Faculty of Mathematics, Computer Science and Natural Sciences Research Group Computer Science 2

RWTH Aachen University, Germany

Bachelor Thesis

Analyzing runtime complexity of non-ndg Term Rewrite Systems

submitted by

Simeon Valchev

Coming soon

Reviewers
Prof. Dr. Jürgen Giesl
TBD

Supervisor
Stefan Dollase, ...?

Statutory Declaration in Lieu of an Oath

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Contents

Chapter 1

Introduction

Term rewrite systems have been extensively studied, with the main problem concerning their complexity analysis. In recent decades many automated tools have been developed to infer upper bounds on runtime complexity (rc), which considers the longest rewrite sequence starting with a basic term disregarding any evaluation strategy. In contrast to rc, innermost runtime complexity (irc) makes use of an innermost evaluation strategy that can be much easier to analyse. While the analysis of irc has some uses, it is rc that is of greater interest.

Recently a technique has been developed to overapproximate some TRSs as non-dup generalized (ndg), with a corresponding proof of the equality of rc and irc for such ndg TRSs. The mentioned criteria combined with the much more powerful analysis of irc allowed inferring upper/lower bounds on rc from upper/lower bounds on irc. This method however makes no claims regarding all the TRSs that are non-ndg. It is the goal of this thesis to extend the technique to non-ndg TRSs.

Presented here is a transformation of the original non-ndg TRS into an encoded version, that shares the same rc. The goal of the encoding is to create a TRS, for which also holds that irc = rc. While the encoded version may not ndg, what is important is that we can apply the techniques used in the analysis of irc to infer the rc of the original non-ndg TRS.

As it will be shown, not all encodings are equal and some yield substantially better results. Two algorithms have been developed to improve the results of the automated analysis. The first - overapproximating the TRS positions that require encoding and a second - detecting when to alter the encoding to not add any non-terminating rules.

The structure of the thesis is as follows. Chapter 2 introduces some of the preliminary knowledge regarding TRSs. Chapter 3 contains a description of the first mentioned algorithm, which is used as the foundation, on which the encoding, described in Chapter 4, is built. Chapter 5 is dedicated to the second algorithm and the changes that come up from its result. In Chapter 6 are presented the results from the implementation of the technique in the tool AProVE and it also serves as a summary.

Chapter 2

Preliminaries

In this chapter is introduced some of the necessary groundwork on term rewrite systems with accompanying examples.

A term rewrite system (TRS) $\mathcal{R}$ is a finite set of rules $\ell \to r$, where ℓ and r are terms in the set $\mathcal{T}(\Sigma, \mathcal{V})$ over the signature Σ and the set of variables $\mathcal{V}$. In the context of a rule $\ell \to r$, ℓ refers to the left-hand side and r to the right-hand side. The rules describe the rewrite relations between the terms in the previously mentioned set, i.e. the left-hand side of a rule can be matched against some part of a term s, rewrite it to the matching right-hand side and give output t.

Example 1 (Addition)

$$\begin{aligned} \alpha_1 &: \mathsf{add}(\mathsf{x}, \mathsf{0}) &&\to \mathsf{x} \\ \alpha_2 &: \mathsf{add}(\mathsf{x}, \mathsf{s}(\mathsf{y})) &&\to \mathsf{add}(\mathsf{s}(\mathsf{x}), \mathsf{y}) \end{aligned}$$

In example 1 a simple TRS for the calculation of addition on natural numbers is presented. Numbers here are represented by the so-called successor function and the base symbol 0, i.e. 1 would be $\mathsf{s}(\mathsf{0})$, 2 - $\mathsf{s}(\mathsf{s}(\mathsf{0}))$ and so on. In this way our base case would be represented in α_1, where $x + 0 = x$. In α_2 the TRS recursively subtracts one from the second argument and adds it to the first argument. This repeats until the second argument reaches 0 and thus the first argument is output. One could intuitively imagine this as follows:

$$3 + 2 \quad = \quad 4 + 1 \quad = \quad 5 + 0 \quad = \quad 5$$

Chapter 3

Algorithm for finding non-ndg locations

Chapter 4

Encoding

Chapter 5

Encoding with terminating rules

Chapter 6

Results