2018 IEEE Conference on Decision and Control (CDC)
Miami Beach, FL, USA, Dec. 17-19, 2018

Verifying rLTL formulas: now faster than ever before!

Tzanis Anevlavis, Matthew Philippe, Daniel Neider and Paulo Tabuada

***Abstract*— Robust Linear Temporal Logic (rLTL) was crafted to incorporate the notion of robustness into Linear-time Temporal Logic specifications. Robustness is ubiquitous in control systems and translates the intuitive notion that "small" violations of environment assumptions should only lead to "small" violations of system guarantees. This notion was formalized in the logic rLTL via 5 different truth values and it led to an increase in the time complexity of the associated model checking problem. In this paper we identify and analyze a fragment of rLTL for which the model checking problem can be solved using generalized Büchi automata with at most $3^{|\varphi|}$ states where $|\varphi|$ denotes the length of an rLTL formula φ. This is a substantial improvement over the previously known bound of $5^{|\varphi|}$ and close to the tight upper bound $2^{|\varphi|}$ for LTL.**

I. Introduction

Robustness is widely recognized in the control community as an essential ingredient of any feedback control loop. However, as we move from control problems that can be described by differential equations to control problems where software plays a more dominant role, such as in Cyber-Physical Systems (CPS), identifying the "correct" notion of robustness has remained a challenge [3], [14], [16]. Steps in this direction were given in [8], [9, Chapter 7], by using models that combine continuous with more discrete behavior. A different approach was given in [4]–[6], [12] by introducing a logic enabling reasoning over real-valued signals.

Two of the authors proposed an alternative in [17] where the notion of Input-to-State Stability (ISS) inspired a new semantics for Linear-time Temporal Logic (LTL) resulting in a new logic termed *robust Linear-time Temporal Logic* (rLTL). To understand how ISS was at the genesis of rLTL, consider the system $\dot{x} = f(x,u)$ that we assume to be ISS and let us take the initial condition to be $x(0) = 0$. The assumption that the disturbance u is always equal to zero implies that $x(t) = 0$ for all $t \in \mathbb{R}_0^+$:

$$u(t) = 0 \Rightarrow x(t) = 0.$$

Robustness comes into play when we weaken the assumption to $\lim_{t\to\infty} u(t) = 0$. In this case we can no longer ensure that $x(t) = 0$ for all $t \in \mathbb{R}_0^+$ but we still have its weakened version $\lim_{t\to\infty} x(t) = 0$:

$$\lim_{t\to\infty} u(t) = 0 \Rightarrow \lim_{t\to\infty} x(t) = 0.$$

This simple observation, that weakening the assumptions (antecedent of the implication) leads to a weakened version of the guarantees (consequent of the implication), does not hold in LTL. The LTL semantics renders:

$$\varphi \Rightarrow \psi,$$

semantically equivalent to:

$$\neg\varphi \vee \psi.$$

When φ does not hold, nothing can be said about the truth value of ψ. Not even if "φ is close to being satisfied".

To encapsulate this, rLTL adopts a 5-valued semantics: the truth value of an rLTL formula is interpreted as corresponding to *true* or to different shades of *false*. For example, the LTL formula $\Box p$ is true if p occurs at every time step and false otherwise. The robust version of the always operator, $\boxdot$, is *five valued* and its truth value is:

1) 1111 if p holds at every time step, i.e., the LTL formula $\Box p$ holds.
2) 0111 if p is violated only finitely many times, i.e., the LTL formula $\Diamond \Box p$ holds.
3) 0011 if p is both satisfied and violated infinitely many times, i.e., the LTL formula $\Box \Diamond p$ holds.
4) 0001 if p holds at most finitely many times, i.e., the LTL formula $\Diamond p$ holds.
5) 0000 if p is always violated, i.e., the LTL formula $\Box\neg p$ holds.

As illustrated with $\boxdot p$, 1111 corresponds to true and the remaining truth values correspond to different shades of false. The truth values are ordered:

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

with higher truth values being closer to 1111. Robustness now enters the picture via the rLTL semantics for implication that only provides the truth value 1111 for the rLTL formula $\boxdot p \Rrightarrow \boxdot q$ when $\Box p$ implies $\Box q$, *and* weakening the assumption $\Box p$ to $\Diamond \Box p$ implies the guarantee $\Diamond \Box q$, *and* weakening $\Diamond \Box p$ to $\Box \Diamond p$ implies $\Box \Diamond q$, *and* weakening $\Box \Diamond p$ to $\Diamond p$ implies $\Diamond q$.

With an increase in the number of truth values comes an increase in the complexity of verifying rLTL specifications. More precisely, verifying an LTL formula φ requires the construction of a Generalized Büchi Automaton (GBA) with $\mathcal{O}(2^{|\varphi|})$ states, where $|\varphi|$ is the

This work was partially supported by the NSF grant 1645824 and by the Army Research Laboratory under Cooperative Agreement W911NF-17-2-0196.

Tzanis Anevlavis, Matthew Philippe and Paulo Tabuada are with the UCLA Electrical and Computer Engineering Department, Los Angeles, CA 90095 `{janis10, matphilippe, tabuada}@ucla.edu`

Daniel Neider is with the Max Planck Institute for Software Systems, Kaiserslautern, Germany `neider@mpi-sws.org`

number of subformulas in φ. However, an extension of the technique to verifying rLTL formulas, see [17, Theorem 4.9], relies on constructing a GBA with $\mathcal{O}(5^{|\varphi|})$ states.

In this paper we aim at refining this complexity upper bound for the rLTL verification problem. To do so, we make use of *temporal testers* [10], [13] to efficiently construct GBAs for model checking.

