Information and Computation 262 (2018) 265–279

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

Cycle detection in computation tree logic ☆

Gaëlle Fontaine [a], Fabio Mogavero [b], Aniello Murano [c], Giuseppe Perelli [b,*], Loredana Sorrentino [c]

[a] Universidad de Chile, Santiago de Chile, Chile
[b] University of Oxford, Oxford, UK
[c] Università degli Studi di Napoli "Federico II", Napoli, Italy

ARTICLE INFO

Article history:
Received 23 April 2017
Available online 6 September 2018

Keywords:
Temporal logic
Model checking
Satisfiability
Verification

ABSTRACT

We introduce Cycle-CTL*, an extension of CTL* with cycle quantifications that are able to predicate over cycles. The introduced logic turns out to be very expressive. Indeed, we prove that it strictly extends CTL* and is orthogonal to μCALCULUS. We also give an evidence of its usefulness by providing few examples involving non-regular properties. We extensively investigate both the model-checking and satisfiability problems for Cycle-CTL* and some of its variants/fragments.

1. Introduction

Temporal logic is a suitable framework largely used in formal system verification [27,8,11,10]. It allows to specify and reason in a rigorous manner about the temporal evolution of a system, without talking explicitly about the elapsing of time. Two fundamental decision problems involving temporal logics have been deeply investigated: *model checking* and *satisfiability*. The former, given a mathematical model of the system, such as a *Kripke* structure, asks whether it satisfies a temporal logic formula specifying its desired behavior. The latter, instead, checks whether the temporal logic specification is consistent and, thus, a corresponding system is feasible [10].

In several situations, reasoning about system correctness and, in particular, solving the above decision questions, reduces to detect precise cycle properties over the system model. For example, in the classical automata-theoretic approach there are settings in which the satisfiability question reduces to first build a Büchi automaton accepting all models of the formula and then to check for its non-emptiness [22]. The latter can be solved by looking for a "lasso", that is a path from the initial state to a final state belonging to a cycle [22,18]. Similarly, if one uses a game-theory approach, solving the model checking or the satisfiability questions reduces to first construct a two-player game, such as a Büchi or a parity game [13,22,23,3, 16,31], and then check for the existence of a winning strategy for a designated player. The latter can be reduced to check whether it has the ability to confine the evolution of the game (a *play*) over some specific cycle over the arena, no matter how the other player behaves.

Depending on the view of the underlying nature of time, two types of temporal logics are mainly considered. In *linear-time temporal logics*, such as LTL [27], time is treated as if each moment in time has a unique possible future. Conversely, in *branching-time temporal logics* such as CTL [8] and CTL* [12] each moment in time may split into various possible futures.

☆ This paper is an extended version of [15].

* Corresponding author.
E-mail address: giuseppe.perelli@leicester.ac.uk (G. Perelli).

https://doi.org/10.1016/j.ic.2018.09.007
0890-5401/

Then, to express properties along one or all the possible futures we make use of existential and universal quantifiers. Noticeably, LTL is suitable to express path properties; CTL is more appropriate to express state-based properties; finally, CTL* has the power to express combinations of path and state properties. In the years, these logics have been extended in a number of ways in order to express very complicated specification properties. Surprisingly, no temporal logic has been introduced so far to reason explicitly about cycles, despite their usefulness. In addition to the technical motivation mentioned above, there are often cases in which it is useful to distinguish between purely infinite behaviors, like those occurring in infinite-state systems, from regular infinite behaviors [6,19], which can be detected by looking for cycles. Moreover, also in finite-state systems there are infinite behaviors that are not regular, like the one referred as *prompt* in [20,26], which also can be detected by looking at cyclic computations.

In this paper we study *Cycle*-CTL* (CTL$^{\star}_{\circlearrowleft}$) [15], an extension of the classical logic CTL* along with the ability to predicate over cycles. For a *cycle* we mean a path that passes through its initial state infinitely often. Syntactically, CTL$^{\star}_{\circlearrowleft}$ is obtained by enriching CTL* with two novel cycle quantifiers, namely the existential one $\mathtt{E}^{\circlearrowleft}$ and the universal one $\mathtt{A}^{\circlearrowleft}$. Note that CTL$^{\star}_{\circlearrowleft}$ still uses the classical quantifiers $\mathtt{E}$ and $\mathtt{A}$. Hence, we can use it to specify models whose behavior results as an opportune combination of standard paths and cycles. In particular, CTL$^{\star}_{\circlearrowleft}$ can specify the existence of a lasso within a model.

