \setminus \{q_{0,b}\}$ be the set of non-initial states of $\mathcal{B}_a$ and $\mathcal{B}_b$, respectively, and δ_a and δ_b their respective transition functions. Let M^a and M^b be the adjacency matrices of the graphs $(Q_a \setminus \{q_0\}, \delta_a)$ and $(Q_b \setminus \{q_0\}, \delta_b)$. Consider that by construction all transitions between two states are labelled by exactly one valuation: so, each element of these two matrices is either a 0 or a 1.

Since the transitions of δ_b are a proper subset of the transitions of δ_a , we have for each element (i, j) $M_{ij}^a \leq M_{ij}^b$, and $M_{kh}^a < M_{kh}^b$ for some (k, h) , that is to say $M_{kh}^a = 0, M_{kh}^b = 1$. Since ϕ_b^{inv} is strongly connected, its adjacency matrix M^b is irreducible (for more details on the correspondence between strongly connected digraphs and irreducible matrices see Chapter 6 of [HJ86]). Moreover, $M^a + M^b$ is also irreducible, since it is the sum of two nonnegative matrices one of which is irreducible. We can therefore apply the property stated in Chapter 2, Corollary 1.5 of [BP94], which guarantees that under the given conditions, $\rho(M^a) < \rho(M^b)$.

Taking the logarithm on both sides, we get $\dim(\phi_a^{inv}) < \dim(\phi_b^{inv})$, finishing the proof. $\square$

Remark 6.8 An interesting strongly connected invariant is the *one-state invariant*, i.e., one that does not use the $\mathbf{X}$ operator: $\phi_s^{inv} = \mathbf{G}B(\mathcal{V})$ whose DBA is shown in Fig. 4. (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:

$$\dim(\phi_s^{inv}) = \log_r |B(\mathcal{V})|$$

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 6.7. $\square$

Fig. 4. DBA of a one-state invariant.

Fig. 5. DBA of $aU b$

Remark 6.9 Typical examples of GR(1) specifications are often created manually (like the ones for device communication protocols) and use of strongly connected environment assumptions. It is indeed natural to allow environments to be reset to their initial state after some steps. However, strong connectedness may not hold especially when the specifications contain Dwyer’s patterns from [DAC99] (which can be converted into GR(1) by following the procedure described in [MR15]). As a simple example, consider the assumption $aU b$: its automaton, shown in Fig. 5, is not strongly connected. In those cases, using the Hausdorff dimension alone to compare the weakness of different formulae may lead to the problem described Example 6.4. $\square$

An interesting property that follows directly from Theorem 6.7 is the following.

Corollary 6.10 Let ϕ_a^{inv} and ϕ_b^{inv} be two invariants with the same Hausdorff dimension $\dim(L(\phi_a^{inv})) = \dim(L(\phi_b^{inv}))$, such that ϕ_b^{inv} is strongly connected and $\phi_a^{inv} \rightarrow \phi_b^{inv}$ is valid. Then ϕ_a^{inv} and ϕ_b^{inv} are equivalent.

Proof If $\phi_b^{inv} \rightarrow \phi_a^{inv}$ were not valid, then by Theorem 6.7 we would have $\dim(L(\phi_a^{inv})) < \dim(L(\phi_b^{inv}))$ and contradict the hypothesis. Therefore $\phi_a^{inv} \leftrightarrow \phi_b^{inv}$ is valid, that is, the two invariants are equivalent. $\square$

This corollary gives us a sufficient condition to determine the equivalence of two ω -languages from their entropy/Hausdorff dimension. We will use it later when defining the third component of the weakness measure.

Theorem 6.7 suggests a first version of the weakness measure for comparing strongly connected invariants. This can be either entropy or Hausdorff dimension, since both have the same value for this class of formulae. We choose entropy as it satisfies Axiom 4.3 in a wider class of formulae, as we will see.

Definition 6.11 (Weakness of strongly connected invariants) Let $\phi = \phi^{inv}$ be a strongly connected invariant. Then its weakness is defined as

$$d_1(\phi) := H(L(\phi^{inv})) . \quad (4)$$

The partial order relation of strongly connected invariants is the order induced by d_1 :

$$d(\phi_a) \leq d(\phi_b) \text{ if and only if } d_1(\phi_a) \leq d_1(\phi_b)$$

$\square$

Hence we have the following result:

Theorem 6.12 In the class of strongly connected invariants the weakness measure $d_1(\phi)$ satisfies Axiom 4.1 (by Proposition 6.2), Axiom 4.2 (by Theorem 6.7), and Axiom 4.3 (by Proposition 6.3).

6.3. Ordering fairness conditions via fairness complements

Consider the generic fairness condition $\phi^{fair} = \mathbf{GFB}(\mathcal{V})$ whose DBA $\mathcal{B}$ is shown in Fig. 6. This language is not closed: take a symbol $\sigma \in \Sigma$ that does not satisfy $B(\mathcal{V})$ and the ω -word σ^ω . It is clear that $\text{pref}(\{\sigma^\omega\}) \subseteq \text{pref}(L(\phi^{fair}))$, but $\sigma^\omega \notin L(\phi^{fair})$.

Fig. 6. DBA of $L(\phi^{fair})$ (left) and DMA of $L(\phi^{cfair})$ (right).

Fig. 7. DBAs of ϕ_a (left) and ϕ_b (right) of Example 6.5.

Notice that $\mathcal{B}$ consists of only one SCC, which is an accepting one. Therefore, the entropy is the spectral radius of the adjacency matrix of $\mathcal{B}$ (see Sect. 5.1). Analogously, the Hausdorff dimension is computed via Algorithm 2: in this case, the only accepting SCC is also the only SCC in $\mathcal{B}$. Therefore, despite the language not being closed, the entropy and the Hausdorff dimension are the same. In particular, a simple calculation shows that $H(L(\phi)) = \dim(L(\phi)) = 1 = H(L(true))$: in other words, entropy alone is insufficient to show that the constant *true* is weaker than any fairness condition that is not always true.

To allow for a distinction to be made, we propose to measure the weakness of the negation of this formula instead. In this section, we show that pairs of negated fairness conditions where one implies the other strictly can be discriminated by their Hausdorff dimensions.

We call an LTL formula of the form $\phi^{cfair} = \mathbf{FG} \neg B(\mathcal{V})$ a *fairness complement*³. It is well known that this formula cannot be represented by a DBA [GTW02]. So to compute the Hausdorff dimension we need to use Algorithm 1, which works on DMAs. The DMA of $L(\phi^{cfair})$ is shown to the right of Fig. 6. The only accepting set is $S' = \{q_2\}$. (Notice that unlike the one to the left, this automaton accepts only words that stay forever in q_2 from a certain point onwards.) The language $C_{S'}$ has an entropy of $\log_r |\neg B(\mathcal{V})|$. Hence

$$\dim(L(\phi^{cfair})) = \log_r |\neg B(\mathcal{V})| = \log_r(r - |B(\mathcal{V})|)$$

where $r = 2^{|\mathcal{V}|}$, and $|B(\mathcal{V})|$ is the number of valuations satisfying $B(\mathcal{V})$.

This formula shows that the Hausdorff dimension of the fairness complement is strictly monotonic with the implication relation, but in reverse order: the weaker the fairness condition, the lower the Hausdorff dimension is. This proves the following theorem:

Theorem 6.13 Let ϕ_a^{fair} and ϕ_b^{fair} be two fairness conditions such that $\phi_a^{fair} \rightarrow \phi_b^{fair}$ is valid and $\phi_b^{fair} \rightarrow \phi_a^{fair}$ is not valid. Then $\dim(L(\neg \phi_a^{fair})) > \dim(L(\neg \phi_b^{fair}))$.

This result suggests a version of the weakness measure that satisfies Axiom 4.2 for fairness conditions.

Definition 6.14 (Weakness of fairness conditions) Let $\phi = \phi^{fair} = \mathbf{GF} B(\mathcal{V})$ a fairness condition over the variables in $\mathcal{V}$. Define $\phi^{cfair} := \neg \phi^{fair} = \mathbf{FG} \neg B(\mathcal{V})$.

³ We will refine this notion later on.

The *weakness* of ϕ^{fair} , denoted by $d(\phi^{fair})$, is defined as

$$d_3(\phi) := \dim(\phi^{cfair}) \quad (5)$$

The order relation on the class of fairness conditions is the following:

$$d(\phi_a) \leq d(\phi_b) \text{ if and only if } d_3(\phi_a) \geq d_3(\phi_b)$$

□

In the definition of the weakness measure for the most general class of GR(1) conjunctions (presented in Sect. 6.6), we will use d_3 as the third component to be checked when measuring the weakness (this explains the use of subscript 3 in d_3).

The weakness measure d_3 satisfies all three axioms of weakness:

Theorem 6.15 The weakness measure d_3 satisfies Axioms 4.1, 4.2 and 4.3 over fairness conditions.

Proof Axiom 4.3 is trivially satisfied. In fact, no fairness condition ϕ^{fair} imposes any restriction on the finite prefixes of $L(\phi^{fair})$, and therefore $|pref_n(L(\phi^{fair}))| = r^n$, independently of ϕ^{fair} . Axioms 4.1 and 4.2 are satisfied as a direct consequence of Theorem 6.13. □

6.4. Ordering GR(1) conjunctions with a common strongly connected invariant

Consider a GR(1) conjunction $\phi = \phi^{inv} \wedge \bigwedge_{i=1}^m \phi_i^{fair}$, where ϕ^{inv} is strongly connected, and take two formulae of this kind with the same invariant. We showed in Example 6.5 that even when ϕ^{inv} is strongly connected, in some cases, entropy and Hausdorff dimension may not distinguish between the weaker and the stronger of the two in the implication sense. This problem has been previously pointed out for entropy in the work of [ABD14a] (see also Sections 2 and 8.2).

In general, entropy cannot distinguish between strongly connected languages with the same invariant. In this case, in fact, the automaton has only one SCC, which coincides with the SCC of the invariant, implying that the entropy and Hausdorff dimension are the same for both languages. We have shown this in Example 6.5, where adding any fairness condition to a strongly connected invariant does not change its entropy or Hausdorff dimension.

To capture the addition of any fairness condition to a fixed invariant, we propose using a language decomposition based on negating fairness conditions. In fact, in the previous subsection we have seen that the Hausdorff dimension can distinguish between negated fairness conditions. Let us consider a simpler case of formulae with the same invariant and a single fairness condition. As before, we exploit the fact that the negation of a fairness condition is a formula of the form $\phi^{cfair} = \mathbf{FG}\neg B(\mathcal{V})$ which can be strictly ordered through the Hausdorff dimension in a similar fashion as in Sect. 6.3. When the formula contains an invariant, however, consistency with implication is obtained by considering the Hausdorff dimension of $\phi^{inv} \wedge \phi^{cfair}$ rather than ϕ^{cfair} alone. We define the *fairness complement* in this case as the formula $\phi^c := \phi^{inv} \wedge \phi^{cfair}$.

