Robust Linear Temporal Logic*

Paulo Tabuada[1] and Daniel Neider[2]

1 **Department of Electrical Engineering, University of California at Los Angeles, Los Angeles, CA 90095, USA**
tabuada@ucla.edu

2 **Department of Electrical Engineering, University of California at Los Angeles, Los Angeles, CA 90095, USA**
neider@automata.rwth-aachen.de

Abstract

Although it is widely accepted that every system should be robust, in the sense that "small" violations of environment assumptions should lead to "small" violations of system guarantees, it is less clear how to make this intuitive notion of robustness mathematically precise. In this paper, we address the problem of how to specify robustness in temporal logic. Our solution consists of a robust version of the Linear Temporal Logic (LTL) fragment that only contains the always and eventually temporal operators. We denote this new logic by rLTL($\boxdot, \diamondsuit$). Its formulas are syntactically identical to LTL formulas but are endowed with a many-valued semantics that encodes robustness. In particular, the semantics of the rLTL formula $\varphi \Rightarrow \psi$ is such that a "small" violation of the environment assumption φ is guaranteed to only produce a "small" violation of the system guarantee ψ. In addition, we study the verification and synthesis problems for this logic. Similarly to LTL, we show that: both problems are decidable; the verification problem can be solved in exponential time; the synthesis problem is solvable in doubly exponential time. All the results for rLTL($\boxdot, \diamondsuit$) smoothly extend to full rLTL, the robust version of full LTL. For reasons of space, such extension is not discussed but available in an extended version [21].

1998 ACM Subject Classification F.4.1 Mathematical Logic, F.1.1 Models of Computation

Keywords and phrases Linear Temporal Logic, Robustness

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.10

1 Introduction

Specifications for open reactive systems are typically written as an implication

$$\varphi \Rightarrow \psi, \tag{1}$$

where φ is an environment assumption and ψ is a system guarantee. In Linear Temporal Logic (LTL), this implication is equivalent to $\neg\varphi \vee \psi$. Hence, whenever the assumption φ is violated the system can behave arbitrarily. This is clearly inadequate since environment assumptions will *inevitably* be violated: the real environment where the system will be deployed is not completely known at design time and thus cannot be accurately described by the formula φ.

We argue that a robust design satisfies the implication in (1) in a robust manner (i.e., a "small" violation of φ results, at most, in a "small" violation of ψ). To make this intuitive

* This research was partially supported by the NSF project ExCAPE: Expeditions in Computer Augmented Program Engineering (CCF-1138996).

notion of *robustness* mathematically precise, we introduce a fragment of a new logic termed *robust Linear Temporal Logic* and simply denoted by rLTL. For reasons of space, we only discuss the fragment of rLTL that contains the always and eventually temporal operators, denoted by rLTL($\boxdot, \diamondsuit$). However, all the decidability and complexity results extend to full rLTL as discussed in [21]. In developing rLTL, we were guided by two objectives: 1) the syntax of rLTL should be similar to the syntax of LTL in order to make the transition from LTL to rLTL as transparent as possible; 2) robustness should be intrinsic to the logic rather than extrinsic (i.e., robustness should not rely on the ability of the designer to provide quantitative information such as ranks, costs, or quantitative interpretations of atomic propositions). This guarantees that verification and synthesis techniques for rLTL are widely applicable as they *only* require an LTL specification.

The main conceptual question to be addressed when developing the semantics of rLTL is how to give mathematical meaning to "small" violations of a formula φ. The approach advocated in this paper can be intuitively explained by regarding LTL formulas of the form $\Box p$, $\diamondsuit\Box p$, $\Box\diamondsuit p$, and $\diamondsuit p$, for an atomic proposition p, as requirements on the number of times that p is satisfied over time. Under this interpretation, and for the formula $\varphi = \Box p$, there is a clear ordering among the possible temporal evolutions of p: p being satisfied at every time instant is preferred to p being violated at finitely many time instants which, in turn, is preferred to p being satisfied and violated at infinitely many time instants. The latter case is preferred to p only being satisfied at finitely many time instants and this case is preferred over p being satisfied at no time instant. A semantics that would distinguish between these different five cases would then enable us to state that violating $\Box p$ while satisfying $\diamondsuit\Box p$ consists of a smaller violation of the formula $\varphi = \Box p$ than violating $\Box p$ while satisfying $\Box\diamondsuit p$. Making these ideas mathematically rigorous requires a 5-valued semantics that we develop in this paper. Interestingly, our specific interpretation of the five different truth values leads to an intuitionistic semantics where negation is dualized and to a corresponding algebraic structure, *da Costa algebras*, which were only very recently investigated [18].

Contributions

The first contribution of this paper (Section 3) is the fragment rLTL($\boxdot, \diamondsuit$) of our new logic rLTL, which enables reasoning about robustness of LTL specifications. The syntax of rLTL($\boxdot, \diamondsuit$) is almost identical to the syntax of LTL with the only notable difference being dotted temporal operators. The 5-valued semantics of rLTL($\boxdot, \diamondsuit$) is, however, different in many regards (e.g., it precludes several LTL identities to hold on rLTL). For that reason, we carefully motivate the need for a many-valued semantics and provide several examples illustrating how rLTL can be used to reason about robustness.

The second contribution (Section 4) is the study of several computational questions related to rLTL($\boxdot, \diamondsuit$). We first show that rLTL($\boxdot, \diamondsuit$) is as expressive as the LTL($\Box, \diamondsuit$) fragment by providing effective translations between both logics. However, the translation from rLTL($\boxdot, \diamondsuit$) to LTL($\Box, \diamondsuit$) involves an exponential blow-up, leaving open the possibility of improved complexity bounds for the rLTL($\boxdot, \diamondsuit$) verification and synthesis problems. Indeed, the exponential blow-up can be avoided by a carefully generalization of the construction that associates with each LTL formula φ a Büchi automaton $\mathcal{A}_\varphi$ recognizing all infinite words satisfying φ. Critical to this new construction are the properties of the da Costa algebra, used to define the rLTL($\boxdot, \diamondsuit$) semantics, which can be leveraged to keep the size of $\mathcal{A}_\varphi$ in $\mathcal{O}(|\mathrm{cl}(\varphi)| \cdot 5^{|\mathrm{cl}(\varphi)|})$ where $\mathrm{cl}(\varphi)$ denotes the set of subformulas of φ. Note that this is the same complexity bound as for LTL where we replace 2 (LTL has a 2-valued semantics) with

5 (rLTL($\boxdot, \diamondsuit$) has a 5-valued semantics). Additional consequences of the construction of $\mathcal{A}_\varphi$ include: 1) rLTL($\boxdot, \diamondsuit$) specifications can be verified in time exponential in the size of specification and polynomial in the size of the system being verified; 2) the time complexity of synthesizing reactive controllers for rLTL($\boxdot, \diamondsuit$) specifications is doubly exponential in the size of the specification and polynomial in the size of the underlying game graph.

Finally, we show in Section 5 that by dualizing the semantics of rLTL in a specific sense one obtains a logic that is adequate to reason about quality.

Related efforts

Several efforts to robustify the implication in (1) have been reported in the literature. Bloem et al. [5] formalized robustness by comparing how often the system violates its assumptions with how often the environment violates its assumptions. Counting the number of violations requires the designer to provide quantitative information in the form of error functions. In contrast, when working with rLTL, the designer only needs to provide an LTL specification. A different notion of robustness appeared in [8] and requires the effect of a sporadic disturbance to disappear in finite time. The semantics of rLTL was built so as to naturally encode this as well as other requirements expressing how a weakening of the system assumptions should lead to a weakening of the system guarantees. Previous work by one of the authors, reported in [22], provided a single notion of robustness encompassing the notion in [8] but requiring the designer to provide quantitative information in the form of a cost. Such cost implicitly specifies how guarantees and assumptions are to be weakened in a robust design and was inspired by the work of Alur et al. [2]. A different formalization of robustness appeared in [9], which considered a specific class of violations of safety assumptions defined by the frequency of violations. In contrast to all the previously described approaches, the results in this paper do not require any additional assumptions or input from a designer beyond an LTL formula.

Robustness for liveness specifications was discussed by Bloem et al. [4], who considered specifications of the form $\wedge_{i\in I}\varphi_i \Rightarrow \wedge_{j\in J}\psi_j$, where φ_i and ψ_j are formulas of the form $\diamondsuit\square p$ for some atomic proposition p (depending on i and j). Robustness is then measured by comparing the number of violated environment assumptions φ_i with the number of violated guarantees ψ_j. This approach is incomparable with ours since the rLTL($\boxdot, \diamondsuit$) semantics does not distinguish between the violation of one or multiple assumptions. It does, however, distinguish between the different ways in which φ_i and ψ_j can be violated. Although robustness is formalized differently, rLTL($\boxdot, \diamondsuit$) can be used to reason about the robustness of both safety and liveness specifications. Also incomparable with rLTL($\boxdot, \diamondsuit$) is the work by Chaudhuri et al. [6] and by Majumdar et al. [14], which considers continuity properties of software expressed by the requirement that a deviation in a program's input causes a proportional deviation in its output. Although natural, these notions of robustness only apply to the Turing model of computation and not to the reactive model employed in this paper. Robustness was also investigated in the context of systems biology (e.g., Rizk et al. [19] and Češka et al. [23]) although the considered models are quite different as they require continuity and stochasticity.

There exists a large body of work on many-valued logics that we will not attempt to review here since it does not directly address questions of robustness. We do, however, allow for one exception: the work by Fainekos and Pappas [10] on robustness of temporal logic over continuous signals and its extensions (e.g., by Donzé and Maler [7]). These results, however, require continuous-valued signals whereas rLTL is to be used in the classical setting of discrete-time and finite valued signals (e.g., as in transition systems).

The last body of work related to the contents of this paper is the work on lattice automata and lattice LTL [13, 1]. The syntax of lattice LTL is similar to the syntax of LTL except that

atomic propositions assume values on a finite lattice. Lattice LTL derives its many-valued character from the atomic propositions, whereas atomic propositions in rLTL($\boxdot, \diamondsuit$) are interpreted classically (i.e., they only assume two truth values). Therefore, the many-valued character of rLTL($\boxdot, \diamondsuit$) arises from the temporal evolution of the atomic propositions and not from the nature of the atomic propositions or their interpretation. In fact, if we only allow two truth values for the atomic propositions in lattice LTL, this logic specializes to LTL. Hence, these two logics capture orthogonal considerations, and results on lattice LTL and lattice automata do not shed light on how to address similar problems for rLTL($\boxdot, \diamondsuit$).

2 Notations and Review of LTL

Let $\mathbb{N} = \{0, 1, \ldots\}$ be the set of natural numbers and $\mathbb{B} = \{0, 1\}$ be the set of Boolean values (0 interpreted as *false* and 1 as *true*). For a set S, let 2^S be the *powerset* of S and S^ω be the set of all *infinite sequences* of elements of S. An *alphabet*, usually denoted by Σ, is a finite, nonempty set whose elements are called *symbols*. An infinite sequence $\sigma = a_0 a_1 \ldots$ of symbols with $a_i \in \Sigma$, $i \in \mathbb{N}$, is called an *infinite word*. For an infinite word $\sigma = a_0 a_1 \ldots \in \Sigma^\omega$ and $i \in \mathbb{N}$, let $\sigma(i) = a_i$ denote the i-th symbol of σ and $\sigma_{i..}$ the (infinite) suffix of σ starting at position i (i.e., $\sigma_{i..} = \sigma_i \sigma_{i+1} \ldots \in \Sigma^\omega$). In particular, we have the equality $\sigma_{0..} = \sigma$.

