Theoretical Computer Science 625 (2016) 1–24

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

A complete refinement procedure for regular separability of context-free languages

Graeme Gange [a], Jorge A. Navas [b], Peter Schachte [a], Harald Søndergaard [a,*], Peter J. Stuckey [a]

[a] Department of Computing and Information Systems, The University of Melbourne, Vic. 3010, Australia
[b] NASA Ames Research Center, Moffett Field, CA 94035, USA

ARTICLE INFO

Article history:
Received 17 December 2014
Received in revised form 14 December 2015
Accepted 12 January 2016
Available online 27 January 2016
Communicated by D. Sannella

Keywords:
Abstraction refinement
Context-free languages
Regular approximation
Separability

ABSTRACT

Often, when analyzing the behaviour of systems modelled as context-free languages, we wish to know if two languages overlap. To this end, we present a class of semi-decision procedures for regular separability of context-free languages, based on counter-example guided abstraction refinement. We propose two effective instances of this approach, one that is complete but relatively expensive, and one that is inexpensive and sound, but for which we do not have a completeness proof. The complete method will prove disjointness whenever the input languages are regularly separable. Both methods will terminate whenever the input languages overlap. We provide an experimental evaluation of these procedures, and demonstrate their practicality on a range of verification and language-theoretic instances.

1. Introduction

We address the problem of checking whether two given context-free languages L_1 and L_2 are disjoint. This is a fundamental language-theoretical problem. It is of interest in many practical tasks that call for some kind of automated reasoning about programs. This can be because program behaviour is modelled using context-free languages, as in software verification approaches that try to capture a program's control flow as a (pushdown-system) path language. Or it can be because we wish to reason about string-manipulating programs, as is the case in software vulnerability detection problems, where various kinds of injection attack have to be modelled.

The problem of context-free disjointness is of course undecidable, but semi-decision procedures exist for non-disjointness. For example, one can systematically generate strings w over the intersection $\Sigma_1 \cap \Sigma_2$, where Σ_1 is the alphabet of L_1 and Σ_2 is that of L_2. If some w belongs to both L_1 and L_2, answer "yes, the languages overlap." It follows that no semi-decision procedure exists for disjointness. However, semi-decision procedures exist for the stronger property of being separable by a regular language. For example, one can systematically generate (representations of) regular languages over $\Sigma_1 \cup \Sigma_2$, and, if some such language R is found to satisfy $L_1 \subseteq R \wedge L_2 \subseteq \overline{R}$, answer "yes, the languages L_1 and L_2 are disjoint".

* Corresponding author.
E-mail addresses: gkgange@unimelb.edu.au (G. Gange), jorge.a.navaslaserna@nasa.gov (J.A. Navas), schachte@unimelb.edu.au (P. Schachte), harald@unimelb.edu.au (H. Søndergaard), pstuckey@unimelb.edu.au (P.J. Stuckey).

http://dx.doi.org/10.1016/j.tcs.2016.01.026
0304-3975/

A radically different approach, which we will follow here, uses so-called *counter-example guided abstraction refinement (CEGAR)* [6] of regular over-approximations. The scheme is based on repeated approximation refinement, as follows:

1. *Abstraction:* Compute *regular approximations* R_1 and R_2 such that $L_1 \subseteq R_1$ and $L_2 \subseteq R_2$. (Here R_1 and R_2 are regular languages, represented using regular expressions or finite-state automata.)
2. *Verification:* Check whether the intersection of R_1 and R_2 is empty using a decision procedure for regular languages. If $R_1 \cap R_2 = \emptyset$ then $L_1 \cap L_2 = \emptyset$, so answer "the languages are disjoint." If $w \in (R_1 \cap R_2)$, $w \in L_1$, and $w \in L_2$ then $L_1 \cap L_2 \neq \emptyset$, so answer "the languages overlap" and provide w as a witness. Otherwise, go to step 3.
3. *Refinement:* Produce new regular approximations R_1' and R_2' such that $L_1 \subseteq R_1' \subseteq R_1$, $L_2 \subseteq R_2' \subseteq R_2$, and $R_i' \subset R_i$ for some $i \in \{1, 2\}$. Tighten the approximations by performing the assignments $R_1, R_2 \leftarrow R_1', R_2'$; then go to step 2.