The idea is illustrated in Fig. 8: Let $\phi_a = \phi^{inv} \wedge \phi_a^{fair}$ and $\phi_b = \phi^{inv} \wedge \phi_b^{fair}$ be two formulae having the same invariant, and suppose that ϕ_a implies ϕ_b . Suppose two formulae $\phi_a^c = \phi^{inv} \wedge \neg \phi_a^{fair}$ and $\phi_b^c = \phi^{inv} \wedge \neg \phi_b^{fair}$ having the same invariant are such that ϕ_a implies ϕ_b , as in the example above. Then the behaviours removed from ϕ^{inv} by the conjunction with ϕ_a^{fair} is a superset of the behaviours removed by ϕ_b^{fair} . The sets of behaviors removed by those conditions are the languages of the formulae $\phi_a^c = \phi^{inv} \wedge \neg \phi_a^{fair}$ and $\phi_b^c = \phi^{inv} \wedge \neg \phi_b^{fair}$. Unlike ϕ_a and ϕ_b , ϕ_a^c and ϕ_b^c can in some cases be discriminated by their Hausdorff dimension, as the following example shows.

Example 6.16 Consider again the formulae from Example 6.5. Let $\phi_a^c = \mathbf{G}(a \rightarrow \mathbf{X}b) \wedge \mathbf{FG}\neg a$ and $\phi_b^c = \mathbf{G}(a \rightarrow \mathbf{X}b) \wedge \mathbf{FG}\neg b$. The DMAs of the resulting languages are shown respectively in Fig. 9. Each of them has just one accepting singleton, so the computation of the Hausdorff dimension is straightforward: $\dim(\phi_a^c) = \frac{1}{2}$ and $\dim(\phi_b^c) = 0$. In summary, since ϕ_a is more restrictive than ϕ_b , the Hausdorff dimension of the ω -language cut out of the invariant by $\mathbf{GF}a$ is higher than the Hausdorff dimension of the behaviours excluded by $\mathbf{GF}b$. □

Fig. 8. Subset relations between $L(\phi_a)$, $L(\phi_b)$, $L(\phi_a^c)$ and $L(\phi_b^c)$ (see Theorem 6.17), $L(\phi_a)$ and $L(\phi_a^c)$ are striped from top left to bottom right and $L(\phi_b)$ and $L(\phi_b^c)$ from top right to bottom left.

Fig. 9. DMAs of ϕ_1^c (left) and ϕ_2^c (right) of Example 6.16.

When the GR(1) conjunction contains more than one fairness conditions, we can guarantee consistency with implication by extending the definition of fairness complements. For $\phi = \phi^{inv} \wedge \bigwedge_{i=1}^m \phi_i^{fair}$, define $\phi^c := \phi^{inv} \wedge \phi^{cfair}$ where $\phi^{cfair} := \bigvee_{i=1}^m \neg \phi_i^{fair}$. The next theorem shows that using this definition of a complement, two formulae can be ordered using the Hausdorff dimension.

Theorem 6.17 Let $\phi_a = \phi^{inv} \wedge \bigwedge_{i=1}^m \phi_{a,i}^{fair}$ and $\phi_b = \phi^{inv} \wedge \bigwedge_{j=1}^l \phi_{b,j}^{fair}$, such that $\phi_a \rightarrow \phi_b$ is valid. Let $\phi_a^c = \phi^{inv} \wedge \bigvee_{i=1}^m \neg \phi_{a,i}^{fair}$ and $\phi_b^c = \phi^{inv} \wedge \bigvee_{j=1}^l \neg \phi_{b,j}^{fair}$. Then $\dim(L(\phi_a^c)) \geq \dim(L(\phi_b^c))$.

Proof Since ϕ_a implies ϕ_b , $L(\phi_a) \subseteq L(\phi_b)$. Furthermore, for $i \in \{a, b\}$, $L(\phi_i) = L(\phi^{inv}) \cap L(\bigwedge_{j=1}^m \phi_{i,j}^{fair})$. Therefore, $L(\phi^{inv}) \setminus L(\bigwedge_{j=1}^m \phi_{a,j}^{fair}) \supseteq L(\phi^{inv}) \setminus L(\bigwedge_{j=1}^l \phi_{b,j}^{fair})$, i.e., $L(\phi_a^c) \supseteq L(\phi_b^c)$. Then, by monotonicity, $\dim(L(\phi_a^c)) \geq \dim(L(\phi_b^c))$, finishing the proof. $\square$

This theorem provides us with a way to increase the resolution of the weakness measure by adding to the entropy of the formula a second component measuring the size of $\phi^c := \phi^{inv} \wedge \bigvee_{i=1}^m \neg \phi_i^{fair}$. This weakness measure provides a partial ordering that applies to sets of GR(1) conjunctions with the same strongly connected invariant.

Definition 6.18 (Weakness of GR(1) conjunctions with strongly connected invariants) The *weakness* of a GR(1) conjunction $\phi = (\phi^{inv} \wedge \bigwedge_{i=1}^m \phi_i^{fair})$, denoted by $d(\phi)$, is the pair $(d_1(\phi), d_3(\phi))$ such that

$$d_1(\phi) := H(L(\phi)), \quad (6)$$

$$d_3(\phi) := \dim(L(\phi^c)) = \dim\left(L\left(\phi^{inv} \wedge \bigvee_{i=1}^m \neg \phi_i^{fair}\right)\right), \quad (7)$$

where the ω -language $L(\phi^c)$ is called the language of *fairness-violating ω -words*.

The following partial ordering is defined on the weakness measure: given $d^i = (d_1^i, d_3^i)$, with $i \in a, b$, for two GR(1) conjunctions, $d^a < d^b$ if

1. $d_1^a < d_1^b$, or
2. $d_1^a = d_1^b$ and $d_3^a > d_3^b$. $\square$

The more general Definition 6.31 uses the Hausdorff dimension $\dim(\phi)$ as the second component d_2 . With strongly connected invariants this is redundant, since a simple application of Theorem 5.10 shows that entropy and Hausdorff dimension are equal.

6.5. Ordering GR(1) conjunctions with strongly connected invariants

We now show that the weakness measure of Definition 6.18 satisfies Axiom 4.1 even when comparing two formulae with different invariants, provided the invariants are strongly connected.

More precisely, next we prove that in this class, if a formula ϕ_a implies a formula ϕ_b then the invariant of ϕ_a implies the invariant of ϕ_b . Therefore, since the invariants are strongly connected, they have the same entropy and Hausdorff dimension if and only if they are equivalent (by Corollary 6.10); finally, in case they are equivalent, the relation induced by d_3 is consistent with implication by the same argument as in Theorem 6.17.

The first step is formalized by the following theorem.

Theorem 6.19 Let $\phi_a = \phi_a^{inv} \wedge \phi_a^{fair}$ and $\phi_b = \phi_b^{inv} \wedge \phi_b^{fair}$ be two GR(1) conjunctions, and suppose ϕ_a is satisfiable and ϕ_a^{inv} is strongly connected. If $\phi_a \rightarrow \phi_b$ is valid, then $\phi_a^{inv} \rightarrow \phi_b^{inv}$ is also valid.

Notice that just the strong connectedness of ϕ_a (the stronger formula) is required for the theorem to hold.

Proof By contradiction, suppose that $L(\phi_a^{inv}) \not\subseteq L(\phi_b^{inv})$, where $\phi_a^{inv} = \mathbf{GB}_a^{inv}(\mathcal{V} \cup \mathbf{XV})$, $\phi_b^{inv} = \mathbf{GB}_b^{inv}(\mathcal{V} \cup \mathbf{XV})$, $\phi_a^{fair} = \mathbf{GF}B_a^{fair}(\mathcal{V})$, and $\phi_b^{fair} = \mathbf{GF}B_b^{fair}(\mathcal{V})$. Then there exists an ω -word w such that $w \models \phi_a^{inv}$ and $w \not\models \phi_b^{inv}$.

Since ϕ_b^{inv} does not contain nested $\mathbf{X}$ operators, there is at least one one-step transition in w violating $B_b^{inv}(\mathcal{V} \cup \mathbf{XV})$. Let i be the minimum index such that $w[i]w[i+1] \cdot \Sigma^\omega \not\models B_b^{inv}(\mathcal{V} \cup \mathbf{XV})$.

Now let us consider a word $\bar{w} \models \phi_a$ such that $\bar{w} = \bar{w}[0]\bar{w}[1] \dots (\bar{w}[f]\bar{v})^\omega$, where for every j , $\bar{w}[j] \in \Sigma$, $\bar{v} \in \Sigma^*$, and $\bar{w}[f] \models B_a^{fair}(\mathcal{V})$. This word satisfies ϕ_a , which means that it satisfies both its invariant and fairness condition.

Since ϕ_a^{inv} is strongly connected, we can construct a third ω -word $\tilde{w}$ satisfying ϕ_a^{inv} : take the first $i+1$ symbols from w , until reaching the violation of b ; then concatenate a word s that takes the automaton of ϕ_a^{inv} back to the initial state (s exists because of the strong connectedness), and finally concatenate $\bar{w}$, which is strongly connected. That is, $\tilde{w} = w[0] \dots w[i]w[i+1]s\bar{w}[0] \dots (\bar{w}[f]\bar{v})^\omega$.

The word $\tilde{w}$ satisfies ϕ_a by construction, since it satisfies its invariant and fairness condition. However, it does not satisfy ϕ_b^{inv} because it contains the transition $w[i]w[i+1]$. Therefore it does not satisfy ϕ_b , which is a contradiction, since we supposed $\phi_a \subseteq \phi_b$. Hence, we must conclude that $\phi_a^{inv} \subseteq \phi_b^{inv}$. $\square$

Example 6.20 Let $\phi_a = \mathbf{G}(a \rightarrow \mathbf{X}(b \wedge c)) \wedge \mathbf{GF}a$ and $\phi_b = \mathbf{G}(a \rightarrow \mathbf{X}b) \wedge \mathbf{GF}b$. It is easy to observe that ϕ_a implies ϕ_b , and also that the strongly connected invariant of ϕ_a alone implies the invariant of ϕ_b . $\square$

By combining this theorem with Corollary 6.10, we obtain the following result:

Corollary 6.21 (Equivalence of strongly connected invariants in GR(1) conjunctions) Let ϕ_a and ϕ_b be two GR(1) conjunctions with strongly connected invariants such that $\phi_a \rightarrow \phi_b$ is valid and $\dim(\phi_a) = \dim(\phi_b)$. Then their invariants are equivalent.