Next, we recapitulate the syntax and semantics of a fragment of *Linear Temporal Logic (LTL)*, abbreviated as LTL($\Box, \diamondsuit$), that is restricted to the always and eventually modalities.

▶ **Definition 1** (LTL($\Box, \diamondsuit$) syntax). Let $\mathcal{P}$ be a nonempty, finite set of atomic propositions. *LTL*($\Box, \diamondsuit$) *formulas* are inductively defined as follows: (a) each $p \in \mathcal{P}$ is an LTL formula and (b) if φ and ψ are LTL formulas, so are $\neg\varphi$, $\varphi \vee \psi$, $\Box\varphi$, and $\diamondsuit\varphi$.

We also allow the formulas *true*, *false*, $\varphi \wedge \psi$, and $\varphi \Rightarrow \psi$ with their usual meaning.

In the following, we define the semantics of LTL($\Box, \diamondsuit$) by a mapping W that maps an infinite word $\sigma \in \Sigma^\omega$, $\Sigma = 2^{\mathcal{P}}$, and an LTL($\Box, \diamondsuit$) formula φ to the element $W(\sigma, \varphi) \in \mathbb{B}$. Although the semantics of LTL is usually defined by a satisfaction relation, we here define it in terms of a function so as to be consistent with our definition of the rLTL($\boxdot, \diamondsuit$) semantics.

▶ **Definition 2** (LTL($\Box, \diamondsuit$) semantics). The *LTL*($\Box, \diamondsuit$) *semantics* is a mapping W, called *valuation*, defined as follows:

$$W(\sigma, p) = \begin{cases} 0 & p \notin \sigma(0); \text{ and} \\ 1 & p \in \sigma(0) \end{cases} \quad (2)$$

$$W(\sigma, \varphi \vee \psi) = \max\{W(\sigma, \varphi), W(\sigma, \psi)\} \quad (3)$$

$$W(\sigma, \neg\varphi) = 1 - W(\sigma, \varphi) \quad (4)$$

$$W(\sigma, \Box\varphi) = \inf_{i \geq 0} W(\sigma_{i..}, \varphi) \quad (5)$$

$$W(\sigma, \diamondsuit\varphi) = \sup_{i \geq 0} W(\sigma_{i..}, \varphi) \quad (6)$$

We often use a compact notation when referring to infinite words over sets of atomic propositions: instead of writing the set of atomic propositions corresponding to a symbol, we use simple propositional formulas, such as p, $\neg p$, and $p \wedge q$, to denote all the sets of atomic propositions where these formulas hold true according to the LTL semantics. For instance, given $\mathcal{P} = \{p, q, r\}$, we write $\neg p$ to denote the sets $\emptyset, \{q\}, \{r\}, \{q, r\} \in \Sigma$, and we write $p \wedge q$ to denote the sets $\{p, q\}, \{p, q, r\} \in \Sigma$.

3 The Syntax and Semantics of rLTL($\boxdot, \diamondsuit$)

In this section we introduce the fragment rLTL($\boxdot, \diamondsuit$) that intrinsically provides a robust interpretation of LTL($\Box, \diamondsuit$) formulas. On the one hand, rLTL($\boxdot, \diamondsuit$) is simple enough that we can provide a lucid intuitive explanation for its semantics. On the other hand,

rLTL($\boxdot, \diamondsuit$) already illustrates most of the technical difficulties brought by robustness. Full rLTL, including all technical results, is presented in [21].

3.1 The Syntax of rLTL($\boxdot, \diamondsuit$)

The syntax of rLTL($\boxdot, \diamondsuit$) closely mirrors the syntax of LTL($\Box, \diamondsuit$) with the only noticeable difference being the use of dotted temporal operators.

▶ **Definition 3** (rLTL($\boxdot, \diamondsuit$) syntax)**.** Let $\mathcal{P}$ be a nonempty, finite set of atomic propositions. *rLTL*($\boxdot, \diamondsuit$) *formulas* are inductively defined as follows: (a) each $p \in \mathcal{P}$ is an rLTL formula and (b) if φ and ψ are rLTL formulas, so are $\neg\varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi$, $\varphi \Rightarrow \psi$, $\boxdot\varphi$, and $\diamondsuit\varphi$.

In LTL, we can derive the conjunction and implication operators from negation and disjunction. This is no longer the case in rLTL($\boxdot, \diamondsuit$) since it has a many-valued semantics. For this reason, conjunction and implication occur explicitly in Definition 3.

3.2 Robustness and Counting

Consider the LTL formula $\Box p$ where p is an atomic proposition. There is only one way in which this formula can be satisfied: p holds at every time step. In contrast, there are several ways in which this formula can be violated, and we seek a semantics that distinguishes them.

It seems intuitively clear to the authors that the worst manner in which $\Box p$ fails to be satisfied occurs when p fails to hold at every time step. Although still violating $\Box p$, we would prefer a situation where p holds for at most finitely many time instants. Better yet would be that p holds at infinitely many instants while it fails to hold also at infinitely many instants. Finally, among all the possible ways in which $\Box p$ can be violated, we would prefer the case where p fails to hold for at most finitely many time instants. Consequently, our robust semantics is designed to distinguish between satisfaction and these four possible different ways to violate $\Box p$. However, as convincing as this argument might be, a question persists: in which sense can we regard these five alternatives as canonical?

We answer this question by interpreting satisfaction of $\Box p$ as a counting problem. The previously discussed five different cases, satisfaction and four different types of violation, can be seen as the result of counting the number of 0s and 1s in the word $\alpha = W(\sigma_{0..}, p)W(\sigma_{1..}, p) \ldots \in \mathbb{B}^\omega$ rather than using the inf-operator in (5). From this perspective, satisfaction corresponds to the number of occurrences of 0 being zero. Among all the possible ways in which $\Box p$ can be violated, the most preferred occurs when p only fails to hold at finitely many time instants. This corresponds to a finite number of 0s in α. All the other ways in which $\Box p$ can be violated are similarly identified by the number of 0s and 1s in α.

We say that an LTL($\Box, \diamondsuit$) formula φ is a counting formula if its valuation $W(\sigma, \varphi)$ only depends on the number of occurrences of each atomic proposition but not on its order. Such formula φ is essentially counting how many times each atomic proposition appears along the word σ. Formally, we say that φ is a counting formula if for every infinite word $\sigma \in \Sigma^\omega$, seen as a map $\sigma\colon \mathbb{N} \to \Sigma$, and for every bijection $f\colon \mathbb{N} \to \mathbb{N}$ we have $W(\sigma, \varphi) = W(\sigma \circ f, \varphi)$. Recall that by composing a sequence of permutations (bijections) one again obtains a bijection. Hence, by permuting the elements of σ, we obtain the word $\sigma \circ f$ where f is the composition of the employed permutations. If we now assume $\mathcal{P} = \{p\}$, then we can always permute the elements of σ so that the permuted word $\sigma \circ f$ is of the form $(\neg p\, p)^k p^\omega$, $(\neg p\, p)^\omega$, or $(\neg p\, p)^k (\neg p)^\omega$, where $k \in \mathbb{N}$. We further recall that formulas in LTL($\Box, \diamondsuit$) can only define

stutter-invariant properties [15]. Therefore, the semantics of LTL($\Box, \Diamond$) cannot distinguish[1] between the words $(\neg p\, p)^{k_1}\, p^\omega$ and $(\neg p\, p)^{k_2}\, p^\omega$ for $k_1 \neq k_2$, and $k_1, k_2 > 0$, although it can distinguish between the case $k_1 = 0$ and $k_2 > 0$. The same argument applies to the words $(\neg p\, p)^{k_1}\, (\neg p)^\omega$ and $(\neg p\, p)^{k_2}\, (\neg p)^\omega$ and shows that there are only five canonical forms that can be distinguished by LTL($\Box, \Diamond$):

$$p^\omega, (\neg p\, p)^+\, p^\omega, (\neg p\, p)^\omega, (\neg p\, p)^+\, (\neg p)^\omega, \text{ and } (\neg p)^\omega. \tag{7}$$

It should be no surprise that these are exactly the five cases we previously discussed.

The considerations in this section suggest the need for a semantics that is 5-valued rather than 2-valued so that we can distinguish between the aforementioned five cases. Therefore, we need to replace Boolean algebras by a different type of algebraic structure that can accommodate a 5-valued semantics for each rLTL($\boxdot, \diamondsuit$) formula.

3.3 da Costa Algebras

According to our motivating example $\Box p$, the desired semantics should have one truth value corresponding to *true* and four truth values corresponding to different shades of *false*. It is instructive to think of truth values as the elements of $\mathbb{B}^4$ (i.e., the four-fold Cartesian product of $\mathbb{B}$) that arise as the possible values of the 4-tuple of LTL formulas:

$$(\Box p, \Diamond\Box p, \Box\Diamond p, \Diamond p). \tag{8}$$

To ease notation, we denote such values interchangeably by $b = b_1b_2b_3b_4$ and $b = (b_1, b_2, b_3, b_4)$ with $b_i \in \mathbb{B}$ for $i \in \{1, 2, 3, 4\}$. The value 1111 then corresponds to *true* since $\Box p$ is satisfied. The most preferred violation of $\Box p$ (p fails to hold at only finitely many time instants) corresponds to 0111, followed by 0011 (p holds at infinitely many instants and also fails to hold at infinitely many instants), 0001 (p holds at most at finitely many instants), and 0000 (p fails to hold at every time instant). Such preferences can be encoded in the linear order

$$0000 \prec 0001 \prec 0011 \prec 0111 \prec 1111 \tag{9}$$

that renders the set $\mathbb{B}_4 = \{0000, 0001, 0011, 0111, 1111\}$ a (bounded) distributive lattice with top element $\top = 1111$ and bottom element $\bot = 0000$. In $\mathbb{B}_4$, the meet $\sqcap$ can be interpreted as minimum and the join $\sqcup$ as maximum with respect to the order in (9). We use $\sqcap$ and $\sqcup$ when discussing lattices in general and use min and max for the specific lattice $\mathbb{B}_4$ or the Boolean algebra $\mathbb{B}$.

The first choice to be made in using the lattice $\mathbb{B}_4$ to define the semantics of rLTL($\boxdot, \diamondsuit$) is the choice of an operation on $\mathbb{B}_4$ modeling conjunction. Consider the formula $\Box p \wedge \Box q$ and the word $\sigma = \neg(p \wedge q)(p \wedge q)^\omega$. As introduced above, the value of $\Box p$ on σ corresponds to 0111 and the value of $\Box q$ on σ corresponds to 0111 since on both cases we have the most preferred violation of the formulas. Therefore, the value of $\Box p \wedge \Box q$ on σ should also be 0111 since the formula $\Box p \wedge \Box q$ is only violated a finite number of times. It thus seems natural[2]

[1] To see why this is the case, note that any word $(\neg p\, p)^k\, p^\omega$ with $k \in \mathbb{N}$ can be permuted to the form $(\neg p)^k\, p^k p^\omega$ and by stutter invariance can be reduced to $\neg p\, p\, p^\omega$.

[2] Note that there are situations where it is convenient to model conjunction differently. In the related work, we referenced the work of Bloem et al. [4], where the specific way in which robustness is modeled requires distinguishing between the number of conjuncts that are satisfied in the assumption $\wedge_{i \in I} \varphi_i$. This cannot be accomplished if conjunction is modeled by min and a different triangular-norm would have to be used for this purpose. Note that both Łukasiewicz's conjunction as well as Goguen's conjunction have the property that their value decreases as the number of conjuncts that are true decreases.

to model conjunction in $\mathbb{B}_4$ by min and, for similar reasons, to model disjunction in $\mathbb{B}_4$ by max.