In particular, our main result states that for the rLTL fragment[1] of formulas of the form $\psi_1 \Rrightarrow \psi_2$, where ψ_1 and ψ_2 are rLTL formulas not containing robust implications or robust releases, the number of states of the testers involved in rLTL model checking is upper bounded by

$$2^{|\varphi|-k(\varphi)}3^{k(\varphi)},$$

where $|\varphi|$ is the number of subformulas in φ, and $k(\varphi)$ is the number of *robust always* ($\boxdot$) operators it contains. In the worst case, $k(\varphi) = |\varphi|$ and the time complexity is proportional to $3^{|\varphi|}$, a considerable improvement over the bound $5^{|\varphi|}$ proved in [17] and closer to the tight LTL bound $2^{|\varphi|}$.

The structure of the paper is as follows. In Section II, we introduce concepts relevant to LTL model checking and in Section II-B we compute tight bounds on the size of temporal testers for LTL formulas. Section III introduces the syntax and semantics of rLTL before utilizing the results of Section II-B to compute complexity bounds for rLTL model checking in Section III-A. Due to space limitations, the proofs to our theorems are omitted.

II. LTL Model Checking

In this section we describe the syntax and semantics of LTL, as well as the concept of *temporal testers* used in LTL model checking.

A. LTL miscellanea

Definition 1 (LTL Syntax): Let $\mathcal{P}$ be a nonempty, finite set of atomic propositions. The set of all LTL formulas on $\mathcal{P}$, written $\mathrm{LTL}(\mathcal{P})$, is the smallest set satisfying:

- $\mathcal{P} \subset \mathrm{LTL}(\mathcal{P})$ and,
- if φ and ψ are elements of $\mathrm{LTL}(\mathcal{P})$, then $\neg\varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi$, $\varphi \Rightarrow \psi$, $\bigcirc\varphi$, $\Box\varphi$, $\Diamond\varphi$, $\varphi \mathcal{R} \psi$ and $\varphi \mathcal{U} \psi$ are elements of $\mathrm{LTL}(\mathcal{P})$ as well.

The length of a formula $\varphi \in \mathrm{LTL}(\mathcal{P})$, denoted by $|\varphi|$, is the number of subformulas it contains.

Given a set of atomic propositions $\mathcal{P}$, $\left(2^{\mathcal{P}}\right)^{\omega}$ is the set of all infinite words $\sigma = \sigma_0, \sigma_1, \ldots$ with $\sigma_i \subseteq \mathcal{P}$. For such a word, we let $\sigma_{i,\ldots}$ be the infinite word $\sigma_i, \sigma_{i+1}, \ldots$.

Definition 2 (LTL Semantics): The *LTL semantics* is a mapping:

$$W : \left(2^{\mathcal{P}}\right)^{\omega} \times \mathrm{LTL}(\mathcal{P}) \to \{0, 1\},$$

defined inductively for $p \in \mathcal{P}$ and $\varphi, \psi \in \mathrm{LTL}(\mathcal{P})$ as follows:

- $W(\sigma, p) = \begin{cases} 0 & \text{if } p \notin \sigma(0), \\ 1 & \text{if } p \in \sigma(0). \end{cases}$
- $W(\sigma, \neg\varphi) = 1 - W(\sigma, \varphi)$.
- $W(\sigma, \varphi \vee \psi) = \max\{W(\sigma, \varphi), W(\sigma, \psi)\}$.
- $W(\sigma, \varphi \wedge \psi) = \min\{W(\sigma, \varphi), W(\sigma, \psi)\}$.
- $W(\sigma, \bigcirc\varphi) = W(\sigma_{1..}, \varphi)$.
- $W(\sigma, \Box\varphi) = \inf_{i \ge 0} W(\sigma_{i..}, \varphi)$.
- $W(\sigma, \Diamond\varphi) = \sup_{i \ge 0} W(\sigma_{i..}, \varphi)$.
- $W(\sigma, \varphi \mathcal{U} \psi) = \sup_{j \ge 0} \min \left\{ \begin{array}{l} W(\sigma_{j..}, \psi), \\ \inf\limits_{0 \le i < j} W(\sigma_{i..}, \varphi) \end{array} \right\}$.
- $W(\sigma, \varphi \mathcal{R} \psi) = \inf_{j \ge 0} \max \left\{ \begin{array}{l} W(\sigma_{j..}, \psi), \\ \sup\limits_{0 \le i < j} W(\sigma_{i..}, \varphi) \end{array} \right\}$.

LTL Model Checking is a fundamental problem [2], [10], [11], [13], [15] in verification. Given a *model* of a system, represented as a finite state machine, the question is to decide whether or not all possible executions of the machine satisfy an LTL specification.
We describe these models using *Generalized Büchi Automata* (GBA), see e.g. [7, Section 3], [17, Definition 4.2].

Problem 1 (LTL model checking): Given a set of atomic propositions $\mathcal{P}$, a set of words $\mathcal{L} \subseteq \left(2^{\mathcal{P}}\right)^{\omega}$ recognized by a GBA $\mathcal{A}_{\mathcal{L}}$, and $\varphi \in \mathrm{LTL}(\mathcal{P})$, compute $\min_{\sigma \in \mathcal{L}} W(\sigma, \varphi)$.