From this corollary, it immediately follows that the ordering given by the weakness measure in Definition 6.18 satisfies Axiom 4.1 even in this class of GR(1) conjunctions: if ϕ_a and ϕ_b have different entropy, then the ordering induced by it is consistent with implication by Proposition 6.2. If the formulae have the same entropy, then their invariants are equivalent by Corollary 6.21; we can therefore repeat the reasoning of Theorem 6.17 and illustrated in Fig. 8.

Fig. 10. DBAs of $L(\phi_1)$ (left) and $L(\phi_2)$ (right). The grayed-out states and edges are parts of the automata of $L(\phi_1^{inv})$ and $L(\phi_2^{inv})$, but not of $L(\phi_1)$ and $L(\phi_2)$.

Fig. 11. DMAs of $L(\phi_1^{inv}) \setminus L(\phi_1^{fair})$ (left) and $L(\phi_2^{inv}) \setminus L(\phi_2^{fair})$ (right).

6.6. Ordering generic GR(1) conjunctions

In this section, we consider generic GR(1) conjunctions where the invariant may be non-strongly connected. For simplicity of presentation we neglect initial conditions and consider only formulae with one fairness condition; however, our results can easily be extended to formulae with an initial condition and more than one fairness conditions. We start by presenting a simple example which shows that the weakness measure of Definition 6.18 is not consistent with implication (Axiom 4.1) in this general case. The example suggests that a more careful negation of the fairness condition, which we call *fairness complement*, may be helpful in extending our previous weakness measure. We then derive conditions guaranteeing that the new weakness measure is consistent with implication.

Example 6.22 Consider two GR(1) conjunctions $\phi_1 = \phi_1^{inv} \wedge \phi_1^{fair}$ and $\phi_2 = \phi_2^{inv} \wedge \phi_2^{fair}$ over the variables $\mathcal{V} = \{a, b, c\}$, where

$$\begin{aligned} \phi_1^{inv} &= \mathbf{G}(((a \wedge \neg b \wedge \neg c) \vee (\neg a \wedge b \wedge \neg c) \vee (\neg a \wedge \neg b \wedge c)) \wedge (a \rightarrow \mathbf{X}(b \vee c)) \wedge (b \rightarrow \mathbf{X}(a \vee b)) \wedge (c \rightarrow \mathbf{X}c)); \\ \phi_2^{inv} &= \mathbf{G}(((\neg b \wedge \neg c) \vee (\neg a \wedge \neg c) \vee (\neg a \wedge \neg b)) \wedge (a \rightarrow \mathbf{X}(b \vee c)) \wedge (b \rightarrow \mathbf{X}(a \vee b)) \wedge ((\neg a \wedge \neg b) \rightarrow \mathbf{X}(\neg a \wedge \neg b))); \\ \phi_1^{fair} &= \mathbf{GF}a; \\ \phi_2^{fair} &= \mathbf{GF}(a \vee b). \end{aligned}$$

The DBAs of $L(\phi_1)$ and $L(\phi_2)$ are shown in Fig. 10. The state s_2 in both automata is redundant, since no run reaching s_2 could be accepted, but it has been depicted since it is part of the automata of the invariants alone.

Fig. 12. DBAs of the ω -languages of accepted suffixes L_{S_1} (left) and L_{S_2} (right).

It is easy to see (from the figure) that the entropy of the two ω -languages is equal and ϕ_1 implies ϕ_2 . However, if we compute the weakness $d = (d_1, d_3)$ according to Definition 6.18, ϕ_1 has a higher weakness than ϕ_2 . In fact, $L(\phi_1^c)$ is the set of words eventually keeping the automaton always in s_1 or always in s_2 , and $L(\phi_2^c)$ is the set of words that eventually end up stuck in s_2 (recall that we defined $\phi^c = \phi^{inv} \wedge \neg \phi^{fair}$ for $\phi = \phi^{inv} \wedge \phi^{fair}$). This defines the DMA of $L(\phi_1^c)$ and $L(\phi_2^c)$ in Fig. 11, and hence, by (3) (and executing algorithm HAUSDORFFDMA), we obtain

$$d_3(\phi_1) = \dim(L(\phi_1^c)) = \dim(L(\phi_1^{inv}) \setminus L(\phi_1^{fair})) = \log_8 1 = 0;$$

$$d_3(\phi_2) = \dim(L(\phi_2^c)) = \dim(L(\phi_2^{inv}) \setminus L(\phi_2^{fair})) = \log_8 2 = 1/3.$$

Given the order relation induced by this measure, ϕ_1 incorrectly appears as the weaker formula, since it has smaller d_3 measure. Thus Axiom 4.1 is violated.

The problem in this example stems from using the language of the invariant above as the 'basis' for negating the fairness condition. Next we show that if the definition of the fairness complement changes (i.e., it is defined differently from $L(\phi^c) = L(\phi^{inv}) \setminus L(\phi^{fair})$) then the resulting quantity, while reducing to the previous one for strongly-connected invariants, reflects weakness better.

Definition 6.23 (Fairness complement) Given a GR(1) conjunction $\phi = \phi^{inv} \wedge \phi^{fair}$, the *fairness complement* $L^c = L_b \setminus L(\phi^{fair})$ is the difference between some ω -language L_b , called the *base language*, and the language of ϕ^{fair} , where the base language L_b is a function of the formula ϕ satisfying the following property: if $\phi_1 \rightarrow \phi_2$ is valid and $H(L_b(\phi_1)) = H(L_b(\phi_2))$, then $L_b(\phi_1) = L_b(\phi_2)$.

This definition allows to prove consistency with implication for $\dim(L^c)$ in the same way as Theorem 6.17: when L_b is a suitable base language and a GR(1) conjunction ϕ_a implies a second one ϕ_b , the sets $L_b(\phi_a)$, $L_b(\phi_b)$, $L(\phi_a^{fair})$, $L(\phi_b^{fair})$, $L(\neg \phi_a^{fair})$, $L(\neg \phi_b^{fair})$ are in the same relation as pictured in Fig. 8, where $L(\phi^{inv})$ is replaced by $L_b(\phi_a) = L_b(\phi_b)$.

Since the Hausdorff dimension of an ω -language is essentially determined by the behavior of its suffixes, we consider the language generated by these suffixes as a candidate of the base language in the above definition. To define the language of suffixes, consider a GR(1) conjunction $\phi = \phi^{inv} \wedge \phi^{fair}$ with Büchi automaton $\mathcal{B} = \langle Q, \Sigma, q_0, \delta, F \rangle$. We call an SCC S *accepting* if it has nonempty intersection with F , (i.e., $S \cap F \neq \emptyset$). Given an ω -word $w \in L(\phi)$ whose induced run ends in the accepting SCC S , we call its *accepted index* any index i_S such that for every $i \geq i_S$, $\delta(q_0, w[1..i]) \in S$; we call $w^S = w[i_S, i_S + 1, \dots]$ an *accepted suffix* of w , and we say that the SCC S *accepts* the suffix w^S . We denote by L_S the set of all suffixes accepted by S , and by $L_{suff} = \bigcup_{S \in \text{accepting SCCs of } \mathcal{B}} L_S$ the set of all suffixes accepted by some SCC of $\mathcal{B}$. L_S and L_{suff} are called the ω -languages of accepted suffixes by S and $\mathcal{B}$, respectively.

Example 6.24 The automata of ϕ_1 and ϕ_2 in Example 6.22 contain only one accepting SCC each. The languages of accepted suffixes are shown in Fig. 12.

Note that, by the finiteness of a Büchi automaton, any ω -word accepted by the automaton has a suffix accepted by one of its SCCs. Also notice that the definition of L_S depends on the particular automaton representing the ω -language $L(\phi)$. However, some properties are invariant to the particular automaton of $L(\phi)$ taken into account. In particular, the next theorem shows that the language of suffixes accepted by an SCC is monotonic under implication:

Theorem 6.25 Let ϕ_a and ϕ_b be two GR(1) conjunctions with corresponding Büchi automata $\mathcal{B}_a$ and $\mathcal{B}_b$. Suppose $\phi_a \rightarrow \phi_b$ is valid. Then for every set of suffixes $W \subset \Sigma^\omega$ accepted by an SCC of $\mathcal{B}_a$ there exists an SCC of $\mathcal{B}_b$ accepting every element of W . In particular, it follows that for every accepting SCC S_a in $\mathcal{B}_a$ there exists an accepting SCC S_b in $\mathcal{B}_b$ such that $L_{S_a} \subseteq L_{S_b}$.

The proof of the theorem uses the following simple lemma, which allows appending an SCC-accepted prefix to a word that reaches the same SCC:

Lemma 6.26 Consider a Büchi automaton $\mathcal{B}$ with an SCC S . Let w_1 be an ω -words accepted by $\mathcal{B}$ such that the path induced by w_1 reaches S , that is, there is an index $j \in \mathbb{N}$ such that $s_1 = \delta(q_0, w_1[1, \dots, j]) \in S$. Furthermore, let $w_2 \in \Sigma^\omega$ be a suffix accepted by S . Then there is a $v \in \Sigma^*$ such that $w = w_1[1, \dots, j]vw_2$ is accepted by $\mathcal{B}$ and the path induced by w remains in S after processing the first j symbols; that is, $\delta(q_0, w[1, \dots, k]) \in S$ for all $k \geq j$.

Proof Since w_2 is accepted by S , there is a prefix $v_2 \in \Sigma^*$ such that v_2w_2 is accepted by $\mathcal{B}$ and its run stays in S after v_2 , that is, $s_2 = \delta(q_0, v_2) \in S$ and $\delta(q_0, v_2w_2[1, \dots, i]) \in S$ for all $i \in \mathbb{N}$. Since S is strongly connected, there is a $v \in \Sigma^*$ that leads from s_1 to s_2 while not leaving S . Therefore, $w_1[1, \dots, j]vw_2$ is accepted by $\mathcal{B}$ and belongs to S after the first j steps. $\square$

Using this result, we can easily prove Theorem 6.25.

Proof The proof is done by contradiction. Suppose there is an SCC S_a of $\mathcal{B}_a$ that accepts every suffix in W , but for every SCC S_b in $\mathcal{B}_b$ there is at least one suffix in W that is not accepted by S_b . Let B denote the number of distinct SCCs of $\mathcal{B}_b$. The contradiction will be achieved by creating an ω -word which belongs to $L(\phi_b)$ but has to visit more than B SCCs of $\mathcal{B}_b$.

