Theoretical Computer Science 578 (2015) 13–29

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Trimming visibly pushdown automata ☆

Mathieu Caralp, Pierre-Alain Reynier *, Jean-Marc Talbot

Laboratoire d'Informatique Fondamentale de Marseille, UMR, 7279, Aix-Marseille Université & CNRS, France

ARTICLE INFO

Article history:
Received 29 October 2013
Received in revised form 2 June 2014
Accepted 14 January 2015
Available online 19 January 2015

Keywords:
Visibly pushdown automata
Trimming

ABSTRACT

We study the problem of trimming visibly pushdown automata (VPA). We first describe a polynomial time procedure which, given a visibly pushdown automaton that accepts only well-nested words, returns an equivalent visibly pushdown automaton that is trimmed. We then show how this procedure can be lifted to the setting of arbitrary VPA. Furthermore, we present a way of building, given a VPA, an equivalent VPA which is both deterministic and trimmed. Last, our trimming procedures can be applied to weighted VPA.

1. Introduction

Visibly pushdown automata (VPA) are a particular class of pushdown automata defined over an alphabet split into call, internal and return symbols [2,3].[1] In VPA, the stack behavior is driven by the input word: when reading a call symbol, a symbol is pushed onto the stack, for a return symbol, the top symbol of the stack is popped, and for an internal symbol, the stack remains unchanged. VPA have been applied in research areas such as software verification (VPA allow one to model function calls and returns, thus avoiding the study of data flows along invalid paths) and XML documents processing (VPA can be used to model properties over words satisfying a matching property between opening and closing tags).

Languages defined by visibly pushdown automata enjoy many properties of regular languages such as (effective) closure by Boolean operations and these languages can always be defined by a deterministic visibly pushdown automaton. However, VPA do not have a unique minimal form [1]. Instead of minimization, one may consider trimming as a way to deal with smaller automata. Trimming a finite state automaton amounts to removing useless states, *i.e.* states that do not occur in some accepting computation of the automaton: every state of the automaton should be both reachable from an initial state, and co-reachable from a final state. This property is important from both a practical and a theoretical point of view. Indeed, most of the algorithmic operations performed on an automaton will only be relevant on the trimmed part of this automaton. Removing useless states may thus avoid the study of irrelevant paths in the automaton, and speed up the analysis. From a theoretical aspect, there are several results holding for automata provided they are trimmed. For instance, the boundedness of finite-state automata with multiplicities can be characterized by means of simple patterns for trimmed automata (see [14,9]). Similarly, Choffrut introduced in [8] the twinning property to characterize sequentiality of (trimmed) finite-state transducers. This result was later extended to weighted finite-state automata in [6]. Both of these results have been extended to visibly pushdown automata and transducers in [7] and [10] respectively, requiring these objects to be trimmed.

☆ This work has been partly supported by the project "Étude et Conception de Systèmes avec Perturbations" funded by the ANR (ANR-09-JCJC-0069) and by the project "Synthesis of Stream Processors" funded by the CNRS.

* Corresponding author.
E-mail addresses: mathieu.caralp@lif.univ-mrs.fr (M. Caralp), pierre-alain.reynier@lif.univ-mrs.fr (P.-A. Reynier), jean-marc.talbot@lif.univ-mrs.fr (J.-M. Talbot).

[1] These automata were first introduced in [5] as "input-driven automata".

http://dx.doi.org/10.1016/j.tcs.2015.01.018
0304-3975/

While trimming finite state automata can be done easily in linear time by solving two reachability problems in the graph representing the automaton, the problem is much more involved for VPA (and for pushdown automata in general). Indeed, in this setting, the current state of a computation (called a configuration) is given by both a "control" state and a stack content. A procedure has been presented in [12] for pushdown automata. It consists in computing, for each state, the regular language of stack contents that are both reachable and co-reachable, and using this information to constrain the behaviors of the pushdown automaton in order to trim it. This approach has however an exponential time complexity.

