Journal of Logical and Algebraic Methods in Programming 85 (2016) 3–33

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp

Termination criteria for tree automata completion

Thomas Genet [1]

IRISA/INRIA/Université de Rennes 1, France

ARTICLE INFO

Article history:
Received 3 September 2014
Received in revised form 22 May 2015
Accepted 26 May 2015
Available online 10 June 2015

Keywords:
Term rewriting
Regular tree languages
Tree automata
Regularity preservation
Tree automata completion
Analysis of functional programs

ABSTRACT

This paper presents two criteria for the termination of tree automata completion. Tree automata completion is a technique for computing a tree automaton recognizing or over-approximating the set of terms reachable w.r.t. a term rewriting system. The first criterion is based on the structure of the term rewriting system itself. We prove that for most of the known classes of linear rewriting systems preserving regularity, the tree automata completion is terminating. Moreover, it outputs a tree automaton recognizing exactly the set of reachable terms. When the term rewriting system is outside of such classes, the set of reachable terms can be approximated using a set of equations defining an abstraction. The second criterion, which holds for any left-linear term rewriting system, defines sufficient restrictions on the set of equations for the tree automata completion to terminate. We then show how to take advantage of this second criterion to use completion as a new static analysis technique for functional programs. Some examples are demonstrated using the Timbuk completion tool.

1. Introduction

Tree automata completion is an algorithm for computing or approximating the set of reachable terms for a Term Rewriting System $\mathcal{R}$ (TRS for short). Computing or approximating the reachability of TRSs has found various applications ranging from static analysis [1,2] to cryptographic protocol verification [3,4] and to termination proofs of TRS [5]. Given an initial language recognized by a tree automaton $\mathcal{A}$ and a TRS $\mathcal{R}$, tree automata completion is able to produce a tree automaton $\mathcal{A}^*$ that recognizes or over-approximates $\mathcal{R}^*(\mathcal{L}(\mathcal{A}))$, *i.e.* all terms that can be reached by rewriting terms of $\mathcal{L}(\mathcal{A})$ with $\mathcal{R}$. Then, $\mathcal{A}^*$ can be used to show that terms recognized by another tree automaton $\mathcal{A}_{bad}$ are not reachable by rewriting terms of $\mathcal{L}(\mathcal{A})$ with $\mathcal{R}$, *i.e.* $\forall s \in \mathcal{L}(\mathcal{A})$, $\forall t \in \mathcal{L}(\mathcal{A}_{bad}) : s \not\rightarrow_{\mathcal{R}}^* t$. To show this, it is enough to check that $\mathcal{L}(\mathcal{A}^*) \cap \mathcal{L}(\mathcal{A}_{bad}) = \emptyset$, *i.e.* to compute the automaton recognizing the intersection between languages recognized by $\mathcal{A}^*$ and $\mathcal{A}_{bad}$ and check that it recognized an empty language. This has been used to verify the safety of protocols [3,4] and of Java programs [1].

A strength of the completion algorithm, and at the same time a weakness, is that its precision is parameterized by a function [6] or a set of equations [7]. It is a strength because tuning the approximation function (or equations) permits to adapt the precision of completion to a specific goal to tackle. This is what made it successful for program and protocol verification. On the other hand, this is a weakness because if the approximation is not strong enough termination is not guaranteed.

E-mail address: genet@irisa.fr.
[1] Tel.: +33 2 99 84 73 44; fax: +33 2 99 84 71 71.

http://dx.doi.org/10.1016/j.jlamp.2015.05.003
2352-2208/© 2015 Elsevier Inc. All rights reserved.

1.1. Contributions

In this paper, we define two sufficient conditions for the tree automata completion algorithm to terminate. This paper builds upon preliminary versions of two termination criteria defined in [8] and [9]. This is the result of a large research effort to make tree automata completion competitive with other program verification techniques. Tree automata completion was shown to efficiently handle the verification of complex infinite state systems [3,4,1] but there were few results about its termination. Thus, it was essentially used as a semi-algorithm for program verification, except in [4]. In this paper, we define *two conditions* for tree automata completion to terminate. Furthermore, we present some experiments showing that those two results open new ways to *precisely* analyze functional programs.

Condition on the TRS. The first termination condition is on the TRS. For most of the known classes of linear TRS for which the set of reachable terms is regular, completion is guaranteed to terminate and computes exactly the set of reachable terms. We here extend the previous results of [6] with classes defined by Toshinori Takai [10] and Pierre Réty [11]. We also show that completion produces equivalent, but more compact, tree automata than the other known algorithm of [12,10]. With regards to verification, this first result guarantees the precision of the verification: without approximation, tree automata completion is as precise as possible.

Condition on the set of equations. When approximations are used, they are defined using a set of equations E and a theorem of [7] guarantees that completion introduces no more approximation than what E defines. The second termination condition is on the set of equations E. This condition is essentially syntactic and simple to check.

Alternative static analysis technique for functional programs. Finally, we show that all these results can be used to achieve efficient and *precise* static analysis of functional programs translated into TRS. Using Timbuk [13], which implements completion and equational simplification, we give some examples showing that the obtained technique provides an interesting alternative static analysis technique for functional programs.

