Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Visibly linear dynamic logic ☆

Alexander Weinert *, Martin Zimmermann

Reactive Systems Group, Saarland University, 66123 Saarbrücken, Germany

ARTICLE INFO

Article history:
Received 17 May 2017
Received in revised form 27 February 2018
Accepted 15 June 2018
Available online xxxx
Communicated by P. Aziz Abdulla

Keywords:
Temporal logic
Visibly pushdown languages
Satisfiability
Model checking
Infinite games

ABSTRACT

We introduce Visibly Linear Dynamic Logic (VLDL), which extends Linear Temporal Logic (LTL) by temporal operators that are guarded by visibly pushdown languages over finite words. In VLDL one can, e.g., express that a function resets a variable to its original value after its execution, even in the presence of an unbounded number of intermediate recursive calls. We prove that VLDL describes exactly the ω-visibly pushdown languages, i.e., that it is strictly more expressive than LTL and able to express recursive properties of programs with unbounded call stacks.

The main technical contribution of this work is a translation of VLDL into ω-visibly pushdown automata of exponential size via one-way alternating jumping automata. This translation yields exponential-time algorithms for satisfiability, validity, and model checking. We also show that visibly pushdown games with VLDL winning conditions are solvable in triply-exponential time. We prove all these problems to be complete for their respective complexity classes.

1. Introduction

Linear Temporal Logic (LTL) [2] is widely used for the specification of non-terminating systems. Its popularity is owed to its simple syntax and intuitive semantics, as well as to the so-called exponential compilation property, i.e., for each LTL formula there exists an equivalent Büchi automaton of exponential size. Due to the latter property, there exist algorithms for model checking in polynomial space and for solving infinite games in doubly-exponential time.

While LTL suffices to express properties of circuits and non-recursive programs with bounded memory, its application to real-life programs is hindered by its inability to express recursive properties. In fact, LTL is too weak to even express all ω-regular properties. There are several approaches to address the latter shortcoming by augmenting LTL, e.g., with regular expressions [3,4], finite automata on infinite words [5], and right-linear grammars [6]. We concentrate on the approach of Linear Dynamic Logic (LDL) [4], which guards the globally- and eventually-operators of LTL with regular expressions. While the LTL-formula $\mathbf{F}\psi$ simply means "Either now, or at some point in the future, ψ holds", the corresponding LDL operator $\langle r \rangle \psi$ means "Either now, or at some point in the future, ψ holds and the infix between these two points matches r".

The logic LDL captures the ω-regular languages. In spite of its greater expressive power, LDL still enjoys the exponential compilation property, hence there exist algorithms for model checking and solving infinite games in polynomial space and doubly-exponential time, respectively.

☆ Supported by the projects "TriCS" (ZI 1516/1-1) and "AVACS" (SFB/TR 14) of the German Research Foundation (DFG).
This is an extended and revised version of work first presented at FSTTCS '16 [1].

* Corresponding author.
E-mail addresses: weinert@react.uni-saarland.de (A. Weinert), zimmermann@react.uni-saarland.de (M. Zimmermann).

https://doi.org/10.1016/j.tcs.2018.06.030
0304-3975/

While the expressive power of LDL is sufficient for many specifications, it is still not sufficient to reason about recursive properties of systems. In order to address this shortcoming, we replace the regular expressions guarding the temporal operators with visibly pushdown languages (VPLs) [7] specified by visibly pushdown automata (VPAs) [7].

A VPA is a pushdown automaton that operates over a fixed partition of the input alphabet into calls, returns, and local actions. In contrast to classical pushdown automata, VPAs may only push symbols onto the stack when reading calls and may only pop symbols off the stack when reading returns. Moreover, they may not even inspect the topmost symbol of the stack when not reading returns. Thus, the height of the stack after reading a word is known a priori for all VPAs using the same partition of the input alphabet. Due to this, VPAs are closed under union and intersection, as well as complementation. The class of languages accepted by VPAs is known as visibly pushdown languages.

