Journal of Applied Logic 12 (2014) 395–416

Contents lists available at ScienceDirect

Journal of Applied Logic

www.elsevier.com/locate/jal

Temporal logics for concurrent recursive programs: Satisfiability and model checking[☆]

Benedikt Bollig[a], Aiswarya Cyriac[a,*], Paul Gastin[a], Marc Zeitoun[b]

[a] LSV, ENS Cachan, CNRS & INRIA, France
[b] LaBRI, Univ. Bordeaux & CNRS, France

ARTICLE INFO

Article history:
Received 8 August 2012
Accepted 7 April 2014
Available online 14 May 2014

Keywords:
Temporal logic
Concurrent recursive programs
Nested words
Mazurkiewicz traces
Satisfiability
Model checking

ABSTRACT

We develop a general framework for the design of temporal logics for concurrent recursive programs. A program execution is modeled as a partial order with multiple nesting relations. To specify properties of executions, we consider any temporal logic whose modalities are definable in monadic second-order logic and which, in addition, allows PDL-like path expressions. This captures, in a unifying framework, a wide range of logics defined for ranked and unranked trees, nested words, and Mazurkiewicz traces that have been studied separately. We show that satisfiability and model checking are decidable in EXPTIME and 2EXPTIME, depending on the precise path modalities.

1. Introduction

We are concerned with the analysis of computer programs and systems that consist of several components sharing an access to resources such as variables or channels. Any component itself might be built of several modules that can be called recursively resulting in complex infinite-state systems. The analysis of such programs, which consist of a fixed number of recursive threads communicating with one another, is particularly challenging, due to the intrinsically high complexity of interaction between its components. All the more, it is important to provide tools and algorithms that support the design of correct programs, or verify if a given program corresponds to a specification.

It is widely acknowledged that linear-time temporal logic (LTL) [32] is a yardstick among the specification languages. It combines high expressiveness (equivalence to first-order logic [22]) with a reasonable complexity of decision problems such as satisfiability and model checking. LTL has originally been considered for finite-state sequential programs. As real programs are often concurrent or rely on recursive procedures, LTL has been extended in two directions.

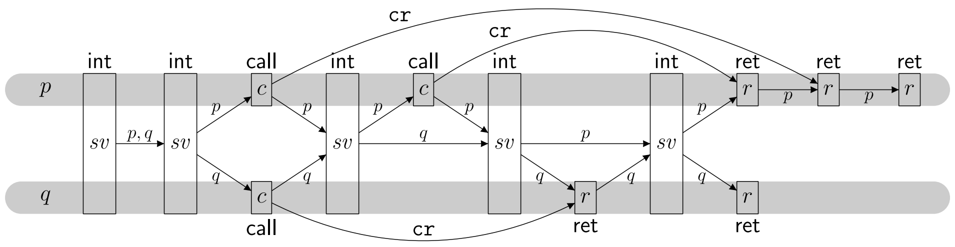
☆ Supported by ARCUS, DOTS (ANR-06-SETIN-003), DIGITEO LoCoReP and ANR 2010 BLAN 0202 01 FREC.
* Corresponding author.
E-mail addresses: bollig@lsv.ens-cachan.fr (B. Bollig), cyriac@lsv.ens-cachan.fr (A. Cyriac), gastin@lsv.ens-cachan.fr (P. Gastin), mz@labri.fr (M. Zeitoun).

http://dx.doi.org/10.1016/j.jal.2014.05.001
1570-8683/© 2014 Elsevier B.V. All rights reserved.

First, asynchronous finite-state programs (asynchronous automata) [36] are a formal model of shared-memory systems and properly generalize finite-state sequential programs. Their executions are no longer sequential (i.e., totally ordered) but can be naturally modeled as graphs or partial orders. In the literature, these structures are known as *Mazurkiewicz traces*. They look back on a long list of now classic results that smoothly extend the purely sequential setting (e.g., expressive equivalence of LTL and first-order logic) [16,17].

Second, in an influential paper, Alur and Madhusudan extend the finite-state sequential model to *visibly pushdown automata* (VPAs) [1]. VPAs are a flexible model for recursive programs, where subroutines can be called and executed while the current thread is suspended. The execution of a VPA is still totally ordered. However, it comes with some extra information that relates a subroutine call with the corresponding return position, which gives rise to the notion of *nested words* [1]. Alur et al. recently defined versions of LTL towards this infinite-state setting [3,2] that can be considered as canonical counterparts of the classical logic introduced by Pnueli.

To model programs that involve both recursion and concurrency, one needs to mix both views. Most approaches to modeling concurrent recursive programs, however, reduce concurrency to interleaving and neglect a behavioral semantics that preserves independencies between program events [5,24,25,33]. The first model for concurrent recursive programs with partial-order semantics was considered in [7]. Executions of their *concurrent VPAs* equip Mazurkiewicz traces with multiple nesting relations, as depicted in the figure below.



Temporal logics have not been considered for this natural concurrency-aware behavior model. Furthermore there is for now no canonical merge of the two existing approaches. It must be noted that satisfiability is undecidable when considering multiple nesting relations, even for simple logics. In fact, local control state reachability is also undecidable as two stacks (multiple nesting relations) are Turing powerful. Yet, it becomes decidable if we impose suitable restrictions to the system behaviors.

The first such restriction called bounded context-switching was proposed in [33] where a bound is placed on the number of times control can be transferred from one process to another. Furthermore experimental results suggest that bugs in programs usually manifest themselves within a few context switches [30]. A generalization of bounded context was proposed in [24] where a bound is placed on the number of *phases*: all processes may progress in a phase making recursive function calls, but at most one process is allowed to return from function calls. Thus a bounded phase behavior may have an unbounded number of context switches. While both these techniques allow under-approximate reachability, bounded phase covers significantly more behaviors than bounded context. Hence we adopt the bounded-phase restriction. We think that our constructions for bounded phase would serve as the first step towards getting similar results for other orthogonal restrictions such as bounded scope [23], ordered [6,11], or even theoretical but generic restrictions on behavior graphs such as bounded tree-width [28] or bounded split-width [14].

In this paper, we present a framework for defining (linear-time) temporal logics for concurrent recursive programs. A temporal logic may be parametrized by a finite set of modalities that are definable in monadic second-order logic (cf. [19]). Thus, existing temporal logics for sequential recursive programs [3,2,15] as well as for concurrent non-recursive programs [16,19,20] are easily definable in our framework. In addition, our