Information Processing Letters 139 (2018) 18–23

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Constant runtime complexity of term rewriting is semi-decidable

Florian Frohn, Jürgen Giesl *

LuFG Informatik 2, RWTH Aachen University, Germany

ARTICLE INFO

Article history:
Received 23 April 2018
Received in revised form 22 June 2018
Accepted 27 June 2018
Available online xxxx
Communicated by José Luiz Fiadeiro

Keywords:
Computational complexity
Decidability
Formal methods
Program correctness
Term rewriting

ABSTRACT

We prove that it is semi-decidable whether the runtime complexity of a term rewrite system is constant. Our semi-decision procedure exploits that constant runtime complexity is equivalent to termination of a restricted form of narrowing, which can be examined by considering finitely many start terms. We implemented our semi-decision procedure in the tool AProVE to show its efficiency and its success for systems where state-of-the-art complexity analysis tools fail.

1. Introduction

There are many techniques to infer upper bounds on the runtime complexity of term rewrite systems (TRSs) or closely related formalisms. As "runtime complexity" corresponds to the usual notion of program complexity, such techniques can be used to analyze the complexity of programs in real-world languages via suitable transformations [6,8,14]. Usually, complexity bounds are inferred to provide guarantees on a program's resource usage. But *constant* bounds are also important for detecting bugs, as constant-time algorithms cannot fully traverse their in- or output if it exceeds a certain size. Thus, if there is a constant bound for an algorithm which is supposed to traverse arbitrarily large data, then the algorithm is incorrect. To find such bugs in real programs, one would have to combine our results with corresponding transformations from programs to TRSs.

In this paper we prove that it is semi-decidable if the runtime complexity of a TRS is constant. A similar result is known for Turing Machines [12].[1] Note that in general there is no complexity-preserving transformation from one language to another, i.e., semi-decidability of constant bounds for one language does not imply semi-decidability for other Turing-complete languages (like term rewriting). After introducing preliminaries in Sect. 2, we present our semi-decision procedure in Sect. 3. Sect. 4 discusses related work and shows the efficiency of our procedure by evaluating our implementation in the tool AProVE [9].

2. Preliminaries

We recapitulate the main notions for TRSs [4]. $\mathcal{T}(\Sigma, \mathcal{V})$ is the set of *terms* over a finite signature Σ and the vari-

[1] However, Turing Machines and TRSs are inherently different. For example, Turing Machines only read a constant part of the input in constant time, whereas TRSs can copy the whole input in constant time using duplicating rules like $f(x) \to g(x, x)$. Similarly, TRSs can compare the whole input in constant time using non-left-linear rules like $f(x, x) \to g(x)$, etc.

* Corresponding author.
E-mail address: giesl@informatik.rwth-aachen.de (J. Giesl).

https://doi.org/10.1016/j.ipl.2018.06.012
0020-0190/

ables $\mathcal{V}$, where $\mathcal{V}(t)$ is the set of variables occurring in t and root(t) is the *root symbol* of a term $t \notin \mathcal{V}$. The *positions* $\text{pos}(t) \subset \mathbb{N}^*$ are $\{\varepsilon\}$ if $t \in \mathcal{V}$ and $\{\varepsilon\} \cup \bigcup_{i=1}^{k} \{i.\pi \mid \pi \in \text{pos}(t_i)\}$ if $t = f(t_1, \ldots, t_k)$. The subterm of t at position $\pi \in \text{pos}(t)$ is $t|_\pi = t$ if $\pi = \varepsilon$ and $t|_\pi = t_i|_{\pi'}$ if $\pi = i.\pi'$ and $t = f(t_1, \ldots, t_k)$. The term that results from replacing $t|_\pi$ with $s \in \mathcal{T}(\Sigma, \mathcal{V})$ is $t[s]_\pi$. The *size* of a term is $|x| = 1$ if $x \in \mathcal{V}$ and $|f(t_1, \ldots, t_k)| = 1 + \sum_{i=1}^{k} |t_i|$. A *TRS* $\mathcal{R}$ is a finite set of rules $\{\ell_1 \to r_1, \ldots, \ell_n \to r_n\}$ with $\ell_i, r_i \in \mathcal{T}(\Sigma, \mathcal{V})$, $\ell_i \notin \mathcal{V}$, and $\mathcal{V}(r_i) \subseteq \mathcal{V}(\ell_i)$ for all $1 \le i \le n$. The *rewrite relation* is defined as $s \to_\mathcal{R} t$ if there are $\pi \in \text{pos}(s)$, $\ell \to r \in \mathcal{R}$, and a substitution σ such that $s|_\pi = \ell\sigma$ and $t = s[r\sigma]_\pi$. Here, $\ell\sigma$ is the *redex* of the rewrite step. For two terms s and t, $s \to^n_\mathcal{R} t$ stands for a rewrite sequence $s = s_0 \to_\mathcal{R} s_1 \to_\mathcal{R} \cdots \to_\mathcal{R} s_{n-1} \to_\mathcal{R} s_n = t$ for some terms $s_1, \ldots, s_{n-1}$. The *defined* (resp. *constructor*) symbols of $\mathcal{R}$ are $\Sigma_d(\mathcal{R}) = \{\text{root}(\ell) \mid \ell \to r \in \mathcal{R}\}$ and $\Sigma_c(\mathcal{R}) = \Sigma \setminus \Sigma_d(\mathcal{R})$. A term $f(t_1, \ldots, t_k)$ is *basic* if $f \in \Sigma_d(\mathcal{R})$ and $t_1, \ldots, t_k$ are *constructor terms* (i.e., $t_1, \ldots, t_k \in \mathcal{T}(\Sigma_c(\mathcal{R}), \mathcal{V})$).