As in intuitionistic logic[3], our implication is defined as the residue of $\sqcap$. In other words, we define the implication $a \to b$ by requiring that $c \preceq a \to b$ if and only if $c \sqcap a \preceq b$ for every $c \in \mathbb{B}_4$. This leads to

$$a \to b = \begin{cases} 1111 & \text{if } a \preceq b; \text{ and} \\ b & \text{otherwise.} \end{cases}$$

However, we now *diverge* from intuitionistic logic (and most many-valued logics) where negation of a is defined by $a \to 0000$. Such negation is not compatible with the interpretation that all the elements of $\mathbb{B}_4$, except for 1111, represent (different shades of) *false* and thus their negation should have the truth value 1111. To make this point clear, the table below presents the intuitionistic negation in $\mathbb{B}_4$ and the desired negation compatible with our interpretation of the truth values in $\mathbb{B}_4$.

Value	Desired negation	Intuitionistic negation
1111	0000	0000
0111	1111	0000
0011	1111	0000
0001	1111	0000
0000	1111	1111

What is then the algebraic structure on $\mathbb{B}_4$ that supports the desired negation? This very same problem was recently investigated by Priest [18], and the answer is *da Costa* algebras.

▶ **Definition 4** (da Costa algebra). A *da Costa* algebra is a 6-tuple $(A, \sqcap, \sqcup, \preceq, \to, \overline{\cdot})$ where

1. $(A, \sqcap, \sqcup, \preceq)$ is a distributive lattice where $\preceq$ is the ordering derived from $\sqcap$ and $\sqcup$;
2. $\to$ is the residual of $\sqcap$ (i.e., $a \preceq b \to c$ if and only if $a \sqcap b \preceq c$ for every $a, b, c \in A$);
3. $a \preceq b \sqcup \overline{b}$ for every $a, b \in A$; and
4. $\overline{a} \preceq b$ whenever $c \sqcup \overline{c} \preceq a \sqcup b$ for every $a, b, c \in A$.

Indeed, $\mathbb{B}_4$ is a da Costa algebra if we use the desired negation defined in the table above.

It should be mentioned that working with a 5-valued semantics has its price. The law of non-contradiction fails in $\mathbb{B}_4$ (i.e., $a \sqcap \overline{a}$ may not equal $\bot = 0000$ as evidenced by taking $a = 0111$). However, since $a \sqcap \overline{a} \prec 1111$, a weak form of non-contradiction still holds as $a \sqcap \overline{a}$ is to be interpreted as a shade of *false* but not necessarily as the least preferred way of violating $a \sqcap \overline{a}$, which corresponds to $\bot$. Contrary to intuitionistic logic, the law of excluded middle is valid (i.e., $a \sqcup \overline{a} = \top = 1111$). Finally, $a = 0111$ shows that $\overline{\overline{a}} \neq a$ although it is still true that $\overline{\overline{a}} \to a$.

[3] This is also done in context of residuated lattices that is more general than the Heyting algebras used in intuitionistic logic. Recall that a residuated lattice is a lattice $(A, \sqcap, \sqcup)$, satisfying same additional conditions, and equipped with a commutative monoid $(A, \otimes, \mathbf{1})$ satisfying some additional compatibility conditions. Since we chose the lattice meet $\sqcap$ to represent conjunction, we have a residuated lattice where $\otimes = \sqcap$ and $\mathbf{1} = \top$.

3.4 Semantics of rLTL($\boxdot, \diamondsuit$) on da Costa Algebras

The semantics of rLTL($\boxdot, \diamondsuit$) is given by a mapping V, called *valuation* as in the case of LTL($\Box, \diamondsuit$), that maps an infinite word $\sigma \in \Sigma^\omega$ and an rLTL($\boxdot, \diamondsuit$) formula φ to an element of $\mathbb{B}_4$. In defining V, we judiciously use the algebraic operations of the da Costa algebra $\mathbb{B}_4$ to give meaning to the logical connectives in the syntax of rLTL($\boxdot, \diamondsuit$). In the following, let $\Sigma = 2^{\mathcal{P}}$ for a finite set of atomic propositions $\mathcal{P}$.

On atomic propositions $p \in \mathcal{P}$, V is defined "classically" by

$$V(\sigma, p) = \begin{cases} 0000 & \text{if } p \notin \sigma(0); \text{ and} \\ 1111 & \text{if } p \in \sigma(0). \end{cases} \tag{10}$$

Since we are using a 5-valued semantics, we provide a separate definition for all the four logical connectives:

$$V(\sigma, \varphi \wedge \psi) = V(\sigma, \varphi) \sqcap V(\sigma, \psi) \tag{11}$$

$$V(\sigma, \neg\varphi) = \overline{V(\sigma, \varphi)} \tag{12}$$

$$V(\sigma, \varphi \vee \psi) = V(\sigma, \varphi) \sqcup V(\sigma, \psi) \tag{13}$$

$$V(\sigma, \varphi \Rightarrow \psi) = V(\sigma, \varphi) \rightarrow V(\sigma, \psi) \tag{14}$$

Note how the semantics mirrors the algebraic structure of da Costa algebras. This is no accident since valuations are typically algebra homomorphisms. Unfortunately, da Costa algebras are not equipped[4] with operations corresponding to $\boxdot$ and $\diamondsuit$, the robust versions of $\Box$ and $\diamondsuit$, respectively. Therefore, we resort to the counting interpretation in Section 3.2 to motivate the semantics of $\boxdot$. Formally, we have

$$V(\sigma, \boxdot\varphi) = \big(\inf_{i\geq 0} V_1(\sigma_{i..}, \varphi),\ \sup_{j\geq 0}\inf_{i\geq j} V_2(\sigma_{i..}, \varphi), \inf_{j\geq 0}\sup_{i\geq j} V_3(\sigma_{i..}, \varphi),\ \sup_{i\geq 0} V_4(\sigma_{i..}, \varphi)\big) \tag{15}$$

where $V_k(\sigma, \varphi) = \pi_k \circ V(\sigma, \varphi)$ for $k \in \{1, 2, 3, 4\}$ and $\pi_k : \mathbb{B}_4 \to \mathbb{B}$ are mappings defined by

$$\pi_k(a_1, a_2, a_3, a_4) = a_k. \tag{16}$$

To illustrate the semantics of $\boxdot$, consider the simple case where φ is an atomic proposition p. This means that one can express $V(\sigma, \boxdot p)$ in terms of the LTL valuation W by

$$V(\sigma, \boxdot p) = \big(W(\sigma, \Box p), W(\sigma, \diamondsuit\Box p), W(\sigma, \Box\diamondsuit p), W(\sigma, \diamondsuit p)\big) \tag{17}$$

thus connecting the semantics of $\boxdot$ to the counting problems described in Section 3.2 and to the 4-tuple of LTL formulas in (8).

The last operator is $\diamondsuit$ and its semantics is given by

$$V(\sigma, \diamondsuit\varphi) = \big(\sup_{i\geq 0} V_1(\sigma_{i..}, \varphi),\ \sup_{i\geq 0} V_2(\sigma_{i..}, \varphi), \sup_{i\geq 0} V_3(\sigma_{i..}, \varphi),\ \sup_{i\geq 0} V_4(\sigma_{i..}, \varphi)\big). \tag{18}$$

According to the counting problems of Section 3.2, there is only one way in which the LTL formula $\diamondsuit p$, for an atomic proposition p, can be violated, namely p never holds. Hence, $V(\sigma, \diamondsuit\varphi)$ is one of only two possible truth values: 1111 or 0000. We further note that $\diamondsuit$ is not dual to $\boxdot$, as expected in a many-valued logic.

Having defined the semantics of rLTL($\boxdot, \diamondsuit$), let us now see if the formula $\boxdot p \Rightarrow \boxdot q$, where $\boxdot p$ is an environment assumption and $\boxdot q$ is a system guarantee with $p, q \in \mathcal{P}$, lives to the expectations set in the introduction and to the intuition provided in Section 3.2.

[4] One could consider developing a notion of da Costa algebras with operators in the spirit of Boolean algebras with operators [12]. We leave such investigation for future work.

1. According to (17), if $\Box p$ holds, then $\boxdot p$ evaluates to 1111 and the implication $\boxdot p \Rightarrow \boxdot q$ is *true* (i.e., the value of $\boxdot p \Rightarrow \boxdot q$ is 1111) if $\boxdot q$ evaluates to 1111 (i.e., if $\Box q$ holds). Hence, the desired behavior of $\Box p \Rightarrow \Box q$, when the environment assumptions hold, is retained.
2. Consider now the case where $\Box p$ fails but the weaker assumption $\Diamond\Box p$ holds. In this case $\boxdot p$ evaluates to 0111 and the implication $\boxdot p \Rightarrow \boxdot q$ is *true* if $\boxdot q$ evaluates to 0111 or higher. This means that $\Diamond\Box q$ needs to hold.
3. A similar argument shows that we can also conclude the following consequences whenever $\boxdot p \Rightarrow \boxdot q$ evaluates to 1111: $\Box\Diamond q$ follows whenever the environment satisfies $\Box\Diamond p$ and $\Diamond q$ follows whenever the environment satisfies $\Diamond p$.

We thus conclude that the semantics of $\boxdot p \Rightarrow \boxdot q$ captures the desired robustness property by which a weakening of the assumption $\boxdot p$ leads to a weakening of the guarantee $\boxdot q$. The following example further motivates the usefulness of the proposed semantics.

3.5 Examples

The usefulness of implications that are not true: We argued in the previous section that rLTL($\boxdot, \diadot$) captures the intended robustness properties for the specification $\boxdot p \Rightarrow \boxdot q$ whenever this formula is *true* (i.e., evaluates to 1111). But does the formula $\boxdot p \Rightarrow \boxdot q$ still provide useful information when its valuation is lower than 1111? It follows from the semantics of implication that $V(\sigma, \boxdot p \Rightarrow \boxdot q) = b$, for $b \prec 1111$, occurs when $V(\sigma, \boxdot q) = b$ (i.e., whenever a value of b can be guaranteed despite b being smaller than $V(\sigma, \boxdot p)$). The value $V(\sigma, \boxdot p \Rightarrow \boxdot q)$ thus describes which weakened guarantee follows from the environment assumption whenever the intended system guarantee does not. This can be seen as another measure of robustness: despite $\boxdot q$ not following from $\boxdot p$, the behavior of the system is not arbitrary, a value of b is still guaranteed.

Non-counting formulas: Consider the non-counting formula $\Box(p \Rightarrow \Diamond q)$, which requires each occurrence of p to be followed by an occurrence of q. This formula is routinely used in the literature to illustrate LTL and, hence, constitutes a litmus test to rLTL($\boxdot, \diadot$). The semantics of the dotted version of $\Box(p \Rightarrow \Diamond q)$ can be expressed using the LTL valuation as

$$V(\sigma, \boxdot(p \Rightarrow \diadot q)) = \\ \big(W(\sigma, \Box(p \Rightarrow \Diamond q)), W(\sigma, \Box\Diamond p \Rightarrow \Box\Diamond q), W(\sigma, \Diamond\Box p \Rightarrow \Box\Diamond q), W(\sigma, \Box p \Rightarrow \Diamond q)\big).$$

It is interesting to observe how the semantics of $\varphi = \boxdot(p \Rightarrow \diadot q)$ recovers: 1) strong fairness, also known as compassion, when the value of φ is 0111; 2) weak fairness, also known as justice, when the value of φ is 0011; and 3) the even weaker notion of fairness described by the formula $\Box p \Rightarrow \Diamond q$, when the value of φ is 0001. The fact that all these different and well known notions of fairness naturally appear in the proposed semantics is another strong indication of rLTL's naturalness and usefulness.