For the abstraction step, note that regular approximations exist, trivially. For the verification step, we could also take advantage of the fact that the class of context-free languages is closed under intersection with regular languages; however, this does not eliminate the need for a refinement procedure. For the refinement step, note that there is no indication of *how* the tightening of approximations should be done; indeed that is the focus of this paper. The step is clearly well-defined since, if $L \subset R$, where R is regular, there is always a regular language $R' \subset R$ such that $L \subseteq R'$.

For a given language L there may well be an infinite chain $R_1 \supset R_2 \supset \cdots \supset L$ of regular approximations. This is a source of possible non-termination of a CEGAR scheme. An interesting question therefore is: Are there refinement techniques that can guarantee termination at least when L_1 and L_2 are *regularly separable* context-free languages, that is, when there exists a regular language R such that $L_1 \subseteq R$ and $L_2 \subseteq \overline{R}$?

In this paper we answer this question in the affirmative. We propose a refinement procedure which can ensure termination of the CEGAR-based loop assuming the context-free languages involved are regularly separable. In this sense we provide a refinement procedure which is *complete* for regularly separable context-free languages. Of course the question of regular separability of context-free languages is itself undecidable [16]. The method we propose will also successfully terminate whenever the given languages overlap.

The method has been implemented in the form of a tool called COVENANT [10]. This tool is publicly available at https://github.com/sav-tools/covenant and is, as far as we know, the only publicly available implementation tackling the problem of (soundly) proving separation of context-free grammars.

Contribution The paper rests on regular approximation ideas by Nederhof [23] and we utilise the efficient *pre** algorithm [9] for intersecting (the language of) a context-free grammar with (that of) a finite-state automaton. We propose various ways to systematically "inflate" a word w in the context of a language L, that is, to enlarge $\{w\}$ to a (preferably infinite) superset without overlapping L. Based on such inflation techniques, we propose a novel refinement procedure for a CEGAR-like method to determine whether context-free languages are disjoint, and we prove the procedure complete for determining regular separability. In the context of regular approximation, where languages must be over-approximated using *regular* languages, separability is equivalent to regular separability, so the completeness means that the refinement procedure is optimal. On the practical side, the method has important applications in software verification and security analysis. We demonstrate its feasibility through an experimental evaluation of COVENANT.

Outline Section 2 introduces concepts, notation and terminology used in the paper. It also recapitulates relevant results about regular separability and language representations. Section 3 describes a CEGAR-based refinement procedure for separating context-free languages by *inflating* counterexamples into regular languages. Section 4 then describes a number of strategies for word inflation. Section 5 provides an example. In Section 6, we construct a proof that the procedure will terminate for any pair of regularly separable or intersecting languages. In Section 7 we place our method in context, comparing with previously proposed refinement techniques. In Section 8 we evaluate the method empirically, comparing COVENANT with the most closely related tool. Section 9 discusses more broadly related work, and Section 10 concludes. An appendix contains a description of the test cases used in the experimental evaluation.

2. Preliminaries

In this section we recall the some basics, including the notion of regular separability. Table 1 gives a glossary of notation used in the paper.

2.1. Regular and context-free languages

We first recall some basic formal-language concepts. These are assumed to be well understood—the only purpose here is to fix our terminology and notation. We shall be developing algorithms that, conceptually, manipulate formal languages, that is, sets of symbol strings. However, the algorithms in fact manipulate *representations* of languages, such as regular expressions or grammars, or language recognisers, such as finite-state automata. Hence we will be careful to distinguish objects such as automata or grammars, on the one hand, from their *denotations*. We shall use the function symbol $\mathcal{L}$ for the function that, applied to some object X, gives the language denoted/generated/recognised by X. In the case of regular