Remark 1: The standard procedure for model checking an LTL formula φ is as follows (see [1, Section 5.2]). Given a GBA $\mathcal{A}_{\mathcal{L}}$ recognizing $\mathcal{L}$, we construct a GBA $\mathcal{A}_{\neg\varphi}$ recognizing the words satisfying the formula $\neg\varphi \in \mathrm{LTL}(\mathcal{P})$. Let $N_{\neg\varphi}^{\mathrm{LTL}}$, $F_{\neg\varphi}^{\mathrm{LTL}}$ be respectively the number of states and terminal conditions of that GBA.
Then composing $\mathcal{A}_{\mathcal{L}}$ with $\mathcal{A}_{\neg\varphi}$ we obtain GBA $\mathcal{A}_{\mathcal{L},\neg\varphi}$, which recognizes *all the words of* $\mathcal{L}$ *that do not satisfy the formula* φ. Finally, we check the *emptiness* of $\mathcal{A}_{\mathcal{L},\neg\varphi}$: if the language recognized by $\mathcal{A}_{\mathcal{L},\neg\varphi}$ is empty, then $\mathcal{L}$ satisfies φ.
If $N_{\mathcal{L},\neg\varphi}$ is the number of states of $\mathcal{A}_{L,\neg\varphi}$ and $M_{\mathcal{L},\neg\varphi}$ the number of transitions, the complexity of this step is

$$\mathcal{O}(N_{\mathcal{L},\neg\varphi} + M_{\mathcal{L},\neg\varphi}), \tag{1}$$

where $N_{\mathcal{L},\neg\varphi} = \mathcal{O}(F_{\neg\varphi}^{\mathrm{LTL}} N_{\neg\varphi}^{\mathrm{LTL}})$, and $M_{\mathcal{L},\neg\varphi} = \mathcal{O}(N_{\mathcal{L},\neg\varphi}^2)$.

This motivates the task of trying to construct a GBA $\mathcal{A}_{\neg\varphi}$ with the smallest possible number of states and accepting conditions for rLTL. In this regard, classical (and tight) bounds for LTL are

$$N_{\neg\varphi}^{\mathrm{LTL}} = \mathcal{O}(2^{|\varphi|}),\ F_{\neg\varphi}^{\mathrm{LTL}} = \mathcal{O}(|\varphi|). \tag{2}$$

Hence, naturally we aim to approach the LTL bounds as much as possible for the rLTL model checking problem.

B. Temporal Testers

Temporal Testers [10], [13] are discrete transition systems equipped with justice conditions that can be used to obtain automata recognizing infinite words satisfying an LTL formula $\varphi \in \mathrm{LTL}(\mathcal{P})$ by composing testers recognizing its subformulas.

[1] We found that more than 60% of the rLTL versions of the LTL formulas available in [18] were in this fragment. In addition, our fragment includes the GR(1) fragment for which efficient verification and synthesis algorithms are known. Hence, we are dealing with a practically significant and powerful fragment.

In this subsection, we study elementary temporal testers arising in the study of rLTL formulas. We show that for any $\varphi \in \mathrm{LTL}(\mathcal{P})$, there exists a tester for $\Diamond\Box\varphi$ with at most *3 times the number of states* of the tester for φ. The operation $\Diamond\Box$ is central in the rLTL semantics, and this result allows us to provide tight bounds for the complexity of rLTL model checking in Section III.

A formal definition of a temporal tester, relying on *Just Discrete Systems*, can be found in [13]. For the sake of brevity and clarity, we provide a less formal, but more direct and intuitive, definition below.

Definition 3: A temporal tester for an LTL formula $\varphi \in \mathrm{LTL}(\mathcal{P})$ is a tuple $T(\varphi) = (V, \Theta, R, \mathcal{J})$ where

- $V = \{x_\psi \mid \psi$ is a subformula of $\varphi\}$ is a set of Boolean variables. The set of *states* of the tester is Σ and each $s \in \Sigma$ is an *assignment of the variables* V to either 1 or 0, i.e., $s(x) \in \Sigma \subseteq \{0,1\}^V$.
- Θ is an assertion over V, i.e., it defines a subset of Σ. The states $s \in \Sigma$ satisfying this assertion are *initial states*.
- R is an assertion over $V \times V$, i.e., it defines a subset of $\Sigma \times \Sigma$. If a pair of states $s_1, s_2 \in \Sigma$ satisfies the assertion, we say that s_2 is a successor of s_1.
- $\mathcal{J}$ is a set of justice requirements, where each $J \in \mathcal{J}$ is an assertion on V, i.e., it defines a subset of Σ.

A *computation* of a tester is an infinite sequence of states $s = s_0, s_1, \ldots$, such that s_0 is an initial state, (s_i, s_{i+1}) satisfies R for $i \geq 0$, and for every $J \in \mathcal{J}$, s contains infinitely many states satisfying J.

Remark 2 (Link with Generalized Büchi Automata): As seen in Remark 1, most of the analysis of LTL model checking relies on GBAs. The testers described here are closely related to GBAs.

For any tester $T(\varphi)$, there is a GBA such that:

- Its runs correspond to the tester's computations.
- Its states are those of the tester. There are at most $2^{|\varphi|}$ states, but there could be less. The initial states of the GBA are those of the tester. Each $J \in \mathcal{J}$ defines an accepting condition for the GBA.
- There is a transition in the GBA between states s and s' if the pair (s, s') satisfies R. The label on that transition is the set $\{p \in \mathcal{P} \mid s'(x_p) = 1\}$, where $s'(x_p)$ is the value assigned to p by the state s'.

A tester allows to detect if any computation satisfies φ or $\neg\varphi$.

To obtain the GBA $\mathcal{A}_\varphi$ recognizing the words satisfying φ from $T(\varphi)$, it suffices to remove the states of the tester where $x_\varphi = 0$ from the set of initial conditions Θ.