3.6 Relating LTL($\Box, \Diamond$) and rLTL($\boxdot, \diadot$)

In this section we discuss, at the technical level, the relationships between rLTL($\boxdot, \diadot$) and LTL($\Box, \Diamond$).

First, we argue that the semantics of LTL($\Box, \Diamond$) can always be recovered from the first component of the semantics of rLTL($\boxdot, \diadot$). To this end, recall the mapping $\pi_1 \colon \mathbb{B}_4 \to \mathbb{B}$ introduced in (16), defined by $\pi_1(a_1, a_2, a_3, a_4) = a_1$. Composing π_1 with a valuation V

of rLTL($\boxdot, \diamondsuit$), we obtain the function $V_1 = \pi_1 \circ V$ mapping an infinite word $\sigma \in \Sigma^\omega$ and a rLTL($\boxdot, \diamondsuit$) formula φ to a Boolean value. In fact, V_1 corresponds to the LTL($\square, \diamondsuit$) semantics, which implies that rLTL($\boxdot, \diamondsuit$) is as expressive as LTL($\square, \diamondsuit$). The proof of this claim is a straightforward case distinction. On atomic propositions $p \in \mathcal{P}$ we have

$$V_1(\sigma, p) = \begin{cases} \pi_1(0000) = 0 & \text{if } p \notin \sigma(0); \text{ and} \\ \pi_1(1111) = 1 & \text{if } p \in \sigma(0). \end{cases}$$

Moreover, the following holds for "Boolean" connectives:

$$\begin{aligned} V_1(\sigma, \varphi \wedge \psi) &= \pi_1\left(V(\sigma, \varphi) \sqcap V(\sigma, \psi)\right) = \min\{V_1(\sigma, \varphi), V_1(\sigma, \psi)\}, \\ V_1(\sigma, \varphi \vee \psi) &= \pi_1\left(V(\sigma, \varphi) \sqcup V(\sigma, \psi)\right) = \max\{V_1(\sigma, \varphi), V_1(\sigma, \psi)\}, \\ V_1(\sigma, \neg\varphi) &= \pi_1(\overline{V(\sigma, \varphi)}) = 1 - \pi_1\left(V(\sigma, \varphi)\right) = 1 - V_1(\sigma, \varphi), \\ V_1(\sigma, \varphi \Rightarrow \psi) &= \pi_1\left(V(\sigma, \varphi) \to V(\sigma, \psi)\right) = \max\left\{1 - V_1(\sigma, \varphi), V_1(\sigma, \psi)\right\}. \end{aligned}$$

Finally, the following follows directly from the semantics of $\boxdot$ and $\diamondsuit$:

$$\begin{aligned} V_1(\sigma, \boxdot\varphi) &= \pi_1\left(V(\sigma, \boxdot\varphi)\right) = \inf_{i \geq 0} V_1(\sigma_{i..}, \varphi), \\ V_1(\sigma, \diamondsuit\varphi) &= \pi_1\left(V(\sigma, \diamondsuit\varphi)\right) = \sup_{i \geq 0} V_1(\sigma_{i..}, \varphi). \end{aligned}$$

It is not hard to verify that V_1 in all cases corresponds to the LTL($\square, \diamondsuit$) valuation.

Conversely, one can translate an rLTL($\boxdot, \diamondsuit$) formula φ into four LTL($\square, \diamondsuit$) formulas $\psi_\varphi^1, \ldots, \psi_\varphi^4$ such that

$$\pi_j(V(\sigma, \varphi)) = V_j(\sigma, \varphi) = W(\sigma, \psi_\varphi^j)$$

for all $\sigma \in \Sigma^\omega$ and $j \in \{1, \ldots, 4\}$. The key idea is to emulate the semantics of each operator occurring in φ component-wise by means of dedicated LTL formulas.

The construction of ψ_φ^j proceeds by induction over the subformulas of φ:

- If $\varphi = p$ for an atomic proposition $p \in \mathcal{P}$, then $\psi_\varphi^j \coloneqq p$ for all $j \in \{1, \ldots, 4\}$.
- If $\varphi = \varphi_1 \vee \varphi_2$, then $\psi_\varphi^j \coloneqq \psi_{\varphi_1}^j \vee \psi_{\varphi_2}^j$ for all $j \in \{1, \ldots, 4\}$.
- If $\varphi = \varphi_1 \wedge \varphi_2$, then $\psi_\varphi^j \coloneqq \psi_{\varphi_1}^j \wedge \psi_{\varphi_2}^j$ for all $j \in \{1, \ldots, 4\}$.
- If $\varphi = \neg\varphi_1$, then $\psi_\varphi^j \coloneqq \neg(\psi_{\varphi_1}^1 \wedge \psi_{\varphi_1}^2 \wedge \psi_{\varphi_1}^3 \wedge \psi_{\varphi_1}^4)$ for all $j \in \{1, \ldots, 4\}$.
- If $\varphi = \varphi_1 \Rightarrow \varphi_2$, then $\psi_\varphi^j \coloneqq \left(\bigvee_{k \in \{1,\ldots,4\}} (\psi_{\varphi_1}^k \wedge \neg\psi_{\varphi_2}^k)\right) \Rightarrow \psi_{\varphi_2}^j$ for all $j \in \{1, \ldots, 4\}$.
- If $\varphi = \diamondsuit\varphi_1$, then $\psi_\varphi^j \coloneqq \diamondsuit\psi_{\varphi_1}^j$ for all $j \in \{1, \ldots, 4\}$.
- If $\varphi = \boxdot\varphi_1$, then $\psi_\varphi^1 \coloneqq \square\psi_{\varphi_1}^1$, $\psi_\varphi^2 \coloneqq \diamondsuit\square\psi_{\varphi_1}^2$, $\psi_\varphi^3 \coloneqq \square\diamondsuit\psi_{\varphi_1}^3$, and $\psi_\varphi^4 \coloneqq \diamondsuit\psi_{\varphi_1}^4$.

It is not hard to verify that the formulas ψ_φ^j have indeed the desired meaning. However, note that the size of ψ_φ^j, measured in the number of subformulas, is exponential in the size of φ due to the recursive substitution of the sub-formulas.

The preceding discussion can be summarized by the following result.

▶ **Proposition 5.** *LTL($\square, \diamondsuit$) and rLTL($\boxdot, \diamondsuit$) are equally expressive.*

Since the translations from LTL($\square, \diamondsuit$) to rLTL($\boxdot, \diamondsuit$) and vice versa are effective, we conclude that any problem for rLTL($\boxdot, \diamondsuit$), whose corresponding problem for LTL($\square, \diamondsuit$) is decidable, is also decidable. In practice, however, the translation from rLTL($\boxdot, \diamondsuit$) to LTL($\square, \diamondsuit$) involves an exponential blow-up. Hence, we now investigate the complexity of several verification and synthesis problems by developing specialized algorithms.

4 Model Checking and Synthesis

Similarly to LTL, rLTL($\boxdot, \diamondsuit$) gives rise to various (decision) problems, some of which we investigate in this section. We are particularly interested in model checking and in reactive synthesis. These two problems are clearly amongst the most important in the context of LTL and, hence, must be investigated for rLTL($\boxdot, \diamondsuit$).

As the translation from rLTL($\boxdot, \diamondsuit$) into LTL($\Box, \Diamond$) potentially results in an exponentially large formula, we now develop a computationally more efficient approach to the model checking and reactive synthesis problems via a translation into (generalized) Büchi automata. Our construction follows the well known translation of LTL into Büchi automata (see, e.g., Baier and Katoen [3]) and results in a generalized Büchi automaton with $\mathcal{O}(k \cdot 5^k)$ states where k counts the subformulas of the given rLTL($\boxdot, \diamondsuit$) formula. This is the same complexity as for the LTL translation – which results in an automation with size in $\mathcal{O}(k \cdot 2^k)$ – once we replace 2 with 5 since rLTL is 5-valued while LTL is 2-valued.

Similarly to LTL, our translation relies on so-called expansion rules, which we introduce in Section 4.1. Based on these rules, we present the translation from rLTL($\boxdot, \diamondsuit$) to generalized Büchi automata in Section 4.2. Subsequently, we consider model checking in Section 4.3 and reactive synthesis in Section 4.4.

4.1 Expansion Rules

The temporal operators $\boxdot$ and $\diamondsuit$ have expansion rules, which relate valuations at time ℓ to that of time $\ell + 1$, similar to their LTL counterparts $\Box$ and $\Diamond$ (see Baier and Katoen [3] for an in-depth discussion of LTL expansion rules). The following proposition states these rules.

▶ **Proposition 6** (Expansion Rules). *For any rLTL($\boxdot, \diamondsuit$) formula φ, any $\sigma \in \Sigma^\omega$, any $\ell \in \mathbb{N}$, and any valuation V, the following equalities (called* expansion rules*) hold:*

$$V_1(\sigma_{\ell..}, \boxdot\varphi) = \min\{V_1(\sigma_{\ell..}, \varphi), V_1(\sigma_{\ell+1..}, \boxdot\varphi)\}, \tag{19}$$
$$V_2(\sigma_{\ell..}, \boxdot\varphi) = \max\{V_1(\sigma_{\ell..}, \boxdot\varphi), V_2(\sigma_{\ell+1..}, \boxdot\varphi)\}, \tag{20}$$
$$V_3(\sigma_{\ell..}, \boxdot\varphi) = \min\{V_4(\sigma_{\ell..}, \boxdot\varphi), V_3(\sigma_{\ell+1..}, \boxdot\varphi)\}, \tag{21}$$
$$V_4(\sigma_{\ell..}, \boxdot\varphi) = \max\{V_4(\sigma_{\ell..}, \varphi), V_4(\sigma_{\ell+1..}, \boxdot\varphi)\}. \tag{22}$$

Moreover, the following holds for each $k \in \{1, \ldots, 4\}$:

$$V_k(\sigma_{\ell..}, \diamondsuit\varphi) = \max\{V_k(\sigma_{\ell..}, \varphi), V_k(\sigma_{\ell+1..}, \diamondsuit\varphi)\}. \tag{23}$$

Before we prove this proposition, let us highlight that Equation (20) does not only recur on V_2 but also on V_1 (an analogous observation is true for Equation (21)). In fact, by recurring on $V_1(\sigma_{\ell..}, \boxdot\varphi)$ instead of $\sup_{k \geq \ell} V_2(\sigma_{k..}, \varphi)$, as one might have expected, we avoid the intermediate computation of $\sup_{k \geq \ell} V_2(\sigma_{k..}, \varphi)$ by the generalized Büchi automaton and, thereby, save auxiliary memory. This is the key property that allows us to prevent an unduly growth in the size of the resulting Büchi automaton and to achieve the desired bound on the number of states.