1.2. Outline

The outline of this paper is the following. We first introduce some background material on TRS and tree automata. In Section 3, we recall the tree automata completion algorithm and prove a new precision result on it in Section 4. In Section 5, we survey most of the known TRS classes preserving regularity. Then, we show that for most of the known linear classes, tree automata completion terminates and computes exactly the set of reachable terms. This is the first termination criterion: if the TRS is inside one of those classes the tree automata completion terminates. For TRS laying outside of those classes, in Section 6, we define a second criterion based on the structure of the set of approximation equations. This condition is quite restrictive but can be refined into a nicer criterion when the TRS under consideration encodes a functional program. We conclude this part with some experiments showing that this last criterion transforms tree automata completion into an alternative static analysis technique for functional programs. In Section 7, we compare the results presented in this paper with related work. Finally, in Section 8 we conclude and give some further research directions.

2. Background

In this section, we introduce some definitions and concepts that will be used throughout the rest of the paper (see also [14,15]). Let $\mathcal{F}$ be a finite set of symbols, each associated with an arity function, and let $\mathcal{X}$ be a countable set of *variables*. $\mathcal{T}(\mathcal{F},\mathcal{X})$ denotes the set of *terms* and $\mathcal{T}(\mathcal{F})$ denotes the set of *ground terms* (terms without variables). The set of variables of a term t is denoted by $\mathcal{V}ar(t)$. A *substitution* is a function σ from $\mathcal{X}$ into $\mathcal{T}(\mathcal{F},\mathcal{X})$, which can be uniquely extended to an endomorphism of $\mathcal{T}(\mathcal{F},\mathcal{X})$. A *position* p for a term t is a finite word over $\mathbb{N}$. The empty sequence λ denotes the top-most position. The set $\mathcal{P}os(t)$ of positions of a term t is inductively defined by $\mathcal{P}os(t)=\{\lambda\}$ if $t\in\mathcal{X}$ or t is a constant and $\mathcal{P}os(f(t_1,\dots,t_n))=\{\lambda\}\cup\{i.p \mid 1\le i\le n \text{ and } p\in\mathcal{P}os(t_i)\}$ otherwise. If $p\in\mathcal{P}os(t)$, then $t(p)$ denotes the symbol at position p in t, $t|_p$ denotes the subterm of t at position p, and $t[s]_p$ denotes the term obtained by replacing the subterm $t|_p$ at position p by the term s. The top symbol of a term is $Root(t)=t(\lambda)$.

A *term rewriting system* (TRS) $\mathcal{R}$ is a set of *rewrite rules* $l\to r$, where $l,r\in\mathcal{T}(\mathcal{F},\mathcal{X})$, $l\notin\mathcal{X}$, and $\mathcal{V}ar(l)\supseteq\mathcal{V}ar(r)$. A rewrite rule $l\to r$ is *left-linear* (resp. right-linear) if each variable occurs only once in l (resp. r). A TRS $\mathcal{R}$ is left-linear (resp. right-linear) if every rewrite rule $l\to r$ of $\mathcal{R}$ is left-linear (resp right-linear). A TRS $\mathcal{R}$ is said to be linear iff $\mathcal{R}$ is left-linear and right-linear. The TRS $\mathcal{R}$ induces a rewriting relation $\to_{\mathcal{R}}$ on terms as follows. Let $s,t\in\mathcal{T}(\mathcal{F},\mathcal{X})$ and $l\to r\in\mathcal{R}$, $s\to_{\mathcal{R}} t$ denotes that there exists a position $p\in\mathcal{P}os(s)$ and a substitution σ such that $s|_p=l\sigma$ and $t=s[r\sigma]_p$. The set of term irreducible by a TRS $\mathcal{R}$ is $\textsc{Irr}(\mathcal{R})$. The reflexive transitive closure of $\to_{\mathcal{R}}$ is denoted by $\to^*_{\mathcal{R}}$. The set of $\mathcal{R}$-descendants of a set of ground terms I is $\mathcal{R}^*(I)=\{t\in\mathcal{T}(\mathcal{F}) \mid \exists s\in I \text{ s.t. } s\to^*_{\mathcal{R}} t\}$.

An *equation set* E is a set of *equations* $l=r$, where $l,r\in\mathcal{T}(\mathcal{F},\mathcal{X})$. The relation $=_E$ is the smallest congruence such that for all equations $l=r$ of E and for all substitutions σ we have $l\sigma =_E r\sigma$. The set of equivalence classes defined by $=_E$ on $\mathcal{T}(\mathcal{F})$ is noted $\mathcal{T}(\mathcal{F})/_{=_E}$. Given a TRS $\mathcal{R}$ and a set of equations E, a term $s\in\mathcal{T}(\mathcal{F})$ is rewritten modulo E into $t\in\mathcal{T}(\mathcal{F})$, denoted $s\to_{\mathcal{R}/E} t$, if there exist $s'\in\mathcal{T}(\mathcal{F})$ and $t'\in\mathcal{T}(\mathcal{F})$ such that $s=_E s'\to_{\mathcal{R}} t'=_E t$. The reflexive transitive closure $\to^*_{\mathcal{R}/E}$ of $\to_{\mathcal{R}/E}$ is defined as usual except that reflexivity is extended to terms equal modulo E, *i.e.* for all $s,t\in\mathcal{T}(\mathcal{F})$ if $s=_E t$ then $s\to^*_{\mathcal{R}/E} t$. The set of $\mathcal{R}$-descendants modulo E of a set of ground terms I is $\mathcal{R}^*_E(I)=\{t\in\mathcal{T}(\mathcal{F}) \mid \exists s\in I \text{ s.t. } s\to^*_{\mathcal{R}/E} t\}$.