Example 1 (The Until Tester): The definition of the tester for $\varphi\,\mathcal{U}\,\psi$, where φ and ψ are LTL formulas, is as follows (see [13, Section 6.2]):

$$T(\varphi\,\mathcal{U}\,\psi) : \begin{cases} V : Vars(\varphi,\psi) \cup \{x_{\varphi\mathcal{U}\psi}\}, \\ \Theta : 1, \\ R : x_{\varphi\mathcal{U}\psi} = [x_\psi \vee (x_\varphi \wedge x'_{\varphi\mathcal{U}\psi})], \\ \mathcal{J} : \{\neg x_{\varphi\mathcal{U}\psi} \vee x_\psi\}. \end{cases} \quad (3)$$

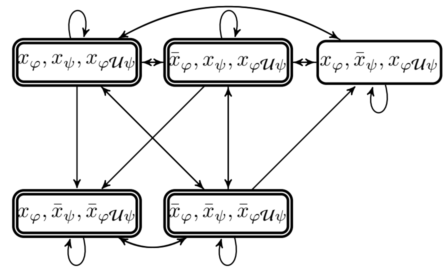

Fig. 1. Tester $T(\varphi\,\mathcal{U}\,\psi)$. We list in each state the variables set to 1. For a subformula β, we write $\bar{x}_\beta = 1$ to denote $x_\beta = 0$. All states are initial. The thicker states satisfy the justice requirements.

In the above, $Vars(\varphi,\psi)$ is a set of one variable x_β for each subformula β involved in φ and ψ. At any state, variables are assigned Boolean values, indicating which (sub)formulas hold true at that state.

If φ and ψ are atomic propositions, the tester can be represented as an automaton with *5 states*, see Figure 1.

In the general case, where φ and ψ are not simply atomic propositions but rather LTL formulas, one can construct $T(\varphi\,\mathcal{U}\,\psi)$ by *composition* of the testers for their subformulas. The following is obtained from [10, Section 3.2] and [13, Section 7].

Definition 4 (Composition of Temporal Testers): The synchronous parallel composition of two testers is

$$(V,\Theta,R,\mathcal{J}) = (V_1,\Theta_1,R_1,\mathcal{J}_1) \;|||\; (V_2,\Theta_2,R_2,\mathcal{J}_2),$$

where $V = V_1 \cup V_2$, $\Theta = \Theta_1 \wedge \Theta_2$, $R = R_1 \wedge R_2$ and $\mathcal{J} = \mathcal{J}_1 \cup \mathcal{J}_2$.

1) For a unary LTL operator op, the tester $T(\mathrm{op}(\varphi_1))$ is

$$T_{p\leftarrow\varphi_1}(\mathrm{op}(p)) \;|||\; T(\varphi_1),$$

where we replace every instance of the variables x_p and $x_{\mathrm{op}(p)}$ of the first tester by the variables x_{φ_1} and $x_{\mathrm{op}(\varphi_1)}$.

2) For a binary LTL operator op, the tester $T(\mathrm{op}(\varphi_1,\varphi_2))$ is

$$T_{p\leftarrow\varphi_1,q\leftarrow\varphi_2}(\mathrm{op}(p,q)) \;|||\; T(\varphi_1) \;|||\; T(\varphi_2),$$

where we replace every instance of the variables x_p, x_q and $x_{\mathrm{op}(p,q)}$ of the first tester by the variables x_{φ_1}, x_{φ_2} and $x_{\mathrm{op}(\varphi_1,\varphi_2)}$ respectively.

Since for every $p \in \mathcal{P}$, $\Diamond p = true\,\mathcal{U} p$ and $\Box p = \neg\Diamond\neg p$, we construct the testers for these formulas from that of $p\,\mathcal{U}\,q$. By composing them, we obtain the tester of $\Box\Diamond p$. These are represented in Figures 2, 3 and 4.

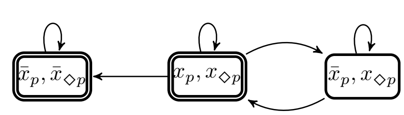

Fig. 2. Tester $T(\Diamond p)$. All states are initial, thicker states satisfy the justice requirement.

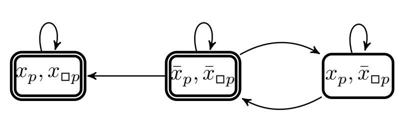

Fig. 3. Tester $T(\Box p)$. All states are initial, thicker states satisfy the justice requirement.

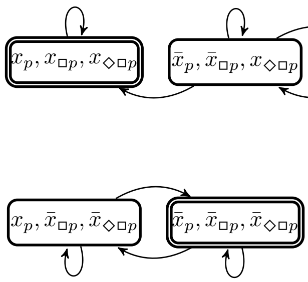

Fig. 4. Tester $T(\Diamond\Box p)$, with two disjoint components. All states are initial, thicker states need to be visited infinitely often.

Definition 5 (Number of states in a Tester): Given a tester $T(\varphi)$, $\varphi \in \mathrm{LTL}(\mathcal{P})$ and $i, j, k \in \{0,1\}$, let

- $|T(\varphi)|$ be its number of states,
- $|T(\varphi)|_i$ be the number of states where $x_\varphi = i$.

For any formulas $\varphi, \psi \in \mathrm{LTL}(\mathcal{P})$,

- for any unary operator op, $|T(\mathrm{op}(\varphi))|_{i,j}$ is the number of states where $x_\varphi = i, x_{\mathrm{op}(\varphi)} = j$,
- for any binary operator op, $|T(\mathrm{op}(\varphi,\psi))|_{i,j,k}$ is the number of states where $x_\varphi = i, x_\psi = j, x_{\mathrm{op}(\varphi,\psi)} = k$.