Proof of Proposition 6. Equality (19) follows directly from the properties of inf:

$$\begin{aligned} V_1(\sigma_{\ell..}, \boxdot\varphi) &= \inf_{i\geq\ell} V_1(\sigma_{i..}, \varphi) \\ &= \inf\{V_1(\sigma_{\ell..}, \varphi), V_1(\sigma_{\ell+1..}, \varphi), V_1(\sigma_{\ell+2..}, \varphi), \ldots\} \\ &= \inf\{V_1(\sigma_{\ell..}, \varphi), \inf\{V_1(\sigma_{\ell+1..}, \varphi), V_1(\sigma_{\ell+2..}, \varphi), \ldots\}\} \\ &= \min\left\{V_1(\sigma_{\ell..}, \varphi), \inf_{i\geq\ell+1} V_1(\sigma_{i..}, \varphi)\right\} \\ &= \min\{V_1(\sigma_{\ell..}, \varphi), V_1(\sigma_{\ell+1..}, \boxdot\varphi)\}. \end{aligned}$$

A similar argument using the properties of sup shows that

$$V_2(\sigma_{\ell..}, \boxdot\varphi) = \sup_{j\geq\ell}\inf_{i\geq j} V_2(\sigma_{i..}, \varphi) = \max\left\{\inf_{i\geq\ell} V_2(\sigma_{i..}, \varphi), \sup_{j\geq\ell+1}\inf_{i\geq j} V_2(\sigma_{i..}, \varphi)\right\}.$$

To conclude the proof of Equality (20), we need to replace the term $\inf_{i\geq\ell} V_2(\sigma_{i..}, \varphi)$ inside the max by $\inf_{i\geq\ell} V_1(\sigma_{i..}, \varphi)$; in other words, we must prove the last equality in the equation

$$\begin{aligned} \max\{V_1(\sigma_{\ell..}, \boxdot\varphi), V_2(\sigma_{\ell+1..}, \boxdot\varphi)\} &= \max\left\{\inf_{i\geq\ell} V_1(\sigma_{i..}, \varphi), \sup_{j\geq\ell+1}\inf_{i\geq j} V_2(\sigma_{i..}, \varphi)\right\} \\ &= \max\left\{\inf_{i\geq\ell} V_2(\sigma_{i..}, \varphi), \sup_{j\geq\ell+1}\inf_{i\geq j} V_2(\sigma_{i..}, \varphi)\right\} \end{aligned} \tag{24}$$

holds for every sequence $\sigma \in \Sigma^\omega$, every rLTL($\boxdot, \diamondsuit$) formula φ, and any valuation V.

To this end, we consider two separate cases. The first case is $\sup_{j\geq\ell+1}\inf_{i\geq j} V_2(\sigma_{i..}, \varphi) = 1$ and immediately leads to the desired equality

$$\begin{aligned} \max\left\{\inf_{i\geq\ell} V_1(\sigma_{i..}, \varphi), \sup_{j\geq\ell+1}\inf_{i\geq j} V_2(\sigma_{i..}, \varphi)\right\} &= 1 \\ &= \max\left\{\inf_{i\geq\ell} V_2(\sigma_{i..}, \varphi), \sup_{j\geq\ell+1}\inf_{i\geq j} V_2(\sigma_{i..}, \varphi)\right\}. \end{aligned}$$

The second case is $\sup_{j\geq\ell+1}\inf_{i\geq j} V_2(\sigma_{i..}, \varphi) = 0$ and the desired equality reduces to

$$\inf_{i\geq\ell} V_1(\sigma_{i..}, \varphi) = \inf_{i\geq\ell} V_2(\sigma_{i..}, \varphi).$$

We now note that $\sup_{j\geq\ell+1}\inf_{i\geq j} V_2(\sigma_{i..}, \varphi) = 0$ implies $\inf_{i\geq\ell+1} V_2(\sigma_{i..}, \varphi) = 0$, which implies $\inf_{i\geq\ell} V_2(\sigma_{i..}, \varphi) = 0$. To conclude the proof, we must show $\inf_{i\geq\ell} V_1(\sigma_{i..}, \varphi) = 0$. We recall that every element $b = (b_1, b_2, b_3, b_4) \in \mathbb{B}_4$ satisfies $b_1 \leq b_2$. In particular, we have $V_1(\sigma_{i..}, \varphi) \leq V_2(\sigma_{i..}, \varphi)$ for every $i \in \mathbb{N}$, and it follows from the monotonicity properties of inf that

$$\inf_{i\geq\ell} V_1(\sigma_{i..}, \varphi) \leq \inf_{i\geq\ell} V_2(\sigma_{i..}, \varphi).$$

The proof of Equality (20) is now finished by noting that the previous inequality and $\inf_{i\geq\ell} V_2(\sigma_{i..}, \varphi) = 0$ imply $\inf_{i\geq\ell} V_1(\sigma_{i..}, \varphi) = 0$.

The proof of Equality (21) is dual to the proof of Equality (20), while the proof of Equality (22) is dual to the proof of Equality (19). ◀

4.2 From rLTL($\boxdot, \diamondsuit$) to Generalized Büchi Automata

Let us begin this section by briefly recalling the definition of generalized Büchi automata.

▶ **Definition 7** (Generalized Büchi automaton)**.** A *generalized Büchi automaton (GBA)* is a tuple $\mathcal{A} = (Q, \Sigma, q_0, \Delta, \mathcal{F})$ consisting of a nonempty, finite set Q of states, a (finite) input alphabet Σ, an initial state $q_0 \in Q$, a (nondeterministic) transition relation $\Delta \in Q \times \Sigma \times Q$, and a set $\mathcal{F} \subseteq 2^Q$ denoting the acceptance conditions.

The *run* of a generalized Büchi automaton on a word $\sigma \in \Sigma^\omega$ (also called *input*) is an infinite sequence of states $\rho = q_0 q_1 \ldots \in Q^\omega$ satisfying $(q_i, \sigma(i), q_{i+1}) \in \Delta$ for all $i \in \mathbb{N}$ (note that every run starts in the initial state q_0). Given a run $\rho = q_0 q_1 \ldots$, we denote the set of states occurring infinitely often during ρ by $\mathrm{Inf}(\rho) = \{q \in Q \mid \forall i \in \mathbb{N}\ \exists j \geq i \colon q_j = q\}$. A run ρ is called *accepting* if $\mathrm{Inf}(\rho) \cap F \neq \emptyset$ for all $F \in \mathcal{F}$ (i.e., the run visits a state of each set $F \in \mathcal{F}$ infinitely often). The *language* of a generalized Büchi automaton $\mathcal{A}$, denoted by $L(\mathcal{A})$, is the set of all infinite words $\sigma \in \Sigma^\omega$ for which an accepting run of $\mathcal{A}$ exists.

Having recapitulated these basic definitions, we can now turn to our translation from rLTL($\boxdot, \diamondsuit$) to generalized Büchi automata. This translation is an adaptation of a standard translation from LTL to generalized Büchi automata that constructs an automaton whose states correspond to valuations of the given LTL formula. An important concept in this context is that of the so-called φ-*expansion*: given an rLTL($\boxdot, \diamondsuit$) formula φ, the φ-expansion of an infinite word $\sigma \in \Sigma^\omega$ tracks the evaluation of φ and its subformulas at each position of σ. The key idea then is to construct a generalized Büchi automaton $\mathcal{A}_\varphi$ that nondeterministically guesses the φ-expansion step-by-step when reading its input and verifies the guess by means of its acceptance conditions. A formal definition of the φ-expansion is given below. To ease the following presentation, we denote the set of sub-formulas of an rLTL($\boxdot, \diamondsuit$) formula φ, which is defined in the usual way, by *closure of* φ, abbreviated as $\mathrm{cl}(\varphi)$.

▶ **Definition 8** (φ-expansion)**.** Let φ be an rLTL($\boxdot, \diamondsuit$) formula. The φ-*expansion* of an infinite word $\sigma \in \Sigma^\omega$ is a mapping $\eta \colon cl(\varphi) \times \mathbb{N} \to \mathbb{B}_4$ satisfying $\eta(\psi, i) = V(\sigma_{i..}, \psi)$ for all $\psi \in cl(\varphi)$ and $i \in \mathbb{N}$.

Note that the φ-expansion is unique for a given word and subsumes the valuation of φ in the sense that $V(\varphi, \sigma) = \eta(\varphi, 0)$. Although the definition of the φ-expansion is not constructive, we can introduce constraints that completely characterize the φ-expansion of a given word. The pivotal idea is to impose constraints for local consistency (e.g., $\eta(\neg\psi, i)$ for $\psi \in \mathrm{cl}(\varphi)$ at some position $i \in \mathbb{N}$ has to be $\overline{\eta(\psi, i)}$) and to exploit the expansion rules of Proposition 6 to relate $\eta(\psi, i)$ and $\eta(\psi, i+1)$. As in the case of valuations V, we use the shorthand-notation $\eta_j(\psi, i)$ instead of the more verbose expression $\pi_j(\eta(\psi, i))$.

In the following, let $\psi \in cl(\varphi)$ and $i \in \mathbb{N}$. The first type of constraints (*local constraints*) are as follows:

A1. If $\psi = p$, then $\eta(\psi, i) = \begin{cases} 0000 & \text{if } p \notin \sigma(i); \text{ and} \\ 1111 & \text{if } p \in \sigma(i). \end{cases}$

A2. If $\psi = \neg\psi_1$, then $\eta(\psi, i) = \overline{\eta(\psi_1, i)}$.

A3. If $\psi = \psi_1 \wedge \psi_2$, then $\eta(\psi, i) = \min\{\eta(\psi_1, i), \eta(\psi_2, i)\}$.

A4. If $\psi = \psi_1 \vee \psi_2$, then $\eta(\psi, i) = \max\{\eta(\psi_1, i), \eta(\psi_2, i)\}$.

A5. If $\psi = \psi_1 \Rightarrow \psi_2$, then $\eta(\psi, i) = \eta(\psi_1, i) \to \eta(\psi_2, i)$.

A6. If $\psi = \diamondsuit\psi_1$, then $\eta(\psi, i) = (b_1, b_2, b_3, b_4)$ where $b_j = \max\{\eta_j(\psi_1, i), \eta_j(\psi, i+1)\}$ for $j \in \{1, \ldots, 4\}$.

A7. If $\psi = \boxdot\psi_1$, then $\eta(\psi, i) = (b_1, b_2, b_3, b4)$ where

(a) $b_1 = \min\{\eta_1(\psi_1, i), \eta_1(\psi, i+1)\}$,
(b) $b_2 = \max\{b_1, \eta_2(\psi, i+1)\}$,
(c) $b_3 = \min\{b_4, \eta_3(\psi, i+1)\}$, and
(d) $b_4 = \max\{\eta_4(\psi_1, i), \eta_4(\psi, i+1)\}$.

To ensure satisfaction of the subformulas involving the temporal operators $\diamondsuit$ and $\boxdot$, we add the following further constraints (*non-local constraints*). These constraints are derived from the expansion rules and translate directly into Büchi conditions.

B1. For each $\diamondsuit\psi \in cl(\varphi)$ and $j \in \{1, \dots, 4\}$, there exists no $k \in \mathbb{N}$ such that for every $\ell \geq k$ both $\eta_j(\diamondsuit\psi, \ell) = 1$ and $\eta_j(\psi, \ell) = 0$.

B2. For each $\boxdot\psi \in cl(\varphi)$,
(a) there exists no $k \in \mathbb{N}$ such that for every $\ell \geq k$ both $\eta_1(\boxdot\psi, \ell) = 0$ and $\eta_1(\psi, \ell) = 1$;
(b) there exists no $k \in \mathbb{N}$ such that for every $\ell \geq k$ both $\eta_2(\boxdot\psi, \ell) = 1$ and $\eta_1(\boxdot\psi, \ell) = 0$;
(c) there exists no $k \in \mathbb{N}$ such that for every $\ell \geq k$ both $\eta_3(\boxdot\psi, \ell) = 0$ and $\eta_4(\boxdot\psi, \ell) = 1$;
(d) there exists no $k \in \mathbb{N}$ such that for every $\ell \geq k$ both $\eta_4(\boxdot\psi, \ell) = 1$ and $\eta_4(\psi, \ell) = 0$.