Let $w_0^* \in W$. By definition, there exists a finite word $v_0 \in \Sigma^*$ of length $|v_0|$ such that $w_0 = v_0w_0^* \in L(\phi_a)$ and the automaton remains in the SCC S_a of $\mathcal{B}_a$ after processing the first $|v_0|$ symbols and reaching $s_0 = \delta(q_{0,a}, v_0) \in S_a$. Since $\phi_a \rightarrow \phi_b$, $w_0 \in L(\phi_b)$. Let $S_{b,1}$ denote an SCC reached in $\mathcal{B}_b$ while processing the w_0^* -part of this word; that is, there exists $j_1 > |v_0|$ such that $\delta(q_{0,b}, w_0[1, \dots, j_1]) \in S_{b,1}$. By our assumption, there is a $w_1^* \in W$ that is not accepted by $S_{b,1}$. On the other hand, w_1^* is a suffix accepted by S_a , hence by Lemma 6.26, there is a finite word $v_1 \in \Sigma^*$ such that $w_1 = w_0[1, \dots, j_1]v_1w_1^* \in L(\phi_a)$ and its induced path on $\mathcal{B}_a$ belongs to S_a after processing the first $|v_0|$ symbols. Since $\phi_a \rightarrow \phi_b$, w_1 also belongs to $L(\phi_b)$. Furthermore, since $S_{b,1}$ does not accept w_1^* , the induced path of w_1 on $\mathcal{B}_b$ must reach an SCC $S_{b,2}$, after reaching $S_{b,1}$, which is distinct from $S_{b,1}$. That is, $\delta(q_{0,b}, w_1[1, \dots, j_2]) \in S_{b,2}$, and since the first j_1 symbols of w_0 and w_1 coincide, $\delta(q_{0,b}, w_1[1, \dots, j_1]) \in S_{b,1}$. Continuing with this construction, using $w_2^* \in W$ not accepted by $S_{b,2}$, we can create $w_2 = w_1[1, \dots, j_2]v_2w_2^*$ which belongs to $L(\phi_a)$, and hence to $L(\phi_b)$, reaches the SCC $S_{b,1}$, $S_{b,2}$ after $j_1 < j_2$ steps, respectively, and since $S_{b,2}$ does not accept w_2^* , it also reaches another SCC of $\mathcal{B}_b$, $S_{b,3}$, which is obviously distinct from $S_{b,2}$, but also distinct from $S_{b,1}$, since otherwise $S_{b,1}$ and $S_{b,3}$ would belong to the same SCC. Repeating this construction for B steps, we end up with a word w_B that reaches $B + 1$ distinct SCCs of $\mathcal{B}_b$, which is a contradiction, as the automaton only has B SCCs. Therefore, $\mathcal{B}_b$ must have an SCC accepting all suffixes in W . The statement about the language inclusion then follows trivially. $\square$

The base language L_b . Despite being an essential element in constructing the base language of the fairness complement, the ω -language of accepted suffixes is not sufficient. Indeed, Example 6.24 shows a case of two formulae such that one implies the other, they have the same entropy and Hausdorff dimension, but the languages of accepted suffixes are different: it is easy to see from Fig. 12 that L_{S_1} only contains ω -words that contain the symbol $\{a\}$ infinitely many times, while L_{S_2} contains any sequence of $\{a\}$ and $\{b\}$.

The problem is that, unlike ϕ^{inv} in Sect. 6.5, the language L_S of suffixes accepted by the SCC S (or L_{suff} in case of multiple accepting SCCs) is not closed. To alleviate this problem, we are going to use its closure $C_S = C(L_S)$ as the base language. The effect of applying the closure operator to an ω -language accepted by a DBA is producing a DBA with the same graph structure where every co-reachable state (that is, a state from which an accepting state is reachable) becomes accepting itself. Therefore, the closure operation does not change the entropy. Moreover, it can be seen easily that $C_S \cap L(\phi^{fair}) = L_S$.

In Example 6.24, it is clear that $C_{S_1} = C_{S_2} = L_{S_2}$, that is, the two closures are equal. The following theorem shows that this connection holds more generally, which is one of the main motivations for choosing $L_b = C_S$.

Theorem 6.27 Let ϕ_a and ϕ_b be two GR(1) conjunctions. Suppose that $\phi_a \rightarrow \phi_b$ is valid and $\dim(L(\phi_a)) = \dim(L(\phi_b)) = H$. Then, for every accepting SCC S_a of $\mathcal{B}_a$ such that $H(L_{S_a}) = H$, there exists an SCC S_b of $\mathcal{B}_b$ such that $C_{S_a} = C_{S_b}$.

Proof By Theorem 6.25 for every SCC S_a in $\mathcal{B}_a$ there exists an SCC S_b in $\mathcal{B}_b$ such that $L_{S_a} \subseteq L_{S_b}$. This implies $\mathcal{C}(L_{S_a}) = C_{S_a} \subseteq \mathcal{C}(L_{S_b}) = C_{S_b}$.

Now assume that the entropy of the language L_{S_a} is equal to H , the Hausdorff dimension of the entire language. Then $H(L_{S_b}) \geq H$. Since $\dim(L(\phi_b)) = H$ by assumption and $H(L_{S_b}) \leq \dim(L(\phi_b))$, we have $H(L_{S_b}) = H$. Hence, also their respective closures have the same entropy: $H(C_{S_a}) = H(C_{S_b}) = H$.

Now, since C_{S_a} and C_{S_b} are closed and strongly connected, their entropy is strictly monotonic with their subset relation by Theorem 6.7. That is, $C_{S_a} \subseteq C_{S_b}$ and $H(C_{S_a}) = H(C_{S_b})$ imply $C_{S_a} = C_{S_b}$. $\square$

The theorem allows us to define a base language as the union of the closure of the accepted suffix languages under some additional conditions, which will be explained later. To be precise, let $\mathcal{B}$ be a Büchi automaton for a GR(1) conjunction ϕ , and let $\mathcal{S}_H(\phi)$ denote the set of its SCCs whose language of accepted suffixes has entropy $\dim(L(\phi))$. Then the base language of L can be defined as

$$L_b(\phi) = \bigcup_{S \in \mathcal{S}_H(\phi)} C_S. \quad (8)$$

Theorem 6.27 shows that if ϕ_a and ϕ_b satisfy the conditions of the theorem, $L_b(\phi_a) \subseteq L_b(\phi_b)$. To be able to use L_b as a base language, we need some extra conditions guaranteeing that $L_b(\phi_a) = L_b(\phi_b)$ also holds. Suppose that C_{S_a} and C_{S_b} are unique for $S_a \in \mathcal{S}_H(\phi_a)$ and $S_b \in \mathcal{S}_H(\phi_b)$, respectively (i.e., among the accepting SCCs of $\mathcal{B}_a$ and $\mathcal{B}_b$ whose entropy achieves the Hausdorff dimension, respectively); note that this assumption seems to be satisfied in practice. Assume furthermore that there is a one-to-one correspondence between S_a and S_b in the theorem. Then $L_b(\phi_a) = L_b(\phi_b)$, which is equivalent to

$$\bigcup_{S_a \in \mathcal{S}_H(\phi_a)} C_{S_a} = \bigcup_{S_b \in \mathcal{S}_H(\phi_b)} C_{S_b},$$

holds if the number of SCCs considered for ϕ_b is not greater than that for ϕ_a . The number of such SCCs is captured in the next definition.

Definition 6.28 (Order of strong connectedness) Given a formula ϕ and its Büchi automaton DBA $\mathcal{B}$, the *order of strong connectedness* (or simply *order*) of $\mathcal{B}$ is defined as $|\mathcal{S}_H(\phi)|$, that is, the number of accepting SCCs of $\mathcal{B}$ such that the entropy of (the closure of) their accepted suffix language equals $\dim(L(\phi))$.

The next theorem shows that under the above assumptions and definition of L_b , the fairness complement can be used to compare two GR(1) conjunctions of the same order and entropy.

Theorem 6.29 Let $\phi_a = \phi_a^{inv} \wedge \phi_a^{fair}$ and $\phi_b = \phi_b^{inv} \wedge \phi_b^{fair}$. Suppose $\phi_a \rightarrow \phi_b$ is valid, and $\dim(\phi_a) = \dim(\phi_b) = H$. Let $\mathcal{B}_a$ and $\mathcal{B}_b$ be two DBAs of the languages of the respective formulae having the same order, and assume all their accepting SCCs are unique in the sense that their closures are different. Let L_a^c and L_b^c denote, respectively, the fairness complements of ϕ_a and ϕ_b with the base language L_b defined in (8). Then $\dim(L_a^c) \geq \dim(L_b^c)$.

Proof Theorem 6.25 implies that $\bigcup_{S \in \mathcal{S}_H(\phi_a)} L_S \subseteq \bigcup_{S \in \mathcal{S}_H(\phi_b)} L_S$. Moreover, by the definition of the closure operation, $\bigcup_{S \in \mathcal{S}_H(\phi_a)} L_S = \bigcup_{S \in \mathcal{S}_H(\phi_a)} C_S \cap L(\phi_a^{fair})$ and a similar equation also holds for ϕ_b . Furthermore, the argument preceding the theorem and our assumption on the order of the automata and the uniqueness of the SCCs imply that $L_b(\phi_a) = L_b(\phi_b)$, which we denote by L_b . Combining the above we get $L_b \cap L(\phi_a^{fair}) \subseteq L_b \cap L(\phi_b^{fair})$, implying $L_b \setminus L(\phi_a^{fair}) \supseteq L_b \setminus L(\phi_b^{fair})$. The latter two sets are, respectively, L_a^c and L_b^c by definition. Hence, by the monotonicity of Hausdorff dimension, we conclude that $\dim(L_a^c) \geq \dim(L_b^c)$. $\square$

Remark 6.30 Notice that the closure of the language of ϕ , $\mathcal{C}(L(\phi))$ cannot be used as the base language (see Definition 6.23). For instance, consider two GR(1) conjunctions ϕ_1 and ϕ_2 defined by the DBAs in Fig. 13. It is easy to see that $\phi_1 \rightarrow \phi_2$ is valid and ϕ_1 and ϕ_2 have the same entropy (since ϕ_2 has twice as many prefixes as ϕ_1 , for any length). At the same time, since the DBA of the closure of ϕ_i can be obtained by making every state accepting, the two formulae have different closures. Hence, they cannot be used as a base language.

Fig. 13. DBAs of ϕ_1 (left) and ϕ_2 (right)

Weakness measure in the general case. This allows us to provide the following general definition of the weakness measure and its order relation, which is only defined among formulae with the same order of strong connectedness. The weakness measure defined as such satisfies our original goal of provable consistency with implication.

