Contents lists available at ScienceDirect
Information and Computation
www.elsevier.com/locate/yinco

Visibly pushdown modular games, [☆], [☆☆]

I. De Crescenzo [a,*], S. La Torre [a,*], Y. Velner [b]

[a] Dipartimento di Informatica, Università degli Studi di Salerno, Italy
[b] The Blavatnik School of Computer Science, Israel

A R T I C L E   I N F O

Article history:
Received 23 April 2015
Available online xxxx

Keywords:
Pushdown
Games
Modular

A B S T R A C T

Recursive game graphs can be used to reason about the control flow of sequential programs with recursion. Here, the most natural notion of strategy is the modular one, i.e., a strategy that is local to a module and is oblivious to previous module invocations. In this work, we study for the first time modular strategies with respect to winning conditions expressed as languages of pushdown automata. We show that pushdown modular games are undecidable in general, and become decidable for visibly pushdown automata specifications. Our solution relies on a reduction to modular games with finite-state winning conditions. We carefully characterize the computational complexity of this decision problem by also considering as winning conditions nondeterministic and universal Büchi or co-Büchi visibly pushdown automata, and CaRet or Nwtl temporal logic formulas. As a further contribution, we present a different synthesis algorithm that improves on known solutions for large specifications and many exits.

1. Introduction

Recursive state machines (RSMs) model the control flow of systems with recursive procedure calls [2]. A recursive state machine is composed of a set of modules, whose vertices can be standard vertices or can correspond to invocations of other modules. A large number of hardware and software systems fit into this class, such as procedural and object-oriented programs, distributed systems, communication protocols and web services.

In the open systems setting, i.e., systems where an execution depends on the interaction of the system with the environment, the natural counterpart of recursive state machines is two-player recursive game graphs. A recursive game graph (RGG) is essentially a recursive state machine where vertices are split into two sets each controlled by one of the players, and thus corresponds to pushdown games with an emphasis on the modules composing the system.

In this paper we focus on solving games on RGG in which the first player is restricted to *modular strategies* [3]. A strategy is a mapping that specifies, for each play ending in a controlled state, the next move. Modular strategies are formed of a set of strategies, one for each RGG module, that are *local* to a module and *oblivious* of the history of previous module activations, i.e., the next move in such strategies is determined by looking only at the local memory of the current module activation (by contrast, if one allows the local memory to *persist* across module activations, deciding these games becomes undecidable already with reachability specifications [3]).

[☆] This work was partially funded by the MIUR grant FARB 2013–2014–2015 from Università degli Studi di Salerno (Italy) and INDAM-GNCS 2015 grant.
[☆☆] A preliminary version of this paper has appeared as [1].

[*] Corresponding author.
E-mail addresses: ilaria.decrescenzo@dia.unisa.it (I. De Crescenzo), slatorre@unisa.it (S. La Torre), yaron172@yahoo.com (Y. Velner).

http://dx.doi.org/10.1016/j.ic.2016.07.007
0890-5401/© 2016 Elsevier Inc. All rights reserved.

The main motivation for considering modular strategies is related to the synthesis of controllers [4,5]: given a description of the system where some of the choices depend upon the input and some represent uncontrollable internal non-determinism, the goal is to design a controller that supplies inputs to the system such that it satisfies a given correctness specification. Synthesizing a controller thus corresponds to computing winning strategies in two-player games, and a modular strategy would correspond to a modular controller. Note that the synthesized modular controller must operate with no assumptions on the uncontrollable nondeterminism of the system. Analogously, in our games we only require the strategy of one player to be modular and no restrictions are placed on the strategy of the other player.

The notion of modular strategy is also of independent interest and has recently found application in other contexts, such as, the automatic transformation of programs for ensuring security policies in privilege-aware operating systems [6], and a general formulation of the synthesis problem from libraries of open components [7].