In fact, these constraints completely characterize the φ-expansion of a given word, which is formally proven in Appendix A.

▶ **Lemma 9.** *Given an rLTL($\boxdot, \diamondsuit$) formula φ over the atomic propositions $\mathcal{P}$ and an infinite word $\sigma \in \Sigma^\omega$ where $\Sigma = 2^\mathcal{P}$, let $\eta\colon cl(\varphi) \times \mathbb{N} \to \mathbb{B}_4$ be a mapping satisfying the compatibility constraints A1 to B2. Then, η is uniquely determined, and it is the φ-expansion of σ.*

Before we define the generalized Büchi automaton $\mathcal{A}_\varphi$ formally, let us sketch its construction. The states of $\mathcal{A}_\varphi$ are mappings $\mu\colon cl(\varphi) \to \mathbb{B}_4$, which encode the φ-expansion of a word σ in the sense that the sequence of states $\mu_0\mu_1\dots$ constituting an accepting run on σ satisfies $\mu_i(\psi) = \eta_i(\psi)$ for all $i \in \mathbb{N}$ and $\psi \in cl(\varphi)$. Clearly, the only states (i.e., mappings μ) of interest are those consistent with the local compatibility constraints A1 to A5.[5] Thus, in order to ease the following definition, we denote the set of such mappings by S (note that the cardinality of S is bounded by $|\mathbb{B}_4|^{|cl(\varphi)|} = 5^{|cl(\varphi)|}$).

When reading an input-word, the automaton $\mathcal{A}_\varphi$ uses its transitions to verify that its guess satisfies the local constraints and its acceptance condition to verify the non-local constraints. The latter is achieved by adding a Büchi condition for each of the Conditions B1 and B2, which translate the respective conditions in a straightforward manner. To capture the many-valued semantics of rLTL($\square, \diamondsuit$), we define the automaton without an initial state. Instead, we introduce states q_b for each $b \in \mathbb{B}_4$ with the property that $\mathcal{A}_\varphi$ accepts a word $\sigma \in \Sigma^\omega$ when starting in the state q_b if and only if $V(\sigma, \varphi) = b$. Thus, an accepting run starting in q_b signals that φ evaluates on σ to b. The formal definition of $\mathcal{A}_\varphi$ is as follows.

▶ **Definition 10** (Automaton $\mathcal{A}_\varphi$)**.** Let φ be an rLTL($\boxdot, \diamondsuit$) formula over the atomic propositions $\mathcal{P}$. Additionally, let $\Sigma = 2^\mathcal{P}$, $a \in \Sigma$, and S be the set of functions $\mu\colon cl(\varphi) \to \mathbb{B}_4$ that satisfy Conditions A1 to A5. We define the *generalized Büchi automaton* $\mathcal{A}_\varphi = (Q, \Sigma, \Delta, \mathcal{F})$ as follows:

- $Q = \{q_b \mid b \in \mathbb{B}_4\} \cup S$;
- the transition relation is defined by:
  - $(q_b, a, \mu) \in \Delta$ if and only if $\mu(\varphi) = b$ and $\mu(p) = \begin{cases} 1111 & \text{if } p \in a \cap \mathrm{cl}(\varphi); \text{ and} \\ 0000 & \text{if } p \in \mathrm{cl}(\varphi) \setminus a; \end{cases}$

[5] By this we mean that the conditions are satisfied if we substitute μ for η.

- $(\mu, a, \mu') \in \Delta$ if and only if the pair $(\mu,\ \mu')$ satisfies Conditions A6 and A7 as well as
$$\mu'(p) = \begin{cases} 1111 & \text{if } p \in a \cap \mathrm{cl}(\varphi); \text{ and} \\ 0000 & \text{if } p \in \mathrm{cl}(\varphi) \setminus a; \end{cases}$$

- $\mathcal{F}$ is the union of the following sets:
  - for each $\diamondsuit\psi \in cl(\varphi)$, we introduce for each $j \in \{1, \ldots, 4\}$ the set
$$F_{\diamondsuit\psi,j} = \{\mu \in S \mid \pi_j(\mu(\diamondsuit\psi)) = 0 \text{ or } \pi_j(\mu(\psi)) = 1\};$$
  - for each $\boxdot\psi \in cl(\varphi)$, we introduce the sets:
$$F_{\boxdot\psi,1} = \{\mu \in S \mid \pi_1(\mu(\boxdot\psi)) = 1 \text{ or } \pi_1(\mu(\psi)) = 0\}$$
$$F_{\boxdot\psi,2} = \{\mu \in S \mid \pi_2(\mu(\boxdot\psi)) = 0 \text{ or } \pi_1(\mu(\boxdot\psi)) = 1\}$$
$$F_{\boxdot\psi,3} = \{\mu \in S \mid \pi_3(\mu(\boxdot\psi)) = 1 \text{ or } \pi_4(\mu(\boxdot\psi)) = 0\}$$
$$F_{\boxdot\psi,4} = \{\mu \in S \mid \pi_4(\mu(\boxdot\psi)) = 0 \text{ or } \pi_4(\mu(\psi)) = 1\}$$

Definition 10 ensures that $\mathcal{A}_\varphi$ accepts $\sigma \in \Sigma^\omega$ if and only if there exists a run $q_b, \mu_0, \mu_1, \ldots$ on σ that visits each $F \in \mathcal{F}$ infinitely often. As an example, suppose that a run visits the set $F_{\diamondsuit\psi,1}$ for $\diamondsuit\psi \in cl(\varphi)$ infinitely often (i.e., $\pi_1(\mu_i(\diamondsuit\psi)) = 0$ or $\pi_1(\mu_i(\psi)) = 1$ holds for infinitely many $i \in \mathbb{N}$). This means that it never happens that from some $k \in \mathbb{N}$ onward both $\pi_1(\mu_k(\diamondsuit\psi)) = 1$ and $\pi_1(\mu_k(\psi)) = 0$. Hence, Condition B1 is fulfilled. Similarly, the remaining sets $F \in \mathcal{F}$ make sure that Conditions B1 and B2 are indeed satisfied. Moreover, the definition of Δ ensures that Conditions A1 to A7 are satisfied along an accepting run of $\mathcal{A}_\varphi$ on σ and, therefore, this run in fact forms the φ-expansion of σ (and is unique). Finally, by using different initial states, we make sure that $\mathcal{A}_\varphi$ accepts σ starting from q_b if only if $b = V(\sigma, \varphi)$ (since all outgoing transitions lead to states μ with $\mu(\varphi) = b$). As a consequence, we obtain the following result.

▶ **Theorem 11.** *Let φ be an rLTL($\boxdot, \diamondsuit$) formula over the set $\mathcal{P}$ of atomic propositions, $\Sigma = 2^{\mathcal{P}}$, and $b \in \mathbb{B}_4$. Then, $\mathcal{A}_\varphi$ accepts $\sigma \in \Sigma^\omega$ when starting in state q_b if and only if $V(\sigma, \varphi) = b$.*

For notational convenience, we denote the generalized Büchi automaton $\mathcal{A}_\varphi$ with initial state q_b by $\mathcal{A}^b_\varphi$. In addition, given a set $B \subseteq \mathbb{B}_4$, we denote the generalized Büchi automaton accepting the union $\bigcup_{b \in B} L(\mathcal{A}^b_\varphi)$ by $\mathcal{A}^B_\varphi$. The construction of $\mathcal{A}^B_\varphi$ is straightforward: first, we add a new state, say q_0, to $\mathcal{A}_\varphi$ and designate it as initial state; second, we add ε-transitions (q_0, ε, q_b) for each $b \in B$, which can subsequently be removed in the same manner as for finite automata with ε-transitions.

We finish this section with a remark about the size of the automata $\mathcal{A}_\varphi$ and $\mathcal{A}^B_\varphi$. Recalls that $\mathcal{A}_\varphi$'s state are (mainly) functions $\mu\colon cl(\varphi) \to \mathbb{B}_4$, of which there are $|\mathbb{B}_4|^{|\mathrm{cl}(\varphi)|} = 5^{|\mathrm{cl}(\varphi)|}$ many. Moreover, recall that we use four acceptance conditions per $\boxdot$ and $\diamondsuit$ operator.

▶ Remark. Both the automaton $\mathcal{A}_\varphi$ and $\mathcal{A}^B_\varphi$ have $\mathcal{O}(5^{|\mathrm{cl}(\varphi)|})$ states and at most $4 \cdot |\mathrm{cl}(\varphi)|$ acceptance sets.

4.3 Model Checking

Broadly speaking, the model checking problem asks whether the model of a given system exhibits a specified behavior (which is described as an rLTL($\boxdot, \diamondsuit$) formula in our case). Usually, a system is modeled as a Kripke structure, which is, for the sake of model checking, translated into a Büchi automaton whose language corresponds to the unraveling of the Kripke structure. For reasons of simplicity, we consider a system – more precisely, a model

thereof – already to be given as a (generalized) Büchi automaton. This leads to the following formulation of the model checking problem.

▶ **Problem 1** (Model checking). *Let φ be an rLTL($\boxdot, \diamondsuit$) formula over the set $\mathcal{P}$ of atomic propositions, let $\mathcal{A}$ be a generalized Büchi automaton over the alphabet $2^{\mathcal{P}}$, and let $B \subseteq \mathbb{B}_4$. Does $V(\sigma, \varphi) \in B$ hold for all $\sigma \in L(\mathcal{A})$?*

Our translation of rLTL($\boxdot, \diamondsuit$) formulas into a generalized Büchi automaton provides a straightforward means to answer the model checking problem: one simply constructs $\mathcal{A}_\varphi^B$ and checks $L(\mathcal{A}) \subseteq L(\mathcal{A}_\varphi^B)$. However, the naive attempt to check this inclusion (i.e., checking whether $L(\mathcal{A}) \cap (\Sigma^\omega \setminus L(\mathcal{A}_\varphi^B)) = \emptyset$ holds) would require to complement $\mathcal{A}_\varphi^B$, which we clearly want to avoid due to the inevitable exponential blowup; moreover, the equality $\Sigma^\omega \setminus L(\mathcal{A}_\varphi^B) = L(\mathcal{A}_{\neg\varphi}^B)$ does not hold in general. Instead, we exploit the fact that one can write the complement of $L(\mathcal{A}_\varphi^B)$ as $\Sigma^\omega \setminus L(\mathcal{A}_\varphi^B) = L(\mathcal{A}_\varphi^{B'})$ for $B' = \mathbb{B}_4 \setminus B$, which leads to the following result; a proof can be found in Appendix B.

▶ **Theorem 12.** *One can decide the model checking problem (Problem 1) for the Büchi automaton $\mathcal{A} = (Q, \Sigma, q_0, \Delta, \mathcal{F})$ and the rLTL($\boxdot, \diamondsuit$) formula φ in time $\mathcal{O}\big((|\mathcal{F}| + |\mathrm{cl}(\varphi)|) \cdot |Q| \cdot 5^{|\mathrm{cl}(\varphi)|}\big)$.*

It is worth mentioning that the corresponding optimization problem, namely to find the largest $b \in \mathbb{B}_4$ such that $V(\sigma, \varphi) \geq b$ holds for all $\sigma \in L(\mathcal{A})$, can be solved by repeatedly solving Problem 1 for decreasing values of b and, hence, falls into the complexity same class.