Definition 6.31 (Weakness of generic GR(1) conjunctions) Let $\phi = \phi^{inv} \wedge \phi^{fair}$ be a GR(1) formula. Its *weakness* $d(\phi)$ is defined as a tuple $d(\phi) := (d_1(\phi), d_2(\phi), d_3(\phi))$, where

- $d_1 = H(L(\phi))$
- $d_2 = \dim(L(\phi))$
- $d_3 = \dim(L^c)$

where L^c is the fairness complement of ϕ with L_b defined in (8).

Given two formulae ϕ_a and ϕ_b with the same order, we say that $d(\phi_a) < d(\phi_b)$ (ϕ_b is *weaker* than ϕ_a) if and only if

- $d_1(\phi_a) < d_1(\phi_b)$, or
- $d_1(\phi_a) = d_1(\phi_b)$ and $d_2(\phi_a) < d_2(\phi_b)$, or
- $d_1(\phi_a) = d_1(\phi_b)$, $d_2(\phi_a) = d_2(\phi_b)$, and $d_3(\phi_a) > d_3(\phi_b)$.

□

Notice that, if ϕ^{inv} is strongly connected, this definition reduces to the first version we have given in the previous subsection. Under the assumption that the closures of the SCCs of the DBA of ϕ_a and ϕ_b are unique, Proposition 6.2 and Theorem 6.29 imply that this weakness measure is consistent with implication (i.e., Axiom 4.1 is satisfied).

Remark 6.32 The results provided so far assume that the formula contains only one fairness condition. If the formula $\phi = \phi^{inv} \wedge \bigwedge_j \phi_j^{fair}$ contains more than one fairness conditions, the same theorems presented in this section can be proved (as they depend on the structure of the automaton, not on the syntax of the formula). In this case, $L(\phi)$ is represented by a generalized deterministic Büchi automaton (GDBA, see Sect. 5.2). In a GDBA $\mathcal{G} = \langle Q, \Sigma, q_0, \delta, \{F_{\mathcal{G},1}, \dots, F_{\mathcal{G},k}\} \rangle$, an SCC S is called *accepting* if it has a non-empty intersection with every accepting set, that is, $S \cap F_{\mathcal{G},j} \neq \emptyset$ for every j . Similarly, the definition of the *fairness complement* L^c can be extended to this case as $L^c = L_b \setminus \bigcap_{j=1}^k L(\phi_j^{fair})$, where L_b is the same base language as defined in (8).

With these two small variations, Proposition 6.2 and Theorem 6.29 can be proved with the same statements (where DBAs are replaced with GDBAs) and following the same steps. This leads to proving that the weakness measure given in Definition 6.31 is consistent with implication also when ϕ has multiple fairness conditions.

Notice that in this general case, d_3 is defined as

$$d_3(\phi) := \dim \left(\bigcup_i C_{S_{max,i}} \setminus L \left(\bigwedge_j \phi_j^{fair} \right) \right) = \dim \left(\bigcup_i C_{S_{max,i}} \cap \bigcup_j L(\neg \phi_j^{fair}) \right),$$

and the relation $d(\phi_a) < d(\phi_b)$ is defined in the same way as in Definition 6.31. By applying the property

of *countable stability* of Hausdorff dimension [Sta15], which states that for any countable set of ω -languages $L_1, L_2, \dots \subseteq \Sigma^\omega$,

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

d_3 can be expressed as

$$d_3(\phi) = \max_{i,j} \dim \left(C_{S_{max,i}} \cap L(\neg \phi_j^{fair}) \right),$$

thereby reducing the definition to the base case of a formula with one strongly connected component and one fairness condition only.

7. Computing the Hausdorff dimension of fairness complements

In this section, we describe how to compute the component $d_3(\phi) = \dim(L^c)$ of the weakness measure given in Definition 6.31. We show how to compute the accepting states of the DMA of L^c from the DBA $\mathcal{B}$ of $L(\phi)$. From this, the Hausdorff dimension is computed via Algorithm 1.

We focus on the simple case $\phi = \phi^{inv} \wedge \phi^{fair}$ having a single fairness condition, and such that $\mathcal{B}$ has a single accepting SCC S with maximum entropy (or equivalently, $L(\phi)$ has the order of strong connectedness 1). We saw in Sect. 6.6 that the case of multiple fairness conditions is reduced to the case of a single one. Moreover, when multiple accepting SCCs are present, the procedure we give simply iterates over these SCCs and returns the maximum over the iterations.

As we saw in Sect. 6.6, in order to compute d_3 we need to compute a Muller automaton $\mathcal{M} = \langle Q_{\mathcal{M}}, \Sigma, q_{\mathcal{M},0}, \delta_{\mathcal{M}}, T_{\mathcal{M}} \rangle$ of the fairness complement $L^c = C_S \setminus L(\mathbf{GF} B^{fair})$ (see Definition 6.23). Actually, it is sufficient to compute the subgraph of $\mathcal{M}$ induced by its accepting sets of states $S' \in T_{\mathcal{M}}$, since they are the only components that affect the Hausdorff dimension (see Algorithm 1).

We, therefore, provide an algorithm to compute the accepting set(s) of the Muller automaton from the Büchi automaton of $L(\phi)$. The procedure is shown in Algorithm 3. The algorithm takes as input the subgraph of $\mathcal{B}$ induced by its accepting SCC S , denoted by $\mathcal{B}[S]$. Lines 1-5 compute the adjacency matrix A_T of the accepting sets $T_{\mathcal{M}}$ of the DMA $\mathcal{M}$. Line 3 computes the element $[i, j]$ of the adjacency matrix as the number of symbols connecting q_i to q_j in $\mathcal{B}[S]$ that violate B^{fair} .

Algorithm 3: COMPUTED3ONSCC

Input: $\mathcal{B}[S] = \langle S, \Sigma, s_0, \delta^S, S \rangle$: DBA of C_S with adjacency matrix A

Input: $\phi_j^{fair} = \mathbf{GF} B_j^{fair}(\mathcal{V})$: fairness condition

Output: $d_{3,S,j}$: Hausdorff dimension of $C_S \cap L(\neg \phi_j^{fair})$

```

1 for  $i = 1$  to  $|S|$  do
2   for  $j = 1$  to  $|S|$  do
3      $A_T[i, j] \leftarrow A[i, j] - |\{\sigma \mid \delta^S(s_i, \sigma) = s_j \text{ and } \sigma \models B^{fair}\}|;$ 
4   end
5 end
6  $T_{\mathcal{M}}^{\max} \leftarrow \text{STRONGLYCONNECTEDCOMPONENTS}(A_T);$ 
7 foreach  $C_j \in T_{\mathcal{M}}^{\max}$  do
8    $\lambda_j \leftarrow \text{Perron-Frobenius eigenvalue of } A_T[C_j];$ 
9 end
10  $\lambda \leftarrow \max_j \lambda_j;$ 
11  $d_{3,S,j} \leftarrow \log_r \lambda;$ 
12 return  $d_{3,S,j}$ 
```

Line 6 computes the SCCs of the graph described by A_T . These correspond to the maximal accepting sets of $\mathcal{M}$ (for a definition of maximal accepting sets, see Sect. 5.2). Hence, lines 7-11 run Algorithm 1 on these accepting sets to compute the desired Hausdorff dimension d_3 .

Fig. 14. DBA $\mathcal{B}$ of the language of $\phi = \mathbf{G}((x \rightarrow \mathbf{X}y) \wedge ((x \vee y) \rightarrow \mathbf{X}(x \vee y)) \wedge \mathbf{GF}x$ (left) and DBA $\mathcal{B}[S]$ of the language C_S (right)

The correctness of this algorithm is based on the following theorem.

Theorem 7.1 (Correctness of Algorithm 3) Let $\mathcal{B}[S]$ be the subgraph induced on $\mathcal{B}$ by its accepting SCC S . Let $T_{\mathcal{M}}$ be the collection of accepting sets of the DMA $\mathcal{M}$ of $L^c = C_S \setminus L(\mathbf{GF}B^{fair})$. Then A_T computed by Algorithm 3 is the subgraph of $\mathcal{M}$ induced by the states in $T_{\mathcal{M}}$.

Proof (Sketch) We sketch the proof of this statement by constructing the DMA $\mathcal{M}$ and showing that its accepting sets of states induce a subgraph described by A_T .

The construction starts with the subgraph induced on the DBA $\mathcal{B}$ of ϕ by the accepting SCC S . Formally, the Büchi automaton of ϕ is defined as $\mathcal{B} = \langle Q, \Sigma, q_0, \delta, F \rangle$. The restriction to S yields the DBA $\mathcal{B}[S] = \langle S, \Sigma, s_0, \delta^S, S \rangle$, which is, by construction, the automaton of C_S (recall that C_S is the closure of L_S , i.e., the set of suffixes accepted by S , see Sect. 6.6). The initial state $s_0 \in S$ is picked arbitrarily, since the Hausdorff dimension does not depend on the choice of the initial state [Sta98]). Here, $\delta^S(s, \sigma)$ is the restriction of δ to $S \times \Sigma$.

Let $S_{\mathcal{M}} = S \times \{0, 1\}$. The states of the kind $(s, 0)$ are the ones where the last symbol read does not satisfy the fairness boolean expression B^{fair} , while those of the kind $(s, 1)$ are the ones where the last symbol read satisfies it. Accordingly, the transition function is defined as

$$\delta_{\mathcal{M}}((s, i), \sigma) = \begin{cases} (s', 0) & \text{if } \delta^S(s, \sigma) = s' \text{ and } \sigma \not\models \neg B^{fair} \\ (s', 1) & \text{if } \delta^S(s, \sigma) = s' \text{ and } \sigma \models \neg B^{fair} \end{cases} \quad \forall s, s' \in S, \forall i \in \{0, 1\}.$$

The initial state is $s_{M,0} = (s_0, 1)$. The accepting condition corresponds to any strongly connected subset of states $(s, 1)$, so that any path remaining inside these states is guaranteed to satisfy $\mathbf{G}\neg B^{fair}$. Any ω -word ending in these states and eventually remaining there satisfies $\mathbf{FG}\neg B^{fair}$, and since the graph obtained is a subgraph of C_S , every such path of this graph is a suffix of an ω -word in C_S . Hence the graph induced by the states $(s, 1)$ comprises all the accepted sets of the DMA $\mathcal{M}$ of $C_S \cap L(\mathbf{FG}\neg B^{fair})$. By construction, it is easy to see that the adjacency matrix A_T of this graph is derived from $\mathcal{B}[S]$ by subtracting from each element $[i, j]$ the number of edges from state i to j satisfying B^{fair} . Figure 15 (left) shows an example of this construction for the language in Fig. 14.