The number of states in a tester can be decomposed as follows for any $\varphi, \psi \in \mathrm{LTL}(\mathcal{P})$:

$$|T(\mathrm{op}(\varphi))| = \sum_{i,j} |T(\mathrm{op}(\varphi))|_{i,j}$$
$$|T(\mathrm{op}(\varphi,\psi))| = \sum_{i,j,k} |T(\mathrm{op}(\varphi,\psi))|_{i,j,k}.$$

Proposition 2.1: Let p, q be two atomic propositions in $\mathcal{P}$ and $\psi_1, \psi_2 \in \mathrm{LTL}(\mathcal{P})$ be two LTL formulas. The following holds:

$$|T(\mathrm{op}(\psi_1))|_{i,j} \le |T(\psi_1)|_i |T(\mathrm{op}(p))|_{i,j}, \tag{4}$$

$$\begin{aligned}&|T(\mathrm{op}(\psi_1,\psi_2))|_{i,j,k} \le \\ &\quad |T(\psi_1)|_i |T(\psi_2)|_j |T(\mathrm{op}(p,q))|_{i,j,k}.\end{aligned} \tag{5}$$

where op denotes an LTL operator.

Corollary 2.2 (Recursive Bounds): Consider a tester $|T(\varphi)|$ for $\varphi \in \mathrm{LTL}(\mathcal{P})$.

- if $\varphi \in \mathcal{P}$,
$$|T(\varphi)| = 2. \tag{6}$$
- if $\varphi = \neg\psi$,
$$\forall i, j : |T(\neg\psi)|_{i,j} = |T(\psi)|_i \text{ if } i \ne j, \text{ 0 else.} \tag{7}$$
- for any unary operator ($\mathrm{op} \in \{\Diamond, \Box, \bigcirc\}$), we have
$$|T(\mathrm{op}(\psi))| \le 2|T(\psi)|. \tag{8}$$
- if $\varphi = \Diamond\Box\psi$ we get
$$\begin{aligned}&|T(\Diamond\Box\psi)| \le 2\cdot|T(\psi)|_1 + 3\cdot|T(\psi)|_0,\\ &\le 3|T(\psi)|.\end{aligned} \tag{9}$$
- if $\varphi = \psi_1 \wedge \psi_2$ or $\psi_1 \vee \psi_2$ or $\psi_1 \Rightarrow \psi_2$, we get
$$|T(\varphi)| \le |T(\psi_1)|\cdot|T(\psi_2)|. \tag{10}$$
- if $\varphi = \psi_1 \,\boldsymbol{\mathcal{U}}\, \psi_2$ or $\varphi = \psi_1 \,\mathcal{R}\, \psi_2$, we get
$$|T(\varphi)| \le 2\cdot|T(\psi_1)|\cdot|T(\psi_2)|. \tag{11}$$

Remark 3 (Growth of justice sets): For the elementary testers studied, the number of justice requirements $|\mathcal{J}|$ is 1. By composition (Definition 4), the number of justice requirements for a LTL formula φ is at most $|\varphi|$. The tester $T(\Diamond\Box p)$ in Figure 4 is peculiar in this regard because it has been optimized. A direct application of the definition leads to having two justice requirements:

$$\mathcal{J} = \{\neg x_{\Diamond\Box p} \vee x_{\Box p}\} \cup \{x_{\Box p} \vee \neg x_p\}.$$

The two justice requirements are met simultaneously at the states $(x_p, x_{\Box p}, x_{\Diamond\Box p})$ and $(\bar{x}_p, \bar{x}_{\Box p}, \bar{x}_{\Diamond\Box p})$. On this ground, we use the single requirement $\mathcal{J} = \{(x_p \wedge x_{\Box p} \wedge x_{\Diamond\Box p}) \vee \neg(x_p \vee x_{\Box p} \vee x_{\Diamond\Box p})\}$ while preserving the computations of $T_{p\leftarrow\Box p}(\Diamond p) \;|||\; T(\Box p)$.

This allows us to focus on bounding the number of states for testers involved in rLTL model checking.

We are now in position to tackle our main problem in this paper, which is computing tight complexity bounds for rLTL Model Checking.

III. rLTL Model Checking

As discussed in the introduction, the main goal of rLTL is to embed a notion of robustness into LTL. With this in mind, the syntax of rLTL closely resembles that of LTL using *robust* versions of LTL operators.

Definition 6 (rLTL syntax): Let $\mathcal{P}$ be a nonempty, finite set of atomic propositions. The set of all rLTL formulas on $\mathcal{P}$, written $\mathrm{rLTL}(\mathcal{P})$, is the smallest set satisfying

- $\mathcal{P} \subset \mathrm{rLTL}(\mathcal{P})$ and
- if φ and ψ are elements of $\mathrm{rLTL}(\mathcal{P})$, then $\neg\varphi$, $\varphi\vee\psi$, $\varphi\wedge\psi$, $\varphi \Rrightarrow \psi$, $\odot\varphi$, $\boxdot\varphi$, $\Diamond\!\!\!\cdot\,\varphi$, $\varphi \,\mathcal{R}\, \psi$ and $\varphi \,\boldsymbol{\mathcal{U}}\, \psi$ are elements of $rLTL(\mathcal{P})$ as well.

The length of a formula $\varphi \in \mathrm{rLTL}(\mathcal{P})$ is denoted by $|\varphi|$ and is the number of subformulas it contains.

Given a word $\sigma \in \left(2^{\mathcal{P}}\right)^\omega$ and a formula $\varphi \in \mathrm{rLTL}(\mathcal{P})$, the semantics of rLTL provides the degree to which σ satisfies the LTL counterpart[2] of φ. This is captured by using a 5-valued semantics, with one truth value corresponding to **true** and the others to *different shades of* **false**.