Example 1. The following TRS $\mathcal{R}$ is a variation of the example `SK90/4.51` from the *Termination Problems Data Base* (TPDB) [18] where two rules which are not reachable from basic terms were removed for the sake of clarity.

$$\mathsf{f}(\mathsf{a}) \to \mathsf{g}(\mathsf{h}(\mathsf{a})) \qquad \mathsf{h}(\mathsf{g}(x)) \to \mathsf{g}(\mathsf{h}(\mathsf{f}(x)))$$

We have $\Sigma_d(\mathcal{R}) = \{\mathsf{f}, \mathsf{h}\}$, $\Sigma_c(\mathcal{R}) = \{\mathsf{g}, \mathsf{a}\}$, and $x \in \mathcal{V}$. An example rewrite sequence (where the underlined subterms are the redexes) is

$$\underline{\mathsf{h}(\mathsf{g}(\mathsf{a}))} \to_\mathcal{R} \mathsf{g}(\mathsf{h}(\underline{\mathsf{f}(\mathsf{a})})) \to_\mathcal{R} \mathsf{g}(\underline{\mathsf{h}(\mathsf{g}(\mathsf{h}(\mathsf{a})))}) \to_\mathcal{R} \mathsf{g}(\mathsf{g}(\mathsf{h}(\mathsf{f}(\mathsf{h}(\mathsf{a})))))).$$

We now define the *runtime complexity* of a TRS $\mathcal{R}$. In the following definition, ω is the smallest infinite ordinal and hence, $\omega > n$ holds for all $n \in \mathbb{N}$. For any $M \subseteq \mathbb{N} \cup \{\omega\}$, $\sup M$ is the least upper bound of M.

Definition 2 *(Runtime complexity [10,11,15]).* The *derivation height* of a term t w.r.t. a relation $\to$ is the length of the longest sequence of $\to$-steps starting with t, i.e., $\text{dh}(t, \to) = \sup\{n \in \mathbb{N} \mid t' \in \mathcal{T}(\Sigma, \mathcal{V}), t \to^n t'\}$. Thus, $\text{dh}(t, \to) = \omega$ if t starts an infinite $\to$-sequence. The *runtime complexity* function $\text{rc}_\mathcal{R}$ maps any $m \in \mathbb{N}$ to the length of the longest $\to_\mathcal{R}$-sequence starting with a basic term whose size is at most m, i.e., $\text{rc}_\mathcal{R}(m) = \sup\{\text{dh}(t, \to_\mathcal{R}) \mid t \text{ is basic}, |t| \le m\}$.

Example 3. There is no longer $\to_\mathcal{R}$-sequence for $\mathsf{h}(\mathsf{g}(\mathsf{a}))$ than the one in Example 1, i.e., $\text{dh}(\mathsf{h}(\mathsf{g}(\mathsf{a})), \to_\mathcal{R}) = 3$. So $|\mathsf{h}(\mathsf{g}(\mathsf{a}))| = 3$ implies $\text{rc}_\mathcal{R}(3) \ge 3$. Our new approach proves $\text{rc}_\mathcal{R}(m) \in \mathcal{O}(1)$ automatically, i.e., $\mathcal{R}$ has constant runtime complexity.

So our goal is to check whether there is an $n \in \mathbb{N}$ such that all evaluations of basic terms take at most n steps. Our semi-decision procedure is based on *narrowing*, which is similar to rewriting, but uses unification instead of matching.

Definition 4 *(Narrowing).* A substitution σ is a *unifier* of $s, t \in \mathcal{T}(\Sigma, \mathcal{V})$ if $s\sigma = t\sigma$, and σ is the *most general unifier* (mgu) if every unifier has the form $\sigma \circ \theta$ for some substitution θ. A term s *narrows* to t ($s \overset{\sigma}{\underset{\pi}{\rightsquigarrow}}_\mathcal{R} t$) if there is a position $\pi \in \text{pos}(s)$ with $s|_\pi \notin \mathcal{V}$, a (variable-renamed) rule $\ell \to r \in \mathcal{R}$ with $\sigma = \text{mgu}(s|_\pi, \ell)$, and $t = s[r]_\pi\sigma$. We omit π or σ if they are irrelevant and write $s \overset{\sigma_1 \circ \cdots \circ \sigma_n}{\rightsquigarrow}{}^n_\mathcal{R} t$ or $s \rightsquigarrow^n_\mathcal{R} t$ if we have $s \overset{\sigma_1}{\rightsquigarrow}_\mathcal{R} \cdots \overset{\sigma_n}{\rightsquigarrow}_\mathcal{R} t$. A finite narrowing sequence $t_0 \overset{\sigma_1}{\rightsquigarrow}_\mathcal{R} \cdots \overset{\sigma_n}{\rightsquigarrow}_\mathcal{R} t_n$ is *constructor based* if $t_0\sigma_1 \cdots \sigma_n$ is a basic term. An infinite narrowing sequence is *constructor based* if all its finite prefixes are constructor based.