The Muller automaton $\mathcal{M}$ needs not be computed explicitly, since only the accepting sets contribute to the Hausdorff dimension. It is sufficient to consider the subgraph of $\mathcal{B}[S]$ and remove all the transitions labeled with valuations that do not satisfy $\neg B^{fair}$. The maximal accepting subsets of $\mathcal{M}$ (see Sect. 5) coincide with the SCCs of the resulting graph. $\square$

Figures 14-15 show an example of this construction. The input to the algorithm is the graph of $\mathcal{B}[S]$ shown in Fig. 14 (right), and the graph described by the adjacency matrix A_T is shown in Fig. 15 (right).

Fig. 15. DMA of the language $C_S \cap L(\mathbf{FG} \vdash x)$ referred to the language in Fig. 14 (left), and subgraph including the accepting states only (right). In the left-hand automaton, the edge labels are omitted to avoid cluttering: each edge's label corresponds to the valuation labelling the destination state of the edge. The singleton of the state $(\{y\}, 1)$ in gray is the only one accepting any ω -word.

8. Evaluation

We evaluate here the weakness measure in Definition 6.31 by demonstrating its applicability for GR(1) assumption refinement. To demonstrate its generality, we show an additional application to the problem of 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 [DAV]. Our implementation makes use of the Spot tool [DLLF⁺16] for LTL-to-automata conversion, and NumPy for eigenvalue computations.

8.1. GR(1) assumption refinement

We investigate the use of the weakness measure in GR(1) assumption refinement approaches [AMT13, AMT15, LDS11, CA17, MRS19], as introduced in Sect. 2. We focus on approaches structured as a search in the space of GR(1) assumptions. The outcome of these approaches is a *refinement tree*, i.e., a tree structure in which each node is associated with a GR(1) conjunction representing environment assumptions. Let ϕ denote 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 these approaches 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) assumption 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 [AMT13, BJP⁺12, LDS11]. 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 ϕ^ε see [AMT13]. 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 [BJP⁺12]. To our knowledge, the AMBA08 specification is one of the biggest benchmarks available in this field, with 55 binary variables, 28 initial assumptions and 157 guarantees.

We show the use of the weakness measure in two ways. First we compare the use of weakness with the use of implication when ordering refinements. Specifically, we provide examples of pairs of refinements that can be ordered by both approaches and refinements that can only be ordered by using our weakness measure.

Secondly, we propose using the weakness measure as a heuristic to drive the search for refinements towards weaker formulae compared with the common breadth-first approach to this search problem used in [CA17, AMT13, MRS19, LDS11]. We show that, despite the fact that this heuristic-based approach produces fewer solutions in the given time, it can reach weaker solutions that are missed by a basic breadth-first search of the refinement tree.

We also show that the computational bottleneck in ordering refinements is the automata computation (i.e., the generation of automata from GR(1) conjunctions [Var96]). Furthermore, implication requires more computation time than weakness to provide an ordering of the entire set of refinements generated during the search process. This part of the evaluation follows a similar methodology to the evaluation presented in [CAG18]. The main difference is that in [CAG18] we used refinements which were generated in [CA17] using an approach that applies breadth-first search and was automated only in part. This in effect slowed down the tree generation process and resulted in smaller trees. Here, instead, we use the larger refinement trees generated by our heuristic search. We conclude the evaluation process by using the comparison data between implication and weakness to validate the correctness of our implementation.

8.1.1. Ordering via weakness and implication

Consistency between weakness and implication. We first consider the lift controller example. Two refinements computed by the automated approach in [CA17] 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: $d(\phi^\varepsilon \wedge \phi_1) = (0.7746, 0.7746, 0)$ and $d(\phi^\varepsilon \wedge \phi_2) = (0.7925, 0.7925, 0.5)$. Notice that $d_1(\phi^\varepsilon \wedge \phi_1) < d_1(\phi^\varepsilon \wedge \phi_2)$ 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 $d(\phi^\varepsilon \wedge \phi_2) = (0.7925, 0.7925, 0.5)$ and $d(\phi^\varepsilon \wedge \phi_3) = (0.7925, 0.7925, 0.695)$. Here, d_1 and d_2 are equal for both formulae and $d_3(\phi^\varepsilon \wedge \phi_2) < d_3(\phi^\varepsilon \wedge \phi_3)$; 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: $d(\phi^\varepsilon \wedge \phi_3) = (0.7925, 0.7925, 0.695)$ and $d(\phi^\varepsilon \wedge \phi_4) = (0.7925, 0.7925, 0.5975)$. This confirms that ϕ_4 removes less behaviours from ϕ^ε 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 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 $\phi_6 = \mathbf{G}\neg \text{alarm}$ yields $d(\phi^\varepsilon \wedge \phi_5) = (0.3694, 0.3694, 0.3207)$ and $d(\phi^\varepsilon \wedge \phi_6) = (0.3746, 0.3746, 0.3346)$. 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 error traces ending in the *stop* state, being then a weaker restriction on the combined behaviors of the controller and the environment.

The following two assumption refinements are computed for the AMBA-AHB case study, instead, with two masters: $\psi_1 = \mathbf{G}(\neg h\text{busreq}_1 \vee \mathbf{X}(h\text{ready} \vee \neg h\text{busreq}_1))$; and $\psi_2 = \mathbf{G}((\neg h\text{grant}_1 \wedge h\text{ready} \wedge h\text{busreq}_1) \rightarrow \mathbf{X}(\neg h\text{ready} \vee \neg h\text{busreq}_1))$. As in the case of the lift example, neither formula implies the other. The weakness of the resulting assumptions is: $d(\psi^\varepsilon \wedge \psi_1) = (0.9503, 0.9503, 0.9068)$ and $d(\psi^\varepsilon \wedge \psi_2) = (0.9607, 0.9607, 0.9172)$. The refinement ψ_2 is weaker than ψ_1 . Such insight into their weakness could be used to guide the refinement approach (e.g., [AMT13, CA17]) 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 permissive formulae. Consider the AMBA-AHB protocol with eight masters and the two alternative refinements: $\theta_1 = \mathbf{GF}(h\text{master}_0 \vee \neg h\text{busreq}_1)$; and $\theta_2 = \mathbf{GF}(h\text{master}_1 \vee \neg h\text{busreq}_2)$. Clearly the two alternatives express the same kind of constraint on different masters. Since the two masters do not have priorities over each other, the two refinements have the same weakness as expected: $d(\theta^\varepsilon \wedge \theta_1) = d(\theta^\varepsilon \wedge \theta_2) = (0.9396, 0.9396, 0.9214)$.

8.1.2. Heuristic search of GR(1) assumption refinements

To the best of our knowledge, assumption refinement is approached in most cases as a breadth-first search over a refinement tree [AMT13, LDS11, CA17, MRS19]. Given an initial, unrealizable specification $\phi = \phi^\varepsilon \rightarrow \phi^S$, a counterstrategy $\mathcal{C}$ is computed (see Sect. 2) and a set of alternative assumptions $\{\psi_1, \dots, \psi_k\}$ inconsistent with $\mathcal{C}$ is generated.

Fig. 16. Refinement tree. The gray nodes are the leaves: ψ_{21} and ψ_{23} are solutions to the refinement problem.

For each ψ_i in turn, the new specification $\phi^\mathcal{E} \wedge \psi_i \rightarrow \phi^\mathcal{S}$ is checked for realizability and in case it is not realizable, a further set of additional assumptions $\{\psi_{i1}, \dots, \psi_{ik'}\}$ is produced to be added to $\phi^\mathcal{E} \wedge \psi_i$. The expansion of one branch of the tree stops either when a solution is reached, or when a refinement made of inconsistent assumptions is generated. An example of a refinement tree is shown in Fig. 16.

We modify this approach and use our weakness measure to sort the nodes that are expanded by decreasing weakness; therefore, if $\phi^\mathcal{E} \wedge \psi_2 \wedge \psi_{22}$ is weaker than $\phi^\mathcal{E} \wedge \psi_1$, the former is expanded before the latter. However, our weakness measure leads the search algorithm to expand refinements consisting only of fairness conditions earlier than any other formula. This happens because fairness conditions typically affect only d_3 , which has the lowest priority in the order relation. Therefore, a heuristic search only based on the weakness measure may not converge to any solution in reasonable time, if there exists no solution only formed by fairness conditions. To circumvent this problem, we impose a limit of 8 fairness conditions accepted in any refinement; our approach does not expand further a node of the tree containing more than 8 fairness conditions. This choice is driven by the preliminary observation of the solutions of BFS reported in [CA17]. This approach found one solution with 7 fairness conditions: our choice is such that this solution is not excluded *a priori* from the search.

We run the heuristic search and compare it with the BFS search in the setting of [CA16]. The two approaches are run for approximately 4 h (14,700 s) each on six benchmark case studies, the three AMBA-AHB cases described in the previous subsection, and three case studies first presented in [KMR17]: *ColorSort*, a robot sorting Lego bricks by color, *GyroAspect*, the specification of a robot with self-balancing capabilities, and *Humanoid*, a robot with humanoid shape. The original AMBA-AHB specification [BJP⁺12] is realizable: in order to obtain an unrealizable specification, we remove the fairness condition **GF***hready*, as done in [AMT13].

Figure 17 shows the solutions found by BFS and heuristic search sorted by decreasing weakness. We observe that in every case study except *ColorSort* the median ranking of BFS solutions is lower than the median ranking of heuristic search: this shows that typically the heuristic approach finds solutions that are weaker than more than half the solutions of BFS. Additionally, in all case studies except *ColorSort*, the heuristic search finds either a solution as weak as the weakest one found by BFS, or an even weaker solution which was missed by BFS (AMBA08). Notice also that while BFS generates several inconsistent solutions (rightmost point of the plots, labeled FALSE), the heuristic search never produces such a node. This observation corresponds with the intuition that if at every step the weakest available refinement is picked, the risk of overconstraining the assumptions and producing inconsistent GR(1) conjunctions as a solution is reduced.

The heuristic search works as a conservative search, that attempts to reach a solution by strengthening the current assumptions as little as possible. However, since this may delay reaching a sufficiently constraining set of assumptions to make the specification realizable, fewer solutions are expected to be found in total. However, the reduction in number is not dramatic, and in two case studies (AMBA02 and *GyroAspect*) the heuristic search even finds more solutions than BFS.

The exception of *ColorSort* is due to the randomness involved in the node expansion procedure. As an intractable number of GR(1) assumptions may be inconsistent with a counterstrategy, randomized assumption generation procedures are employed to generate a subset of these assumptions [CA16]; in this randomization, the weakest refinements found by BFS are missed when running the heuristic search.

Fig. 17. Results for BFS and heuristic search. The results of the two approaches are ranked together by decreasing weakness. The x-axis shows the positions of the solutions in the ranking, from the weakest (leftmost) to the strongest (rightmost), with FALSE being the strongest. The y-axis is the number of solutions in each position of the ranking. The filled circles on the x-axis represent the median rankings of BFS and heuristic search solutions.