Formally, the truth value of an rLTL formula is a 4-tuple belonging to the set

$$\begin{aligned}\mathbb{B}_5 &= \{0000, 0001, 0011, 0111, 1111\},\\ &= \{\mathbb{B}_5[0], \mathbb{B}_5[1], \mathbb{B}_5[2], \mathbb{B}_5[3], \mathbb{B}_5[4]\},\end{aligned}$$

where $\mathbb{B}_5[n] \in \mathbb{B}_5$, for $0 \le n \le 4$, is the truth value with n bits set to 1. The truth values are ordered as follows

$$0000 \prec 0001 \prec 0011 \prec 0111 \prec 1111, \tag{12}$$

with 1111 corresponding to **true** and the remaining ones corresponding to different shades of **false**. With respect

[2]The LTL counterpart of any rLTL formula is obtained by removing all the dots or dashes superimposed on the operators.

TABLE I

THE FULL SEMANTICS OF rLTL AND THE ltl OPERATOR.

Operator	Symbol	Semantics, for $p \in \mathcal{P}$, $\varphi, \psi \in \mathrm{rLTL}(\mathcal{P})$.
Atomic Proposition		$\forall 1 \le i \le 4 : \mathrm{ltl}(i, p) = p.$
Negation	$\neg$	$\forall 1 \le i \le 4 : \mathrm{ltl}(i, \neg\varphi) = \neg\mathrm{ltl}(1, \varphi).$
Disjunction	$\vee$	$\forall 1 \le i \le 4 : \mathrm{ltl}(i, \varphi \vee \psi) = \mathrm{ltl}(i, \varphi) \vee \mathrm{ltl}(i, \psi).$
Conjunction	$\wedge$	$\forall 1 \le i \le 4 : \mathrm{ltl}(i, \varphi \wedge \psi) = \mathrm{ltl}(i, \varphi) \wedge \mathrm{ltl}(i, \psi).$
Robust Implication	$\Rrightarrow$	$\forall 1 \le i \le 3 : \mathrm{ltl}(i, \varphi \Rrightarrow \psi) = (\mathrm{ltl}(i, \varphi) \Rightarrow \mathrm{ltl}(i, \psi)) \wedge \mathrm{ltl}(i+1, \varphi \Rrightarrow \psi),$ $\mathrm{ltl}(4, \varphi \Rrightarrow \psi) = (\mathrm{ltl}(4, \varphi) \Rightarrow \mathrm{ltl}(4, \psi)).$
Next	$\odot$	$\forall 1 \le i \le 4 : \mathrm{ltl}(i, \odot\varphi) = \bigcirc\mathrm{ltl}(i, \varphi).$
Robust Always	$\boxdot$	$\mathrm{ltl}(1, \boxdot\varphi) = \Box\mathrm{ltl}(1, \varphi),$ $\mathrm{ltl}(2, \boxdot\varphi) = \Diamond \Box\, \mathrm{ltl}(2, \varphi),$ $\mathrm{ltl}(3, \boxdot\varphi) = \Box \Diamond\, \mathrm{ltl}(3, \varphi),$ $\mathrm{ltl}(4, \boxdot\varphi) = \Diamond\mathrm{ltl}(4, \varphi).$
Robust Eventually	$\diamonddot$	$\forall 1 \le i \le 4 : \mathrm{ltl}(i, \diamonddot\varphi) = \Diamond\mathrm{ltl}(i, \varphi).$
Robust Until	$\boldsymbol{\mathcal{U}}$	$\forall 1 \le i \le 4 : \mathrm{ltl}(i, \varphi\, \boldsymbol{\mathcal{U}}\, \psi) = \mathrm{ltl}(i, \varphi)\, \mathcal{U}\, \mathrm{ltl}(i, \psi).$
Robust Release	$\boldsymbol{\mathcal{R}}$	$\mathrm{ltl}(1, \varphi\, \boldsymbol{\mathcal{R}}\, \psi) = \mathrm{ltl}(1, \varphi)\, \mathcal{R}\, \mathrm{ltl}(1, \psi),$ $\mathrm{ltl}(2, \varphi\, \boldsymbol{\mathcal{R}}\, \psi) = \Diamond \Box\, \mathrm{ltl}(2, \psi) \vee \Diamond\mathrm{ltl}(2, \varphi),$ $\mathrm{ltl}(3, \varphi\, \boldsymbol{\mathcal{R}}\, \psi) = \Box \Diamond\, \mathrm{ltl}(3, \psi) \vee \Diamond\mathrm{ltl}(3, \varphi),$ $\mathrm{ltl}(4, \varphi\, \boldsymbol{\mathcal{R}}\, \psi) = \Diamond\mathrm{ltl}(4, \psi) \vee \Diamond\mathrm{ltl}(4, \varphi).$

to the example $\boxdot p$ in the introduction, the truth value $\mathbb{B}_5[4] = 1111$ corresponds to the LTL formula $\Box p$ being satisfied, $\mathbb{B}_5[3] = 0111$ corresponds to $\Diamond \Box p$, $\mathbb{B}_5[2] = 0011$ corresponds to $\Box \Diamond p$, $\mathbb{B}_5[1] = 0001$ corresponds to $\Diamond p$, and $\mathbb{B}_5[0] = 0000$ corresponds to $\Box\neg p$.

Hence, a truth value in $\mathbb{B}_5$ can be viewed as a sequence of 4 bits. In order to introduce the rLTL semantics, we assign to each bit of an rLTL truth value an LTL formula. The definition below is equivalent to that of [17].