Contributions. In this work, we present a procedure for trimming visibly pushdown automata. The running time of this procedure is bounded by a polynomial in the size of the input VPA. We first tackle the case of VPA recognizing only so-called *well-nested* words, *i.e.* words which have no unmatched call or return symbols. This class of VPA is called well-nested VPA, and denoted by wnVPA. We actually present a construction for reducing wnVPA, *i.e.* ensuring that every run starting from an initial configuration can be completed into an accepting run. Symmetrically, we define a construction for co-reduction acting in a dual fashion. Combination of both constructions yields a trimming procedure. In a second step, we address the general case. To do so, we present a construction which modifies a VPA in order to obtain a wnVPA. This construction has to be reversible, in order to recover the original language, and to be compatible with the trimming procedure. In addition, we also design this construction in such a way that it allows to prove the following result: given a VPA, we can effectively build an equivalent VPA which is both deterministic and trimmed. Moreover, when considering deterministic inputs, we prove that there is no trimming procedure running in polynomial time while preserving determinism. Finally, we show that our constructions can be applied to weighted VPA.

Organization of the paper. In Section 2 we introduce definitions. We address the case of well-nested VPA in Section 3 and the general case in Section 4. We consider the issue of determinization in Section 5 and the extension to weighted VPA in Section 6. Last, a summary of our results is presented in Section 7.

Related models. VPA are tightly connected to several models:

Tree automata: It is shown in [2] that VPA and tree automata are equivalent models and actually recognize precisely the set of regular (ranked) tree languages, using the encoding of so-called *stack-trees*, rather similar to the first-child next-sibling encoding. Trimming ranked tree automata is standard (and can be performed in linear time), and one can wonder whether this approach could yield a polynomial time trimming procedure for VPA. But this is not possible as trimming the stack-tree encoding of a VPA, and then translating back the result into a VPA yields an automaton which is reduced but not trimmed. To figure this out we have to look at the construction of [2] transforming a stack-tree automaton into an equivalent VPA: intuitively, the produced VPA memorizes the state to be reached at the end of the current hedge (a hedge is a well-matched word surrounded by a call letter and a return letter). This is sufficient to ensure the reduction of the VPA, but not the co-reduction. In order to do so we would need also to memorize the state reached at the beginning of the current hedge. This is actually what is done in the construction we present for the co-reduction.

Context-free grammars: It is well-known that pushdown automata are equivalent to context-free grammars, and there exist classical procedures to construct a context-free grammar from a pushdown automaton producing the same language, and conversely (see [13]). It is possible to trim a context-free grammar in linear time, and as for tree automata it could possibly induce a trimming procedure for pushdown automata. But this may not respect the structure of the alphabet. In the special case of VPA the construction transforming a pushdown automaton into a context-free grammar only produces grammars in quadratic Greibach normal form. As suggested in [15] it is possible in this case to construct an equivalent VPA. However, as for tree automata, one can observe that the resulting VPA is reduced but not co-reduced.

Nested word automata [3] (NWA): Words over a structured alphabet can be seen as words equipped with a matching relation. NWA operate on such inputs and are equally expressive to VPA. In [3] R. Alur and P. Madhusudan present a construction recognizing the set of prefixes of a language of nested words. In order to do that, as a side effect, the construction reduces the NWA. This procedure is very similar to the one presented in this paper, and with minor modifications it can lead to a procedure reducing VPA. In NWA, The property of being co-reduced can be seen as the dual of the property of being reduced. In contrast, the two properties are not dual in VPA, as an initial configuration have an empty stack while final configuration do not. As a consequence, being co-reduced in VPA is incomparable with being co-reduced in NWA, and one cannot derive from the construction presented in [3] a procedure for co-reducing VPA.

2. Definitions

2.1. Preliminaries

Words and well-nested words. A *structured alphabet* Σ is a finite set partitioned into three disjoint sets Σ_c, Σ_r and Σ_ι, denoting respectively the *call*, *return* and *internal* alphabets. We denote by Σ^* the set of words over Σ and by ε the empty word.

The set of *well-nested* words Σ^*_{wn} is the smallest subset of Σ^* such that $\varepsilon \in \Sigma^*_{\mathsf{wn}}$ and for all $a \in \Sigma_\iota$, $c \in \Sigma_c$, $r \in \Sigma_r$ and all $u, v \in \Sigma^*_{\mathsf{wn}}$, $au \in \Sigma^*_{\mathsf{wn}}$ and $curv \in \Sigma^*_{\mathsf{wn}}$.