Example 5. $\mathcal{R}$ from Example 1 has the constructor-based narrowing sequence

$$\underline{\mathsf{h}(x)} \overset{\{x/\mathsf{g}(x')\}}{\underset{\varepsilon}{\rightsquigarrow}}_\mathcal{R} \mathsf{g}(\mathsf{h}(\underline{\mathsf{f}(x')})) \overset{\{x'/\mathsf{a}\}}{\underset{1.1}{\rightsquigarrow}}_\mathcal{R} \mathsf{g}(\underline{\mathsf{h}(\mathsf{g}(\mathsf{h}(\mathsf{a})))}) \overset{\varnothing}{\underset{1}{\rightsquigarrow}}_\mathcal{R} \mathsf{g}(\mathsf{g}(\mathsf{h}(\mathsf{f}(\mathsf{h}(\mathsf{a})))))).$$

3. Constant bounds for runtime complexity of term rewriting

For our semi-decision procedure, we will show that the runtime complexity of a TRS $\mathcal{R}$ is constant iff $\mathcal{R}$ has no infinite constructor-based narrowing sequence.

Example 6. To see why constructor-based narrowing terminates for $\mathcal{R}$ of Example 1, first consider sequences starting with basic terms of the form $\mathsf{h}(t)$. If $t \in \mathcal{V}$, then narrowing $\mathsf{h}(t)$ terminates after three steps, cf. Example 5. The same holds if $t = \mathsf{g}(t')$ with $t' \in \mathcal{V}$ or $t' = \mathsf{a}$. For other constructor terms t', narrowing $\mathsf{h}(\mathsf{g}(t'))$ terminates after one step. Finally, if $t \notin \mathcal{V}$ and root$(t) \neq \mathsf{g}$, then $\mathsf{h}(t)$ is a normal form w.r.t. $\rightsquigarrow_\mathcal{R}$. Now we consider basic start terms $\mathsf{f}(t)$. If $t \in \mathcal{V}$ or $t = \mathsf{a}$, then narrowing $\mathsf{f}(t)$ terminates after one step: $\mathsf{f}(t) \rightsquigarrow_\mathcal{R} \mathsf{g}(\mathsf{h}(\mathsf{a}))$. If $t \neq \mathsf{a}$ is a non-variable constructor term, then $\mathsf{f}(t)$ is a normal form w.r.t. $\rightsquigarrow_\mathcal{R}$. This covers all constructor-based narrowing sequences, i.e., $\mathcal{R}$'s runtime complexity is constant.

In contrast, if we change the second rule to $\mathsf{h}(\mathsf{g}(x)) \to \mathsf{g}(\mathsf{h}(x))$, then the runtime complexity becomes linear and constructor-based narrowing becomes non-terminating: $\underline{\mathsf{h}(x)} \overset{\{x/\mathsf{g}(x')\}}{\rightsquigarrow}_\mathcal{R} \mathsf{g}(\underline{\mathsf{h}(x')}) \overset{\{x'/\mathsf{g}(x'')\}}{\rightsquigarrow}_\mathcal{R} \mathsf{g}(\mathsf{g}(\underline{\mathsf{h}(x'')})) \rightsquigarrow_\mathcal{R} \cdots$.

Unfortunately, the reasoning in Example 6 is hard to automate, since it explicitly considers all sequences that start with any of the infinitely many basic terms. For automation, we show that constructor-based narrowing sequences can be "generalized" such that one only has to regard finitely many start terms. Then a semi-decision procedure for termination of constructor-based narrowing is obtained by enumerating only those sequences that begin with these start terms.

We first define a partial ordering $\gtrsim$ that clarifies which narrowing sequences are more general than others, and prove the equivalence between constant runtime and termination of constructor-based narrowing afterwards.

Definition 7 *(Ordering narrowing sequences).* Let $\mathcal{R}$ be a TRS and let $s_0 \overset{\sigma_1}{\underset{\pi_1}{\rightsquigarrow}}\mathcal{R}\ s_1 \overset{\sigma_2}{\underset{\pi_2}{\rightsquigarrow}}\mathcal{R} \cdots \overset{\sigma_n}{\underset{\pi_n}{\rightsquigarrow}}\mathcal{R}\ s_n$ and $t_0 \overset{\theta_1}{\underset{\pi_1}{\rightsquigarrow}}\mathcal{R}\ t_1 \overset{\theta_2}{\underset{\pi_2}{\rightsquigarrow}}\mathcal{R}$