Definition 7 (rLTL semantics): For a set of atomic propositions $\mathcal{P}$, we define the operator

$$\mathrm{ltl} : \{1, \dots, 4\} \times \mathrm{rLTL}(\mathcal{P}) \to \mathrm{LTL}(\mathcal{P}) \tag{13}$$

as in Table I. The rLTL semantics is defined as a function

$$V : (2^{\mathcal{P}})^{\omega} \times \mathrm{rLTL}(\mathcal{P}) \to \mathbb{B}_5,$$

where for any $\sigma \in (2^{\mathcal{P}})^{\omega}, \varphi \in \mathrm{rLTL}(\mathcal{P})$ and $1 \le i \le 4$, the ith bit $V_i(\sigma, \varphi)$ of the valuation $V(\sigma, \varphi)$ is given by:

$$V_i(\sigma, \varphi) = W(\sigma, \mathrm{ltl}(i, \varphi)).$$

The difficulty of dealing with the 5-valued semantics of rLTL formulas lies in the fact that the four bits of a truth value are coupled by robust implications and negations. Intuitively, a negation changes true to false and *all shades of false* to true. This is done effectively through the first bit of an rLTL formula and explains the coupling in the evaluation of any bit. In a similar manner, each bit of the robust implication, needs the value of the next less important bit. The following example will provide good intuition to this problem.

Example 2: Consider the rLTL formula

$$\varphi = \neg(a \Rrightarrow (b\, \mathcal{R}\, c)),$$

where a, b and c are atomic propositions. To compute for example the 4th bit of the rLTL valuation, one needs to unfold the corresponding LTL formula. Using the semantics in Table I, we obtain

$$\begin{aligned}\mathrm{ltl}(4, \varphi) &= \neg\mathrm{ltl}(1, a \Rrightarrow (b\, \mathcal{R}\, c)),\\ &= \neg[\mathrm{ltl}(1, a) \Rightarrow \mathrm{ltl}(1, (b\, \mathcal{R}\, c)) \wedge \\ &\quad \mathrm{ltl}(2, (a \Rrightarrow (b\, \mathcal{R}\, c)))],\\ &= a \wedge \neg(b\, \mathcal{R}\, c) \vee \neg\mathrm{ltl}(2, (a \Rrightarrow (b\, \mathcal{R}\, c))).\end{aligned}$$

Continuing unfolding the formula, we see that to check its 4th bit one needs to check a relatively large LTL formula.

A. The rLTL Model Checking Problem

The model checking problem for LTL asks whether or not a model (set of words) satisfies an LTL specification. In rLTL, the model checking problem is intuitively understood as the question "how much does a model satisfy a specification"?

Problem 2 (The Model Checking Problem for rLTL): Given a set of atomic propositions $\mathcal{P}$, a set set of words $\mathcal{L} \subseteq \left(2^{\mathcal{P}}\right)^{\omega}$ recognized by a Generalized Büchi Automaton $\mathcal{A}$, and $\varphi \in \mathrm{rLTL}(\mathcal{P})$, compute

$$b(\mathcal{L}, \varphi) = \min_{\sigma \in \mathcal{L}} V(\sigma, \varphi). \tag{14}$$

Note that in (14), $V(\sigma, \varphi) \in \mathbb{B}_5$, and the minimum follows from the ordering defined in (12).

Remark 4: In [17, Theorem 4.9], the authors provide a technique for rLTL model checking that follows the standard steps described in Remark 1.
There, given an rLTL formula φ, a GBA with $N_{\varphi}^{\mathrm{rLTL}}$ states and $F_{\varphi}^{\mathrm{rLTL}}$ accepting conditions, where

$$N_{\varphi}^{\mathrm{rLTL}} = \mathcal{O}(5^{|\varphi|}),\ F_{\varphi}^{\mathrm{rLTL}} = \mathcal{O}(|\varphi|) \tag{15}$$

is constructed, and composed[3] with the GBA recognizing the language $\mathcal{L}$.

[3] From there, one can deduce complexity bounds for rLTL model checking by applying (1). Note that there is a typo in the statement of [17, Theorem 4.9], where the authors omitted the quadratic terms due to the presence of the number of transitions in (1).

Data: A language $\mathcal{L}$ generated by a GBA $\mathcal{A}$, a formula $\varphi \in \text{rLTL}(\mathcal{P})$.
Result: Computes $b(\mathcal{L}, \varphi)$ (see (14)).
for $j = 0, \ldots, 3$ **do**
    $w := \inf_{\sigma \in \mathcal{L}} W(\sigma, \text{ltl}(4 - j, \varphi))$.
    if $w = 0$ **then**
        return $\mathbb{B}_5[j]$
    end
end
return $\mathbb{B}_5[4]$

Algorithm 1: rLTL model checking algorithm.

The bound (15) on the number of states is already non-trivial. Assume a formula $\boxdot p \Rrightarrow \boxdot q$, and we wish to check $V(\sigma, \varphi) = 0111$ for every $\sigma \in \mathcal{L}$. This is equivalent to model checking

$$\text{ltl}(2, \boxdot p \Rrightarrow \boxdot q) = (\Box \Diamond p \Rightarrow \Box \Diamond q) \wedge (\Diamond \Box p \Rightarrow \Diamond \Box q) \\ \wedge (\Diamond p \Rightarrow \Diamond q).$$

The original rLTL formula has length 5, and the LTL formula above has length 20. Bound (15) dictates complexity proportional to the 5^5 states of the GBA used, which is an improvement over the 2^{20} states from (2).

In this section, we show that for the fragment consisting of formulas of the form $\varphi \Rrightarrow \psi$ for $\varphi, \psi \in \widetilde{\text{rLTL}}$ (see below), rLTL model checking can be performed using automata with at most

$$\mathcal{O}\Big(2^{|\varphi| - k(\varphi)} 3^{k(\varphi)}\Big)$$

states, where $k(\varphi)$ is the number of $\boxdot$ operators in the rLTL formula φ.