We study the expressiveness of CTL$^{\star}_{\circlearrowleft}$ and show that it is strictly more expressive than CTL* but orthogonal to μ-calculus. To give an evidence of the power and usefulness of the introduced logic, we provide some examples along the paper. Precisely, we first show how CTL$^{\star}_{\circlearrowleft}$ can be used to reasoning, in a very natural way, about liveness properties restricted to cycles. Precisely, we show how to specify that some designated properties recurrently occurs in the starting state of a cycle. As another example, we show the ability of the logic to handle non-regular properties such as the "*prompt-parity condition*" [26]. In temporal logic, we can specify properties that will eventually hold, but this gives no bound on the moment they will occur. Prompt temporal logics and games have been deeply investigated in order to restrict reasoning about properties that only occur in bounded time [7,1,21,4,26].

We investigate both the model checking and the satisfiability questions for CTL$^{\star}_{\circlearrowleft}$ and provide some automata-based solutions. For the model checking question we provide a PSPACE upper-bound by opportunely extending the classical approach that is used for CTL* [22]. Specifically, we add a machinery consisting of an appropriate Büchi automaton that checks in parallel whether a path is a cycle and satisfies a required formula. Concerning the satisfiability question, we introduce instead a novel approach that makes use of two-way automata [28]. These automata, largely investigate and used in formal verification [5,14,19], allow to traverse trees both in forward and backward. The reason why we cannot use and extend the classical approach provided for CTL* (see [22]) resides on the fact that such an approach makes strongly use of some positive properties that hold for CTL*, among the others the tree- and the finite-model ones. Unluckily and unsurprisingly, due to the ability in CTL$^{\star}_{\circlearrowleft}$ to force (and even more to forbid) the existence of cycles, we lose in this logic both these properties. This requires the introduction of novel and *ad hoc* definitions of bisimulation and tree-like unwinding to be used along with the automata-based approach. In particular, two-way tree automata are used to collect all tree representations of such tree-like unwinding structures. By means of this machinery we show that the satisfiability question for the full logic is 3EXPTIME.

In addition to CTL$^{\star}_{\circlearrowleft}$, we also introduce Simple-CTL$^{\star}_{\circlearrowleft}$: a semantic variant of the logic in which the cycle quantifications predicate only on simple cycles. By using a similar reasoning as for CTL$^{\star}_{\circlearrowleft}$, also Simple-CTL$^{\star}_{\circlearrowleft}$ strictly includes CTL*. In particular, it is orthogonal to μ-calculus. We investigate both the model-theoretic and the satisfiability problems for Simple-CTL$^{\star}_{\circlearrowleft}$, showing that their complexities correspond to the ones for CTL*. Finally, we investigate Cycle-CTL and Simple-Cycle-CTL as the natural CTL-like fragments of the introduced logics.

Outline of the paper. The paper is divided into sections as follows. In Section 2, we introduce the syntax and semantics of *Cycle*-CTL* and *Simple-Cycle*-CTL*, as well as the fragments corresponding to CTL. We also introduce some example to make the reader familiar with the new logic. In Section 3, we analyze the model-theoretic properties of these logics. In particular, we first prove that they are not invariant under the classic notion of bisimulation, this showing that they cannot be embedded into either CTL* or μ-calculus. Then, we introduce the notion of *Cycle-bisimulation*, a refinement of the classic bisimulation for which our logic and its fragments are invariant. In Section 4 we analyze the computational complexities of both the model-checking and the satisfiability problem. Finally, in Section 5 we provide some discussion and future work.

2. Computation-tree logic with cycle detection

In this section we introduce and discuss the syntax and semantics of Cycle-CTL* (CTL$^{\star}_{\circlearrowleft}$, for short) and Simple-Cycle-CTL* (CTL$^{\star}_{s\circlearrowleft}$, for short), as well as their fragments CTL$_{\circlearrowleft}$ and CTL$_{s\circlearrowleft}$, respectively. To do this, we first recall the concept of Kripke structure and some related basic notions. Finally, we also discuss some interesting problems that can be expressed in our logic.

Models. We first provide the definition of the underlying model for our logics.

Definition 1 *(Kripke structure).* A *Kripke structure* (KS, for short) [17] over a finite set of *atomic propositions* AP is a tuple $\mathcal{K} \triangleq \langle \mathrm{AP}, \mathrm{W}, R, \mathrm{L}, w_0 \rangle$, where W is an enumerable non-empty set of *worlds*, $w_0 \in \mathrm{W}$ is a designated *initial world*, $R \subseteq \mathrm{W} \times \mathrm{W}$