

Regular Linear Temporal Logic^{*}

Martin Leucker¹ and César Sánchez^{2,3}

¹ Institut für Informatik

TU München, Germany

² Computer Science Department

Stanford University, Stanford, USA

³ Computer Engineering Department

University of California, Santa Cruz, USA

Abstract. We present regular linear temporal logic (RLTL), a logic that generalizes linear temporal logic with the ability to use regular expressions arbitrarily as sub-expressions. Every LTL operator can be defined as a context in regular linear temporal logic. This implies that there is a (linear) translation from LTL to RLTL.

Unlike LTL, regular linear temporal logic can define all ω -regular languages, while still keeping the satisfiability problem in PSPACE. Unlike the extended temporal logics ETL*, RLTL is defined with an algebraic signature. In contrast to the linear time μ -calculus, RLTL does not depend on fix-points in its syntax.

1 Introduction

We present *regular linear temporal logic* (RLTL), a formalism to express properties of infinite traces by conveniently *fusing* regular-expressions and linear-temporal logic. Moreover, we show that the satisfiability and equivalence of RLTL expressions are PSPACE-complete problems.

The linear temporal logic (LTL) [19,16] is a modal logic over a linear frame, whose formulas express properties of infinite traces using two modalities: *next-time* and *until*. LTL is a widely accepted formalism for the specification and verification of concurrent and reactive systems. However, Wolper [26] showed that LTL cannot express all ω -regular properties (the properties expressible by finite-state automata on infinite words, known as Büchi automata [4]). In particular, it cannot express the property “ p holds at every other moment”. In spite of being a useful specification language, this lack of expressivity seems to surface in practice [20] and it has been pointed out (see for example [3]) that regular-expressions are sometimes very convenient in addition to LTL, in formal specifications. Actually, in the industry standard specification language PSL, arbitrary mixtures of regular expressions and LTL are allowed [1].

^{*} Part of this work was done during the first author’s stay at Stanford University and was supported by ARO DAAD190310197. The second author has been supported in part by NSF grants CCR-01-21403, CCR-02-20134, CCR-02-09237, CNS-0411363, and CCF-0430102, and by NAVY/ONR contract N00014-03-1-0939.

To solve the expressivity problem, Wolper introduced the so called extended temporal logic ETL where new operators are defined as right linear grammars, and language composition is used to compose operators. ETL was later extended [25] to different kinds of automata. The main drawback of the extended temporal logics is that, in order to obtain the full expressivity, an infinite number of operators is needed.

An alternative approach consists on adapting the modal μ -calculus [5,12] to the linear setting, which gives rise to the linear time μ -calculus, denoted as ν TL [2]. Here, the full expressivity is obtained by allowing the use of fix point operators. It can be argued that this formalism is not algebraic either since one needs to specify recursive equations to describe temporal properties. Moreover, the only modality is the *nexttime*. Even though every ground regular expression can be translated into a ν TL expression (see [14]), the concatenation operator cannot be directly represented in ν TL, i.e., there is no context of ν TL that captures concatenation. On the other hand, extending ν TL with concatenation (the so-called fix point logic with chop FLC [18,15]) allows expressing non-regular languages. This extra expressive power comes at the price of undecidable satisfiability and equivalence problems. A more restricted extension of ν TL allowing only left concatenation with regular expressions is possible along the lines presented here, but this is out of the scope of this paper.

There have also been dynamic logics that try to merge regular expressions (for the program part) and LTL (for the action part), for example, Regular Process Logic [7]. However, it makes the satisfiability problem non-elementary by allowing arbitrary combinations of negations and regular operators. Dynamic linear-temporal logic DLTL [8] keeps the satisfiability problem in PSPACE, but restricts the use of regular expressions only as a generalization of the until operator. While the generalized until present in DLTL and the power operators present in RLTL are complementary (in the sense that none can be defined in terms of each other), the power operators are more suitable for extensions that can handle past, as discussed in Section 5.

An arbitrary mixture of (sequentially extended) regular expressions and LTL is possible in PSL [1,6]. However, decision procedures for satisfiability etc. and their complexities are still an area of active research (for full PSL). Thus, RLTL can be understood as subset of PSL for which an efficient satisfiability procedure (PSPACE) is available.

The logic that we present here is a generalization of linear temporal logic and ω -regular expressions, based on the following observation. It is common for different formalisms to find the following three components in the (recursive) definition of operators:

1. *attempt*: an expression that captures the first try to satisfy the enclosing expression.
2. *obligation*: an expression that must be satisfied, if the attempt fails, to continue trying the enclosing expression. If both the attempt and the obligation fail, the sequence is not matched.