Definition 8 ($\widetilde{\text{rLTL}}$): Given a set of atomic propositions $\mathcal{P}$, define the fragment $\widetilde{\text{rLTL}}(\mathcal{P}) \subset \text{rLTL}(\mathcal{P})$ as the set of all rLTL formulas without operators $\Rrightarrow$ or $\mathcal{R}$.

Remark 5: Our main result, considers a fragment larger than $\widetilde{\text{rLTL}}$, which allows one initial implication.

This fragment has the particularity that the truth value for a bit i of a rLTL valuation is independent from the bits $j \neq i$ (see Table I).

Lemma 3.1: Given a set of atomic propositions $\mathcal{P}$, for any $\varphi \in \widetilde{\text{rLTL}}(\mathcal{P})$, and for any $1 \leq i \leq 4$,

$$|T(\text{ltl}(i, \varphi))| \leq 2^{|\varphi| - k(\varphi)} 3^{k(\varphi)}, \tag{16}$$

where $k(\varphi)$ is the number of operators $\boxdot$ in φ.

For our main result, Theorem 3.2, we consider Algorithm 1 for rLTL model checking.

Theorem 3.2: Consider a set of atomic propositions $\mathcal{P}$, a set $\mathcal{L} \subseteq \left(2^{\mathcal{P}}\right)^{\omega}$ recognized by a GBA $\mathcal{A}$ with N states. Let φ be any formula in the rLTL fragment

$$\widetilde{\text{rLTL}}(\mathcal{P}) \cup \{\psi_1 \Rrightarrow \psi_2 \mid \psi_1, \psi_2 \in \widetilde{\text{rLTL}}(\mathcal{P})\}. \tag{17}$$

Algorithm 1 computes $b(\mathcal{L}, \varphi) = \mathbb{B}_5[\ell]$, $0 \leq \ell \leq 4$ by performing $\min(\ell + 1, 4)$ LTL model-checking steps, each using an automaton of size at most

$$\mathcal{O}\left(2^{|\varphi| - k(\varphi)} 3^{k(\varphi)}\right). \tag{18}$$

IV. Conclusions

In this paper we have identified a fragment of rLTL for which the time complexity of the model checking problem approaches that of LTL. We believe this complexity result combined with the syntactic similarity between LTL and rLTL will motivate the widespread use of rLTL to specify and verify robustness properties. To further contribute towards this objective, the authors are currently implementing the algorithms described in this paper in a verification tool for rLTL.

References

[1] C. Baier, J.-P. Katoen, and K. G. Larsen, *Principles of model checking*. MIT press, 2008.

[2] E. M. Clarke, O. Grumberg, and D. Peled, *Model checking*. MIT press, 1999.

[3] E. Dallal, D. Neider, and P. Tabuada, "Synthesis of safety controllers robust to unmodeled intermittent disturbances," in *Decision and Control (CDC), 2016 IEEE 55th Conference on*. IEEE, 2016, pp. 7425–7430.

[4] A. Donzé and O. Maler, "Robust satisfaction of temporal logic over real-valued signals," in *International Conference on Formal Modeling and Analysis of Timed Systems*. Springer, 2010, pp. 92–106.

[5] G. E. Fainekos and G. J. Pappas, "Robustness of temporal logic specifications," in *Formal Approaches to Software Testing and Runtime Verification*. Springer, 2006, pp. 178–192.

[6] ——, "Robustness of temporal logic specifications for continuous-time signals," *Theoretical Computer Science*, vol. 410, no. 42, pp. 4262–4291, 2009.

[7] R. Gerth, D. Peled, M. Y. Vardi, and P. Wolper, "Simple on-the-fly automatic verification of linear temporal logic," in *Protocol Specification, Testing and Verification XV*. Springer, 1995, pp. 3–18.

[8] R. Goebel, J. Hespanha, A. R. Teel, C. Cai, and R. Sanfelice, "Hybrid systems: Generalized solutions and robust stability," *IFAC Proceedings Volumes*, vol. 37, no. 13, pp. 1–12, 2004.

[9] R. Goebel, R. G. Sanfelice, and A. R. Teel, *Hybrid Dynamical Systems: modeling, stability, and robustness*. Princeton University Press, 2012.

[10] Y. Kesten, A. Pnueli, and L.-o. Raviv, "Algorithmic verification of linear temporal logic specifications," in *International Colloquium on Automata, Languages, and Programming*. Springer, 1998, pp. 1–16.

[11] O. Lichtenstein and A. Pnueli, "Checking that finite state concurrent programs satisfy their linear specification," in *Proceedings of the 12th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*. ACM, 1985, pp. 97–107.

[12] R. Majumdar, E. Render, and P. Tabuada, "A theory of robust omega-regular software synthesis," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 13, no. 3, p. 48, 2013.

[13] A. Pnueli and A. Zaks, "On the merits of temporal testers," in *25 Years of Model Checking*. Springer, 2008, pp. 172–195.

[14] M. Rungger and P. Tabuada, "A notion of robustness for cyber-physical systems," *IEEE Transactions on Automatic Control*, vol. 61, no. 8, pp. 2108–2123, 2016.

[15] P. Schnoebelen, "The complexity of temporal logic model checking." *Advances in modal logic*, vol. 4, no. 393-436, p. 35, 2002.

[16] P. Tabuada, S. Y. Caliskan, M. Rungger, and R. Majumdar, "Towards robustness for cyber-physical systems," *IEEE Transactions on Automatic Control*, vol. 59, no. 12, pp. 3151–3163, 2014.

[17] P. Tabuada and D. Neider, "Robust linear temporal logic," *arXiv preprint arXiv:1510.08970*, 2015.

[18] Y.-K. Tsay, M.-H. Tsai, J.-S. Chang, and Y.-W. Chang, "Büchi store: an open repository of büchi automata," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2011, pp. 262–266.