4.4 Synthesis of Reactive Systems

In the context of reactive synthesis, we consider infinite-duration two-player games over finite graphs with rLTL($\boxdot, \diamondsuit$) winning conditions. In particular, we show, given a game with rLTL($\boxdot, \diamondsuit$) winning condition, how to construct a finite-state winning strategy. Throughout this section, we assume familiarity with games over finite graphs and follow the definitions and notations by Grädel, Thomas, and Wilke [11].

We consider games of the following kind.

▶ **Definition 13** (rLTL($\boxdot, \diamondsuit$) games). Let $\mathcal{P}$ be a finite set of atomic propositions. An *rLTL($\boxdot, \diamondsuit$) game* is a pair $\mathfrak{G} = (\mathcal{G}, (\varphi, B))$ consisting of

- a finite, labeled *game graph* $\mathcal{G} = (V, E, \lambda)$ where V is a finite set of vertices that is partitioned into two disjoint sets $V_0, V_1 \subseteq V$, $E \subseteq V \times V$ is an edge relation, and $\lambda \colon V \to 2^{\mathcal{P}}$ is a function labeling each vertex with atomic propositions; and
- a pair (φ, B) consisting of an rLTL($\boxdot, \diamondsuit$) formula φ over $\mathcal{P}$ and a set $B \subseteq \mathbb{B}_4$ (this pair constitutes the *winning condition* as we formalize shortly).

An rLTL($\boxdot, \diamondsuit$) game is played as usual by two players, Player 0 and Player 1, who construct a *play* $\rho = v_0 v_1 \ldots \in V^\omega$ (i.e., an infinite sequence of vertices) by moving a token along the edges of the game graph. A play $\rho = v_0 v_1 \ldots$ induces an infinite word $\lambda(\rho) = \lambda(v_0)\lambda(v_1) \ldots \in (2^{\mathcal{P}})^\omega$, and the value of the formula φ on $\lambda(\rho)$ determines the winner of the play. More precisely, we call a play $\rho \in V^\omega$ *winning for Player 0* if $V(\lambda(\rho), \varphi) \in B$; symmetrically, we call a play *winning for Player 1* if it is *not winning* for Player 0. Finite-state strategies and winning strategies are defined as usual (see Grädel, Thomas, and Wilke [11] for further details).

Given an rLTL($\boxdot, \diamondsuit$) game and a vertex $v \in V$, we are interested in *solving the game* (i.e., in deciding which player has a winning strategy from v and in computing such a strategy), which is formalized next.

▶ **Problem 2** (Solving rLTL($\boxdot, \diamondsuit$) games). *Let an rLTL($\boxdot, \diamondsuit$) game $\mathfrak{G} = (\mathcal{G}, (\varphi, B))$ over the set V of vertices and a vertex $v_0 \in V$ be given.*

1. *Determine the player who has a winning strategy from vertex v_0.*
2. *Compute a winning strategy from vertex v_0.*

To solve this problem, we follow the Safra-based approach using the following four-step process. A detailed description of each step can be found in Appendix C.

1. We construct a nondeterministic Büchi automaton $\mathcal{B}^B_\varphi$ with $L(\mathcal{B}^B_\varphi) = \{\sigma \in (2^{\mathcal{P}})^\omega \mid V(\sigma, \varphi) \in B\}$.
2. We determinize $\mathcal{B}^B_\varphi$ using Safra's construction [20], resulting in a (deterministic) Rabin automaton $\mathcal{C}^B_\varphi$ that is language-equivalent to $\mathcal{B}^B_\varphi$.
3. We construct a Rabin game $\mathfrak{G}'$ by taking the product of the game graph $\mathcal{G}$ and the Rabin automaton $\mathcal{C}^B_\varphi$.
4. We apply standard techniques to solve $\mathfrak{G}'$, which allows us to decide which player has a winning strategy from v and to construct a winning strategy for the corresponding player.

Using this process, we obtain the following results (the complexity results immediately follow from the size of $\mathcal{A}_\varphi$, the subsequent Safra construction, and the standard technique to solve the resulting Rabin game).

▶ **Theorem 14.** *Given an rLTL($\boxdot, \diamondsuit$) game $\mathfrak{G} = (\mathcal{G}, (\varphi, B))$ with $\mathcal{G} = (V, E, \lambda)$ and a vertex $v_0 \in V$, one can*

1. *decide which player has a winning strategy from v_0 (i.e., Problem 2 Part 1) and*
2. *compute a winning strategy for the corresponding player (i.e., Problem 2 Part 2)*

in time $\mathcal{O}(n^{k+3}kk!)$ where $n = |V| \cdot 2^{5^{c_0|\mathrm{cl}(\varphi)|}}$, $k = 5^{c_1|\mathrm{cl}(\varphi)|}$, and c_0, c_1 are suitable constants.

5 Quality is Dual to Robustness

We motivated rLTL($\boxdot, \diamondsuit$) by the need to distinguish between the different ways in which safety properties can be violated. One can take a dual view and seek to distinguish between the different ways in which guarantee properties are satisfied. To illustrate this point, consider the LTL formula $\diamondsuit p \Rightarrow \diamondsuit q$ where $\diamondsuit p$ is an environment assumption and $\diamondsuit q$ is a system guarantee. According to the motto *more is better* we would prefer the system to guarantee the stronger property $\square\diamondsuit q$ whenever the environment satisfies the stronger property $\square\diamondsuit p$. By now, the reader can already complete our argument: $\diamondsuit\square p$ should lead to $\diamondsuit\square q$ and $\square p$ should lead to $\square q$. Formalizing these ideas would still take us to a 5-valued logic where, however, negation needs to be defined differently. Although we can still use the linear order

$$0000 \prec 0001 \prec 0011 \prec 0111 \prec 1111$$

on the set of truth values, one now needs to interpret the values differently. The value 0000 still corresponds to *false* but the remaining truth values now correspond to different quality values for *true* with 0001 being the lowest quality and 1111 the highest. Negation should then take 0000 to 1111 and all the remaining truth values to 0000. Such negation is no more than the intuitionistic negation already discussed in Section 3.3 and would equip $\mathbb{B}_4$ with the structure of an Heyting algebra instead of the da Costa algebra used in this paper. This observation justifies the title of this section and suggests the following question: is there an extension of LTL that can be used to reason about *both* robustness and quality? This is a question we leave for further research.

6 Discussion

The logic rLTL offers a transparent way to reason about the robustness of LTL specifications. Given an LTL formula φ, one obtains the corresponding rLTL formula ψ simply by dotting the temporal operators in φ. The technical development of the semantics was based on the insight that the temporal operators $\Box$ and $\Diamond$ count how often the formula they are applied to is satisfied thereby leading to a 5-valued logic. We studied the verification and synthesis problems for rLTL and showed they can be solved in exponential and doubly exponential time, respectively. These complexity bounds are the same as those for LTL once we replace 2, since LTL is Boolean valued, with 5, since rLTL is 5-valued. It remains an open problem to determine if these complexity upper bounds are tight. In addition to this question, we sketched in Section 5 a variant of rLTL tailored to quality and raised the question of how to combine robustness and quality in a single logic.

References

1 Shaull Almagor and Orna Kupferman. Latticed-ltl synthesis in the presence of noisy inputs. In *FOSSACS 2014*, volume 8412 of *LNCS*, pages 226–241. Springer, 2014.

2 Rajeev Alur, Aditya Kanade, and Gera Weiss. Ranking automata and games for prioritized requirements. In *CAV 2008*, volume 5123 of *LNCS*, pages 240–253. Springer, 2008.

3 Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008.

4 Roderick Bloem, Krishnendu Chatterjee, Karin Greimel, Thomas A. Henzinger, and Barbara Jobstmann. Robustness in the presence of liveness. In *CAV 2010*, volume 6174 of *LNCS*, pages 410–424. Springer, 2010.

5 Roderick Bloem, Karin Greimel, Thomas A. Henzinger, and Barbara Jobstmann. Synthesizing robust systems. In *FMCAD 2009*, pages 85–92. IEEE, 2009.

6 Swarat Chaudhuri, Sumit Gulwani, and Roberto Lublinerman. Continuity analysis of programs. In *POPL 2010*, pages 57–70. ACM, 2010.

7 Alexandre Donzé and Oded Maler. Robust satisfaction of temporal logic over real-valued signals. In *FORMATS 2010*, volume 6246 of *LNCS*, pages 92–106. Springer, 2010.

8 Laurent Doyen, Thomas A. Henzinger, Axel Legay, and Dejan Nickovic. Robustness of sequential circuits. In *ACSD 2010*, pages 77–84. IEEE, 2010.

9 Rüdiger Ehlers and Ufuk Topcu. Resilience to intermittent assumption violations in reactive synthesis. In *HSCC 2014*, pages 203–212. ACM, 2014.

10 Georgios E. Fainekos and George J. Pappas. Robustness of temporal logic specifications for continuous-time signals. *Theor. Comput. Sci.*, 410(42):4262–4291, 2009.

11 Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *LNCS*. Springer, 2002.

12 B. Jonsson and A. Tarski. Boolean algebras with operators. Part I. *American Journal of Mathematics*, 73(4):891–939, 1951.

13 Orna Kupferman and Yoad Lustig. Lattice automata. In *VMCAI 2007*, volume 4349 of *LNCS*, pages 199–213. Springer, 2007.

14 Rupak Majumdar and Indranil Saha. Symbolic robustness analysis. In *RTSS 2009*, pages 355–363. IEEE, 2009.

15 Doron Peled and Thomas Wilke. Stutter-invariant temporal properties are expressible without the next-time operator. *Information Processing Letters*, 63(5):243–246, 1997.

16 Dominique Perrin and Jean-Eric Pin. *Infinite Words*, volume 141 of *Pure and Applied Mathematics*. Elsevier, 2004.

17 Nir Piterman and Amir Pnueli. Faster solutions of rabin and streett games. In *LICS 2006*, pages 275–284. IEEE, 2006.

18 G. Priest. Dualising Intuitionist Logic. *Principia*, 13(2):165–184, 2009.

19 Aurélien Rizk, Grégory Batt, François Fages, and Sylvain Soliman. A general computational method for robustness analysis with applications to synthetic gene networks. *Bioinformatics*, 25(12), 2009. doi:10.1093/bioinformatics/btp200.

20 Shmuel Safra. On the complexity of omega-automata. In *FOCS 1988*, pages 319–327. IEEE, 1988.

21 P. Tabuada and D. Neider. Robust Linear Temporal Logic, 2015. arXiv:1510.08970.

22 Paulo Tabuada, Sina Yamac Caliskan, Matthias Rungger, and Rupak Majumdar. Towards robustness for cyber-physical systems. *IEEE Trans. Automat. Contr.*, 59(12):3151–3163, 2014.

23 Milan Češka, David Šafránek, Sven Dražan, and Luboš Brim. Robustness analysis of stochastic biochemical systems. *PLoS ONE*, 9(4), 2014. doi:10.1371/journal.pone.0094553.

A φ-Expansion

Proof of Lemma 9. To prove the lemma, we need to establish that $V(\sigma_{i..}, \psi) = \eta(\psi, i)$ holds for all $\psi \in cl(\varphi)$ and $i \in \mathbb{N}$. The proof proceeds by structural induction over the subformulas of φ.