The class of such languages over infinite words, i.e., ω-visibly pushdown languages, are known to allow for the specification of many important properties in program verification such as "there are infinitely many positions at which at most two functions are active", which may, e.g., express repeated returns to a main-loop, or "every time the program enters a module m while p holds true, p holds true upon exiting m" [8]. The extension of VPAs to their variant operating on infinite words is, however, not well-suited to the specification of such properties in practice, as Boolean operations on such automata do not preserve the logical structure of the original automata. By guarding the temporal operators introduced in LDL with VPAs, VLDL allows for the modular specification of recursive properties while capturing ω-VPAs.

1.1. Our contributions

We begin with an introduction of VLDL and give examples of its use. We then provide translations from VLDL to VPAs over infinite words, so-called ω-VPAs, and vice versa. For the direction from logic to automata we translate VLDL formulas into one-way alternating jumping automata (1-AJA), which are known to be translatable into ω-VPAs of exponential size due to Bozzelli [9]. For the direction from automata to logic we use a translation of ω-VPAs into deterministic parity stair automata by Löding et al. [10], which we then translate into VLDL formulas. Afterwards, we compare and contrast VLDL and Visibly Linear Temporal Logic (VLTL), another logic capturing visibly pushdown languages. The logics VLDL and VLTL share the basic mechanism of guarding temporal operators with languages of finite words.

Secondly, we prove the satisfiability problem and the validity problem for VLDL to be ExpTime-complete. Membership in ExpTime follows from the previously mentioned constructions, while we show ExpTime-hardness of both problems by a reduction from the word problem for polynomially space-bounded alternating Turing machines adapting a similar reduction by Bouajjani et al. [11].

As a third result, we show that model checking visibly pushdown systems against VLDL specifications is ExpTime-complete as well. Membership in ExpTime follows from ExpTime-membership of the model checking problem for 1-AJAs against visibly pushdown systems. ExpTime-hardness follows from ExpTime-hardness of the validity problem for VLDL.

Moreover, solving visibly pushdown games with VLDL winning conditions is proven to be 3ExpTime-complete. Membership in 3ExpTime follows from the exponential translation of VLDL formulas into ω-VPAs and the membership of solving pushdown games against ω-VPA winning conditions in 2ExpTime due to Löding et al. [10]. 3ExpTime-hardness is due to a reduction from solving pushdown games against LTL specifications, again due to Löding et al. [10].

Finally, we show that replacing the visibly pushdown automata used as guards in VLDL by deterministic pushdown automata yields a logic with an undecidable satisfiability problem.

Our results show that VLDL allows for the concise specification of important properties in a logic with intuitive semantics. In the case of satisfiability and model checking, the complexity jumps from PSpace-completeness for LDL to ExpTime-completeness. For solving infinite games, the complexity gains an exponent moving from 2ExpTime-completeness to 3ExpTime-completeness.

We choose VPAs for the specification of guards in order to simplify arguing about the expressive power of VLDL. In order to simplify the modeling of ω-VPLs, other formalisms that capture VPLs over finite words may be used. We discuss one such formalism in the conclusion.

1.2. Related work

The need for specification languages able to express recursive properties has been identified before and there exist other approaches to using visibly pushdown languages over infinite words for specifications, most notably VLTL [12] and CaRet [8]. While VLTL captures the class of ω-visibly pushdown languages, CaRet captures only a strict subset of it. For both logics there exist exponential translations into ω-VPAs. In this work, we provide exponential translations from VLDL to ω-VPAs and vice versa. Hence, CaRet is strictly less powerful than VLDL, but every CaRet formula can be translated into an equivalent VLDL formula, albeit with a doubly-exponential blowup. Similarly, every VLTL formula can be translated into an equivalent VLDL formula and vice versa, with doubly-exponential blowup in both directions. For a fragment of VLDL, however, a translation with only exponential blowup exists. This fragment retains the expressiveness of the full logic. We discuss the connections between VLDL and VLTL in more detail in Section 6.

Other logical characterizations of visibly pushdown languages include characterizations by a fixed-point logic [9] and by monadic second order logic augmented with a binary matching predicate (MSO_μ) [7]. Even though these logics also capture the class of visibly pushdown languages, they feature neither an intuitive syntax nor intuitive semantics and thus are less applicable than VLDL in a practical setting.

Please cite this article in press as: A. Weinert, M. Zimmermann, Visibly linear dynamic logic, Theoret. Comput. Sci. (2018), https://doi.org/10.1016/j.tcs.2018.06.030