The problem of deciding the existence of a modular strategy in a recursive game graph has been already studied with respect to ω-regular specifications. The problem is known to be NP-complete for reachability specifications [8], Exptime-complete for specifications given as deterministic and universal Büchi or Co-Büchi automata, and 2Exptime-complete for Ltl specifications [9].

In this paper, we study this problem with respect to several classes of specifications that can be expressed as pushdown automata. We show that in the general case, i.e., by allowing any pushdown automaton as a specification, the problem is undecidable. For this, we give a reduction from the problem of checking the emptiness of the intersection of deterministic context-free languages. We thus focus on visibly pushdown automata (VPA) [10] specifications with Büchi or co-Büchi acceptance. In the following, we refer to this problem as the MVPG (*Modular Visibly Pushdown Game*) problem and leaving the acceptance condition unspecified.

Our main contributions are:

- We show a polynomial time reduction from the MVPG problem with deterministic or universal VPA specifications to recursive modular games over ω-regular specifications. By [9], we get that this problem is Exptime-complete. We then use this result to show the membership to 2Exptime for the MVPG problem with nondeterministic VPA specifications.
- We also give a new algorithm for recursive games with finite-state automata specifications. Denoting with P the VPA and with G the RGG, our approach yields an upper bound of $|G|\,2^{O(d^2(k+\log d)+\beta)}$ for the MVPG problem, where d is the number of P states, k is the number of G exits, and β is the number of *call edges* of G. The known solution [9] yields an $|G|\,2^{O(kd^2\log(kd))}$ upper bound. Thus, our solution is faster when k and d are large, and matches the known Exptime lower bound [9]. In addition we use one-way nondeterministic/universal tree automata instead of two-way alternating tree automata, thus we explicitly handle aspects that are hidden in the construction from [9].
- We show that when the winning condition is expressed as a formula of the temporal logics CaRet [11] and Nwtl [12] (that subsume logic Ltl [13]) the MVPG problem is 2Exptime-complete, and hardness can be shown also for very simple fragments of the logics. In particular, we show a 2Exptime lower bound for the fragment containing only conjunctions of disjunctions of bounded-size path formulas (i.e., formulas expressing either the requirement that a given finite sequence be a subsequence of a word or its negation), that is in contrast with the situation in finite game graphs where Pspace-completeness holds for larger significant fragments (see [14]). On the positive side, we are able to show an exponential-time algorithm to decide the MVPG problem for specifications given as conjunctions of temporal logic formulas that can be translated into a polynomial-size VPA (such formulas include the path formulas).

Related work. Besides the already mentioned work that has dealt with modular games, but only for ω-regular specifications [9] or reachability [3], other research on pushdown games has focused on the general (i.e., non-modular) notion of winning strategy. We recall that determining the existence of a general winning strategy in pushdown games with reachability specifications is known to be Exptime-complete [15]. Deciding such games is 2Exptime-complete for nondeterministic visibly pushdown specifications and 3Exptime-complete for Ltl and CaRet specifications [16].

Synthesis from recursive-component libraries defines a pushdown game which is *orthogonal* to the MVPG problem: there the modules are already synthesized and the game is on the function calls. Deciding such games for Nwtl is 2Exptime-complete [17]. Synthesis from open recursive-component libraries combines both this synthesis problem and MVPG synthesis. Deciding the related game problem is Exptime-complete for reachability [7], and has the same computational complexity as MVPG synthesis for VPA, CaRet and Nwtl specifications [18]. Other synthesis problems dealing with compositions of component libraries are [19,20], where modules are represented as transducers.

Modular synthesis can also be captured by a more general setting studied in [21], where a module can be expressed as a term of the λY-calculus. Weaker forms of modular strategies in pushdown games with temporal logic specifications have been considered in [22]. A decidable formulation of the synthesis of programs with recursive function calls is studied [23]. Algorithmic synthesis of nonterminating reactive programs has been studied in several papers (see [5,24] and references therein).

2. Preliminaries

Given two positive integers i and j, $i \le j$, we denote with $[i, j]$ the set of integers k with $i \le k \le j$, and with $[j]$ the set $[1, j]$.