Base case. In the case of atomic propositions, the claim holds by definition of V.

Induction step. In the case of the operators $\neg$, $\vee$, $\wedge$, and $\Rightarrow$, the claim follows immediately from applying the induction hypothesis and by definition of V.

In the case of $\psi = \diamondsuit\psi_1$, a straightforward induction that applies (a) Condition A6, (b) the expansion rule for $\diamondsuit$ (see Proposition 6, Equation (22)), and (c) the induction hypothesis for ψ_1 (i.e., $V(\sigma_{i..}, \psi_1) = \eta(\psi_1, i)$ for all $i \in \mathbb{N}$) shows that the following is true for each $j \in \{1, \ldots, 4\}$: if $\eta_j(\psi_1, k) = 1$ for a $k \in \mathbb{N}$, then $\eta_j(\psi, \ell) = 1$ and, hence, $V_j(\sigma_{\ell..}, \psi) = \eta_j(\psi, \ell)$ for all $\ell \leq k$. Therefore, if infinitely many k with $\eta_j(\psi_1, k) = 1$ exist, then $V_j(\sigma_{i..}, \psi) = \eta_j(\psi, i)$ for all $i \in \mathbb{N}$. If this is not the case, then there exists a $k \in \mathbb{N}$ such that $\eta_j(\psi_1, \ell) = 0$ for all $\ell \geq k$. Then, Condition B1 asserts for all $\ell \geq k$ that $\eta_j(\psi, \ell) = 0$ and, hence, $V_j(\sigma_{\ell..}, \psi) = \eta_j(\psi, \ell)$ is satisfied by the semantics of $\diamondsuit$ and the induction hypothesis for ψ_1; this, in turn, implies $V_j(\sigma_{i..}, \psi) = \eta_j(\psi, i)$ for all $i \in \mathbb{N}$. These arguments are true for all $j \in \{1, \ldots, 4\}$ and, therefore, $V(\sigma_{i..}, \psi) = \eta(\psi, i)$ holds for all $i \in \mathbb{N}$.

The case $\psi = \boxdot\psi_1$ can be proven using similar arguments as in the case of the $\diamondsuit$ operator, but the semantics of $\boxdot$ requires to split the proof into four parts and prove $V_j(\sigma_{i..}, \psi) = \eta_j(\psi, i)$ individually for each $j \in \{1, \ldots, 4\}$. So as not to clutter this proof too much, we provide a detailed proof for $j = 1$ and skip the remaining. However, it is important to note that the claim needs to be proven first for $j = 1$ and $j = 4$ since the proofs for $j = 2$ and $j = 3$ rely thereon (the expansion rules recur on $V_1(\sigma_{i..}, \psi)$ and $V_4(\sigma_{i..}, \psi)$, respectively).

To prove $V_1(\sigma_{i..}, \psi) = \eta_1(\psi, i)$ for all $i \in \mathbb{N}$, we first observe that $\eta_1(\psi_1, k) = 0$ for a $k \in \mathbb{N}$ implies $V_1(\sigma_{\ell..}, \psi) = \eta_1(\psi, \ell)$ for all $\ell \leq k$; analogous to the case of the operator $\diamondsuit$, an induction using Condition A7(a), the expansion rule for $\boxdot$ (see Proposition 6, Formula (19)), and the induction hypothesis for ψ_1 establishes this. Hence, if infinitely many k with $\eta_1(\psi_1, k) = 0$ exist, then $V_1(\sigma_{i..}, \psi) = \eta_1(\psi, i)$ for all $i \in \mathbb{N}$. If this is not the case, then there exists a $k \in \mathbb{N}$ such that $\eta_1(\psi_1, \ell) = 1$ for all $\ell \geq k$. Then, Condition B2(a) asserts for all $\ell \geq k$ that $\eta_1(\psi, \ell)) = 1$ and, hence, $V_1(\sigma_{\ell..}, \psi) = \eta_1(\psi, \ell)$ holds by the

semantics of $\boxdot$ and the induction hypothesis of ψ_1. This implies $V_1(\sigma_{i..}, \psi) = \eta_1(\psi, i)$ for all $i \in \mathbb{N}$.

As mentioned above, the case $j = 4$ and the subsequent cases $j = 2$ and $j = 3$ are analogous. ◀

B Model Checking

Proof of Theorem 12. Let φ be an rLTL($\boxdot, \diamondsuit$) formula over the atomic propositions $\mathcal{P}$, $\mathcal{A} = (Q, \Sigma, q_0, \Delta, \mathcal{F})$ a generalized Büchi automaton over the alphabet $2^{\mathcal{P}}$, and $B \subseteq \mathbb{B}_4$.

First, let $B' = \mathbb{B}_4 \setminus B$. Then, it is not hard to verify that

$$\begin{aligned} V(\sigma, \varphi) \in B \text{ for all } \sigma \in L(\mathcal{A}) &\Leftrightarrow L(\mathcal{A}) \subseteq L(\mathcal{A}_\varphi^B) \\ &\Leftrightarrow L(\mathcal{A}) \cap \left(\Sigma^\omega \setminus L(\mathcal{A}_\varphi^B)\right) = \emptyset \\ &\Leftrightarrow L(\mathcal{A}) \cap \mathcal{A}_\varphi^{B'} = \emptyset. \end{aligned}$$

Recall that $\mathcal{A}_\varphi^{B'}$ has $5^{|\mathrm{cl}(\varphi)|} + 5$ states and also at most $4 \cdot |\mathrm{cl}(\varphi)|$ acceptance sets.

Second, given two generalized Büchi automata $\mathcal{A}_1 = (Q_1, \Sigma, q_0^1, \Delta_1, \mathcal{F}_1)$ and $\mathcal{A}_2 = (Q_2, \Sigma, q_0^2, \Delta_2, \mathcal{F}_2)$, it is well-known that one can construct a generalized Büchi automaton accepting $L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$ using a simple product construction (see, e.g., [16]). This construction results in an automaton with $|Q_1| \cdot |Q_2|$ states and $|\mathcal{F}_1| + |\mathcal{F}_2|$ acceptance sets. Since $\mathcal{A}_\varphi^{B'}$ consists of $5^{|\mathrm{cl}(\varphi)|} + 5$ states and has at most $4 \cdot |\mathrm{cl}(\varphi)|$ acceptance sets, this implies that one can construct a generalized Büchi automaton $\mathcal{B}$ with $L(\mathcal{B}) = L(\mathcal{A}) \cap L(\mathcal{A}_\varphi^{B'})$ consisting of $|Q| \cdot (5^{|\mathrm{cl}\varphi|} + 5)$ states and at most $|\mathcal{F}| + 4 \cdot |\mathrm{cl}(\varphi)|$ acceptance sets.

Finally, it is left to check whether $L(\mathcal{B}) = \emptyset$. This problem is fundamental in LTL model checking, and there exist efficient algorithms that solve this problem in time linear in the product of the number of states of the input automaton and the number of its acceptance sets (see, e.g., [3]). Hence, one can solve Problem 1 in $\mathcal{O}\big((|\mathcal{F}| + |\mathrm{cl}(\varphi)|) \cdot |Q| \cdot 5^{|\mathrm{cl}(\varphi)|}\big)$ time. ◀

C Computing Strategies in rLTL($\boxdot, \diamondsuit$)-Games

We compute a winning strategy in an rLTL($\boxdot, \diamondsuit$)-game using a four-step process:

1. First, we construct the generalized Büchi automaton $\mathcal{A}_\varphi^B$; recall that this automaton comprises $5^{|\mathrm{cl}(\varphi)|} + 5$ states and at most $4 \cdot |\mathrm{cl}(\varphi)|$ acceptance sets. Subsequently, we construct a nondeterministic Büchi automaton $\mathcal{B}_\varphi^B$ accepting the same language; the standard conversion results in a Büchi automaton that comprises $\mathcal{O}(4 \cdot |\mathrm{cl}(\varphi)| \cdot (5^{|\mathrm{cl}(\varphi)|} + 5))$ states.
2. Using Safra's determinization procedure [20], we obtain a (deterministic) Rabin automaton[6] $\mathcal{C}_\varphi^B$ that is language-equivalent to $\mathcal{B}_\varphi^B$. The automaton $\mathcal{C}_\varphi^B$ has $2^{5^{c_0 |\mathrm{cl}(\varphi)|}}$ states and $5^{c_1 \cdot |\mathrm{cl}(\varphi)|}$ Rabin pairs where $c_0 > c_1$ are suitable constants.
3. Next, we construct a Rabin game[7] as follows. We first construct the (unlabeled) product game graph $\mathcal{G}' = (V', E')$ of the game graph $\mathcal{G} = (V, E, \lambda)$ and the Rabin automaton

[6] A Rabin automaton is a tuple $\mathcal{C} = (Q, \Sigma, q_0, \delta, \Omega)$ where Q, Σ, and q_0 are as in Büchi automata, $\delta \colon Q \times \Sigma \to Q$ is a (deterministic) transition function, and $\Omega \subseteq 2^Q \times 2^Q$ is the acceptance condition. The *run* of a Rabin automaton on a word $\sigma \in \Sigma^\omega$ is an infinite sequence of states $\rho = q_0 q_1 \ldots$ satisfying $\delta(q_i, \sigma(i)) = q_{i+1}$ for all $i \in \mathbb{N}$. A run ρ is called *accepting* if there exists a pair $(E, F) \in \Omega$ such that $E \cap \mathrm{Inf}(\rho) = \emptyset$ and $F \cap \mathrm{Inf}(\rho) \neq \emptyset$.

[7] A Rabin game is a game played over an unlabeled game graph $\mathcal{G} = (V, E)$ with nonempty, finite set V of vertices and directed edge relation $E \subseteq V \times V$. The winning condition of a Rabin game is a set $\Omega \subseteq 2^V \times 2^V$, and a play $\rho = v_0 v_1 \ldots \in V^\omega$ is said to be winning for Player 0 if there exists a

$\mathcal{C}^B_\varphi = (Q, 2^{\mathcal{P}}, q_0, \delta, \Omega)$ such that $V' = V \times Q$ and

$$\big((v,q),(v',q')\big) \in E' \Leftrightarrow (v,v') \in E \text{ and } \delta(q,\lambda(v)) = q'.$$

Then, we define the Rabin winning condition of $\mathfrak{G}'$ to be

$$\Omega' = \big\{((V,E),(V,F)) \in V' \times V' \mid (E,F) \in \Omega\big\}.$$

The desired Rabin game is then $\mathfrak{G}' = (\mathcal{G}', \Omega')$.
An induction over the length of plays in $\mathfrak{G}'$ shows that Player 0 wins a play $\rho' = (v_0,q_0)(v_1,q_1)\ldots$ if and only if Player 0 wins the play $\rho = v_0 v_1 \ldots$ in $\mathfrak{G}$.

4. Finally, by applying the method by Piterman and Pnueli [17], we solve the resulting Rabin game in time $\mathcal{O}(n^{k+3}kk!)$ where $n = |V| \cdot 2^{5^{c_0|\mathrm{cl}(\varphi)|}}$ is the number of vertices and $k = 5^{c_1|\mathrm{cl}(\varphi)|}$ is the number of Rabin pairs of $\mathfrak{G}'$.

pair $(E,F) \in \Omega$ such that $E \cap \mathrm{Inf}(\rho) = \emptyset$ and $F \cap \mathrm{Inf}(\rho) \neq \emptyset$; by slight abuse of notation, $\mathrm{Inf}(\rho)$ here corresponds to the set of all vertices occurring infinitely often in the play ρ.