Performance. We aim to assess the discriminative power and time requirements of the weakness measure compared with implication. To this end we perform an experiment on the refinement trees generated by the heuristic search: each pair of formulae in the refinement trees is compared both via implication and via the weakness measure from Definition 6.31. 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) [Lut14] of the formulae $\phi_1 \wedge \neg \phi_2$ and $\phi_2 \wedge \neg \phi_1$, and checking whether any of them is empty [RDLKP13]. In our implementation, the TGBAs are computed via the same SPOT tool used to compute weakness.

Table 2 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 ordered strictly via implication (**%Impl/#Pairs**) and via weakness (**%Weak/#Pairs**), the total time to perform all the comparisons via implication (**Time impl**) and via weakness (**Time weakness**).

To assess the discriminative power of the two approaches, 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 that can be discriminated via logical implication (either because one strictly implies the other or because the two formulae are logically equivalent).

Table 2. Comparison between the discriminative power of implication and weakness on the refinement trees of the heuristic search

Case study	#Nodes (k)	#Pairs	%Impl/#Pairs	%Weak/#Pairs	Time impl	Time weakness
AMBA02	318	50,403	35.80%	97.80%	46,535 s	695 s
AMBA04	58	1,653	36.78%	97.64%	1,350.241 s	145.67 s
AMBA08	40	780	32.95%	93.46%	31,735.33 s	20,197.20 s

The table shows that, despite weakness does not capture implication strictly in all cases, it still allows for the discrimination of a larger set of assumptions, by virtue of Axiom 4.3.

Regarding the computation time, the following general consideration holds. With implication it is necessary to produce two automata, one for $L(\phi_1) \cap L(\neg \phi_2)$ and one for $L(\phi_2) \cap L(\neg \phi_1)$, and then run an emptiness check on each of them. When comparing k formulae, this operation must be repeated for $O(k^2)$ pairs of formulae. On the other hand, for a set of formulae containing at most m fairness conditions and at most $|\mathcal{S}_H|$ accepting SCCs with maximum entropy (where $|\mathcal{S}_H|$ is the order of strong connectedness, see Definition 6.28), our weakness measure requires $1 + m * |\mathcal{S}_H|$ DBA/DMA computations, yielding $O(|\mathcal{S}_H|mk)$ automata needed for comparing k formulae. In general, in a refinement tree k is larger than m and $|\mathcal{S}_H|$: in the AMBA case studies, the values of k reported in the table are all significantly higher than m (which has been fixed to 8 by design) and the order (which is observed to be 1 in all the refinements explored). In this respect, the advantage of our weakness measure resides in the reduced number of DBA/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 [EKS16]. The effect of the complexity can be observed in particular in AMBA08. Since this case study contains several variables, the DBAs of its refinements have a large state space, and require much more time to be computed than the TGBAs used for implication (up to 40 times longer). Thereby, even if the total number of automata computed for the weakness measure is significantly lower than the number of automata for implication, the gain in total time achieved by using the weakness measure is not as significant as in AMBA02 and AMBA04.

Correctness of implementation. We use the data from the comparison between implication and weakness to assess the correctness of our implementation. We simply observe that in no case the weakness measure and the implication checks induce opposite orderings on the pairs of formulae. Despite it being possible that both implementations are defective, the probability of this being the case is lower than just the weakness computation being defective.

8.2. Hausdorff dimension for quantitative model checking

In this section we provide an application of computing Hausdorff dimension on fairness complements in a quantitative model checking example from [ABD14a].

In the cited work, the goal is to identify an alternative measure to probability in order to quantify the infinite runs of an automaton $\mathcal{B}$ that violate a formula ϕ . The reason adduced by the authors is that under any typical stochastic model used in model checking (such as Markov chains) the probability of an infinite behaviour to satisfy interesting properties like invariants is either 0 or 1. The work proposes to use the entropy $H(L(\mathcal{B}) \cap L(\neg \phi))$ instead of probability measures, and shows that this measure captures the “likelihood” of reaching a violating state when dealing with safety properties (that correspond to invariants in our context). However it points out the limitation of this measure with fairness conditions, which cannot be discriminated by entropy alone (see Sect. 6.3). In particular, if the automaton is strongly connected, one has $H(L(\mathcal{B}) \cap L(\neg \phi)) = H(L(\mathcal{B}))$ if some runs of $\mathcal{B}$ do not satisfy ϕ . Hausdorff dimension, instead, can capture the difference between the two languages.

We apply it to the Dining Philosophers example with three philosophers from [ABD14a]. The model consists of three philosophers and three forks; the philosophers alternately think and eat, but before eating each of the philosophers must grab the fork to his left and the one to his right; if one of the two is taken by another philosopher,

the philosopher needs to wait until the fork becomes available; finally, once the philosopher has eaten, he releases his forks for use by everyone.

A deadlock state is any state in which all three philosophers have taken their left fork and no progress can be made from there. Let ϕ^{DP} be the GR(1) formula describing all the philosophers' behaviors that do not reach the deadlock state. This formula uses the three model variables $state1$, $state2$ and $state3$ describing the state of each philosopher, which is any of the values *THINK*, *LEFTFORK* and *EAT*. It can be expressed in GR(1) as $\phi^{DP} = \mathbf{G}(\neg (state1 = \text{LEFTFORK} \wedge state2 = \text{LEFTFORK} \wedge state3 = \text{LEFTFORK}) \wedge B_T)$, where ϕ^T expresses the transitions allowed to each philosophers: $B_T = (state1 = \text{THINK} \rightarrow \mathbf{X}(state1 = \text{THINK} \vee state1 = \text{LEFTFORK}) \wedge \dots)$.

The goal is to assign a measure to the subset of these behaviors such that none of the philosophers starves. This condition is expressed by the fairness formula $\phi_{fair}^{DP} = \mathbf{GF}(state1 = \text{EAT}) \wedge \mathbf{GF}(state2 = \text{EAT}) \wedge \mathbf{GF}(state3 = \text{EAT})$. When using entropy only, $H(L(\phi^{DP})) = H(L(\phi^{DP} \wedge \phi_{fair}^{DP})) = 0.0718$. When using our weakness measure from Definition 6.18, the second component distinguishes the presence of the fairness condition. We use this simplified definition since the invariant ϕ^{DP} is strongly connected. The result is $d_3(\phi^{DP} \wedge \phi_{fair}^{DP}) = \dim(L(\phi^{DP}) \setminus L(\phi_{fair}^{DP})) = 0.0479 \neq d_3(\phi^{DP}) = 0$. Therefore, contrary to the work in [ABD14a], our measure is able to capture the difference in the permissiveness of the two formulae.

9. Conclusion

In this paper, we proposed a new measure for assessing the weakness of GR(1) conjunctions quantitatively. The weakness measure represents a rigorously defined quantity for ordering sets of GR(1) conjunctions according to their permissiveness with respect to their models. The measure is based on two quantities, entropy and Hausdorff dimension, whose induced order relation is consistent with the notion of implication between formulae. These two quantities are used to define a three-component measure consisting of: the entropy of the formula's language, its Hausdorff dimension, and the Hausdorff dimension of the fairness complement. The third component overcomes the limitation entropy and Hausdorff dimension present when quantifying the behavior of fairness conditions.

There remain two limitations of this approach, which are left for future work: one theoretical, related to the satisfaction of Axiom 4.3 when entropies are equal, and one relating with the tractability of computing the weakness measure. The former may be tackled through measures that take into account the eigenvalues of SCCs that are discarded in the computation of entropy and Hausdorff dimension as of Sect. 5.1. As for the latter, the major bottleneck of computing the measure is the need for deterministic automata to be produced, which induces high computation time because of the determinization process [EKS16]. Future work should be conducted to investigate whether Hausdorff dimension can be computed, or approximated at a controlled accuracy, on nondeterministic ω -automata. In the average case, these automata are cheaper to compute than their deterministic counterpart. We intend to investigate this as part of our future work. As a further direction for 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 [MS94].

Acknowledgements

We would like to thank the anonymous reviewers for their thoughtful comments which helped improve the presentation of the paper. This work was supported in part by the EPSRC HiPEDS CDT (EP/L016796/1) and the Imperial College Elsie Widdowson Fellowship.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds

the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

References

- [AAK13] Almagor S, Avni G, Kupferman O (2013) Automatic generation of quality specifications. In: International conference on computer aided verification (CAV). Springer, Berlin, pp 479–494
- [ABD14a] Asarin E, Blockelet M, Degorre A (2014) Entropy model checking. In: Workshop on Quantitative Aspects of Programming Languages (QAPL)—joint with european joint conference on theory and practice of software (ETAPS)
- [ABD⁺14b] Asarin E, Blockelet M, Degorre A, Dima C, Mu C (2014) Asymptotic behaviour in temporal logic. In: Joint meeting of the annual conference on computer science logic and the annual symposium on logic in computer science (CSL/LICS). ACM Press, pp 1–9
- [ADG16] Albarghouthi A, Dillig I, Gurfinkel A (2016) Maximal specification synthesis. ACM SIGPLAN Notices 51(1):789–801
- [AMT13] Alur R, Moarref S, Topcu U (2013) Counter-strategy guided refinement of GR(1) temporal logic specifications. In: International conference on formal methods in computer-aided design (FMCAD). Springer, Berlin, pp 26–33
- [AMT15] Alur R, Moarref S, Topcu U (2015) Pattern-based refinement of assume-guarantee specifications in reactive synthesis. In: International conference on tools and algorithms for the construction and analysis of systems (TACAS). Springer, Berlin, pp 501–516
- [BCG⁺10] Bloem R, Cimatti A, Greimel K, Hofferek G, Könighofer R, Roveri M, Schuppan V, Seeber R (2010) RATSY: a new requirements analysis tool with synthesis. In: International conference on computer aided verification (CAV). Springer, Berlin, pp 425–429
- [BCG⁺14] Bloem R, Chatterjee K, Greimel K, Henzinger TA, Hofferek G, Jobstmann B, Könighofer B, Könighofer R (2014) Synthesizing robust systems. Acta Inf 51(3–4):193–220
- [BCHJ09] Bloem R, Chatterjee K, Henzinger TA, Jobstmann B (2009) Better quality in synthesis through quantitative objectives. In: International conference on computer aided verification (CAV). Springer, Berlin, pp 140–156
- [BDP⁺13] Braberman V, D'Ippolito N, Piterman N, Sykes D, Uchitel S (2013) Controller synthesis: from modelling to enactment. In: International Conference on Software Engineering (ICSE), pp 1347–1350. IEEE
- [BJP⁺12] Bloem R, Jobstmann B, Piterman N, Pnueli A, Sa'Ar Y (2012) Synthesis of reactive(1) designs. J Comput Syst Sci 78(3):911–938
- [BKvS12] Babiak T, Křetínský M, Řehák V, Strejček J (2012) LTL to Büchi automata translation: fast and more deterministic. In: International conference and tools and algorithms for the construction and analysis of systems (TACAS). Springer, Berlin, pp 95–109
- [BP94] Berman A, Plemmons R (1994) Nonnegative matrices in the mathematical sciences. Society for Industrial and Applied Mathematics, Philadelphia
- [CA16] Cavezza DG, Alrajeh D (2016) Interpolation-based GR(1) assumptions refinement. CoRR, [arXiv:1611.07803](https://arxiv.org/abs/1611.07803)
- [CA17] Cavezza DG, Alrajeh D (2017) Interpolation-based GR(1) assumptions refinement. In: International conference on tools and algorithms for the construction and analysis of systems (TACAS). Springer, Berlin, pp 281–297
- [CAG18] Cavezza DG, Alrajeh D, György A (2018) A weakness measure for GR(1) formulae. In: International symposium on formal methods (FM), pp 110–128
- [CDF⁺06] Chatterjee K, De Alfaro L, Faella M, Henzinger TA, Majumdar R, Stoelinga M (2006) Compositional quantitative reasoning. In: International conference on the quantitative evaluation of systems (QEST). Springer, Berlin, pp 179–188
- [CGP03] Cobleigh JM, Giannakopoulou D, Păsăreanu CS (2003) Learning assumptions for compositional verification. In: International conference on tools and algorithms for the construction and analysis of systems (TACAS). Springer, Berlin, pp 331–346
- [CHJ08] Chatterjee K, Henzinger TA, Jobstmann B (2008) Environment assumptions for synthesis. In: International conference on concurrency theory (CONCUR). Springer, Berlin, pp 147–161
- [CJS09] Calude CS, Jürgensen H, Staiger L (2009) Topology on words. Theor Comput Sci 410(24–25):2323–2335
- [CL08] Cassandras CG, Lafortune S (2008) Introduction to discrete event systems. Springer, Berlin
- [CRST08] Cimatti A, Roveri M, Schuppan V, Tchaltsev A (2008) Diagnostic information for realizability. In: International conference on verification, model checking, and abstract interpretation (VMCAI). Springer, Berlin, pp 52–67
- [CT06] Cover TM, Thomas JA (2006) Elements of information theory, 2nd ed. Wiley, New York
- [DAC99] Dwyer MB, Avrunin GS, Corbett JC (1999) Patterns in property specifications for finite-state verification. In: International conference on software engineering (ICSE). ACM, pp 411–420
- [DAV] <https://gitlab.doc.ic.ac.uk/dgc14/FMextRepo>.
- [DBK⁺14] D'Ippolito N, Braberman V, Kramer J, Magee J, Sykes D, Uchitel S (2014) Hope for the best, prepare for the worst: multi-tier control for adaptive systems. In: International conference on software engineering (ICSE), pp 688–699
- [DBPU10] D'Ippolito NR, Braberman V, Piterman N, Uchitel S (2010) Synthesis of live behaviour models. In: International symposium on foundations of software engineering (FSE). ACM Press, p 77
- [DBSU15] D'Ippolito N, Braberman V, Sykes D, Uchitel S (2015) Robust degradation and enhancement of robot mission behaviour in unpredictable environments. In: International workshop on control theory for software engineering (CTSE). ACM, pp 26–33
- [DCA⁺18] Degiovanni R, Castro P, Arroyo M, Ruiz M, Aguirre N, Frias M (2018) Goal-conflict likelihood assessment based on model counting. In: International conference on software engineering (ICSE). ACM Press, New York, pp 1125–1135
- [DGT18] Dimitrova R, Ghasemi M, Topcu U (2018) Maximum realizability for linear temporal logic specifications. In: International symposium on automated technology for verification and analysis (ATVA). Springer, Berlin, pp 458–475

- [DLLF⁺16] Duret-Lutz A, Lewkowicz A, Fauchille A, Michaud T, Renault E, Xu L (2016) Spot 2.0—a framework for LTL and ω -automata manipulation. In: International symposium on automated technology for verification and analysis (ATVA). Springer, Berlin, pp 122–129
- [EKS16] Esparza J, Ketinský J, Sickert S (2016) From LTL to deterministic automata: a safrless compositional approach. *Formal Methods Syst Des* 49(3):219–271
- [Fal04] Falconer K (2004) *Fractal geometry: mathematical foundations and applications*. Wiley, London
- [GL02] Giannakopoulou D, Lerda F (2002) From states to transitions: improving translation of LTL formulae to Büchi automata. In: International conference on formal techniques for networked and distributed systems (FORTE), pp 308–326
- [GM03] Giannakopoulou D, Magee J (2003) Fluent model checking for event-based systems. In: European software engineering conference held jointly with international symposium on foundations of software engineering (ESEC/FSE). ACM, pp 257–266
- [GTW02] Grädel E, Thomas W, Wilke T (2002) *Automata logics, and infinite games: a guide to current research*. Springer, New York
- [Hen10] Henzinger T (2010) From boolean to quantitative notions of correctness. *ACM SIGPLAN Notices* 45(1):157
- [HJ86] Horn RA, Johnson CR (1986) *Matrix analysis*. Cambridge University Press, New York
- [HJ94] Hansson H, Jonsson B (1994) A logic for reasoning about time and reliability. *Formal Asp Comput* 6(5):512–535
- [HJ12] Horn RA, Johnson CR (2012) *Matrix analysis*
- [HO13] Henzinger TA, Otop J (2013) From model checking to model measuring. In: International conference on concurrency theory (CONCUR). Springer, Berlin, pp 273–287
- [KE12] Kretinský J, Esparza J (2012) Deterministic automata for the (F,G)-fragment of LTL. In: International conference on computer aided verification (CAV). Springer, Berlin, pp 7–22
- [KGF09] Kress-Gazit H, Fainekos GE, Pappas GJ (2009) Temporal-logic-based reactive mission and motion planning. *IEEE Trans Robot* 25(6):1370–1381
- [KHB09] Konighofer R, Hofferek G, Bloem R (2009) Debugging formal specifications using simple counterstrategies. In: International conference on formal methods in computer-aided design (FMCAD), pp 152–159
- [KMR17] Kuvent A, Maoz S, Ringert JO (2017) A symbolic justice violations transition system for unrealizable GR(1) specifications. In: European software engineering conference held jointly with international symposium on foundations of software engineering (ESEC/FSE), number 1. ACM Press, New York, pp 362–372
- [Kup12] Kupferman O (2012) Recent challenges and ideas in temporal synthesis. In: International conference on current trends in theory and practice of computer science (SOFSEM). Springer, Berlin, pp 88–98
- [Kwi07] Kwiatkowska M (2007) Quantitative verification: models, techniques and tools. In: European software engineering conference held jointly with international symposium on foundations of software engineering (ESEC/FSE). ACM Press, p 449
- [LDS11] Li W, Dworkin L, Seshia SA (2011) Mining assumptions for synthesis. In: International conference on formal methods and models for codesign (MEMOCODE). ACM/IEEE, pp 43–50
- [LSWW10] Lomuscio A, Strulo B, Walker N, Wu P (2010) Assume-guarantee reasoning with local specifications. In: International conference on formal engineering methods (ICFEM), pp 204–219
- [Lut14] Lutz AD (2014) LTL translation improvements in spot 1.0. *Int J Crit Comput Based Syst* 5(1/2):31
- [MR15] Maoz S, Ringert JO (2015) GR(1) synthesis for LTL specification patterns. In: European software engineering conference held jointly with international symposium on foundations of software engineering (ESEC/FSE), number 1. ACM Press, New York, pp 96–106
- [MRS19] Maoz S, Ringert JO, Shalom R (2019) Symbolic repairs for GR(1) specifications. In: Proceedings of the international conference on software engineering (ICSE)
- [MS94] Merzenich W, Staiger L (1994) Fractals, dimension, and formal languages. *Inf Théor Appl* 28(3–4):361–386
- [NA06] Nam W, Alur R (2006) Learning-based symbolic assume-guarantee reasoning with automatic decomposition. In: International symposium on automated technology for verification and analysis (ATVA), pp 170–185
- [Pnu77] Pnueli A (1977) The temporal logic of programs. In: Annual symposium on foundations of computer science, pp 46–57
- [PPS06] Piterman N, Pnueli A, Sa’ar Y (2006) Synthesis of reactive(1) designs. In: International conference on verification, model checking, and abstract interpretation (VMCAI). Springer, Berlin, pp 364–380
- [PR89] Pnueli A, Rosner R (1989) On the synthesis of a reactive module. In: Principles of programming languages (POPL). ACM, New York, pp 179–190
- [RDLKP13] Renault E, Duret-Lutz A, Kordon F, Poitrenaud D (2013) Three SCC-based emptiness checks for generalized Büchi automata. In: International conference on logic for programming artificial intelligence and reasoning (LPAR). Springer, Berlin, pp 668–682
- [SB00] Somenzi F, Bloem R (2000) Efficient Buchi automata from LTL formulae. In: International conference on computer aided verification (CAV), vol 1855, pp 1–17
- [Sen06] Seneta E (2006) *Non-negative matrices and Markov chains*. Springer, Berlin
- [Ses15] Seshia SA (2015) Combining induction, deduction, and structure for verification and synthesis. *Proc IEEE* 103(11):2036–2051
- [Sta98] Staiger L (1998) The Hausdorff measure of regular ω -languages is computable. Technical Report August, Martin-Luther-Universität
- [Sta15] Staiger L (2015) On the Hausdorff measure of regular omega-languages in Cantor space. *Discrete Math Theor Comput Sci* 17(1):357–368
- [Tar72] Tarjan R (1972) Depth-first search and linear graph algorithms. *SIAM J Comput* 1(2):146–160
- [TN16] Tabuada P, Neider D (2016) Robust linear temporal logic. In: Annual conference on computer science logic (CSL), pp 10:1–10:21. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik
- [TUS⁺17] Tomita T, Ueno A, Shimakawa M, Hagiwara S, Yonezaki N (2017) Safrless LTL synthesis considering maximal realizability. *Acta Inf* 54(7):655–692
- [Var96] Vardi MY (1996) An automata-theoretic approach to linear temporal logic. *Logics for concurrency*, pp 238–266

Received 20 July 2019

Accepted in revised form 16 September 2020 by Erik de Vink, Ana Cavalcanti, Jan Peleska and Bill Roscoe

Published online 2 November 2020