Complexity Analysis of Term Rewrite Systems

Georg Moser

Institute of Computer Science
University of Innsbruck

IFIP WG 1.6, July 2, 2009

Overview

- The Fundamentals
- The Past
- The Present
- The Future

The Fundamentals

Definition

derivation length

$$\mathsf{dl}(t, \to) = \max\{n \mid \exists u\ t \to^n u\}$$

$$\mathsf{dl}(n, T, \to) = \max\{\mathsf{dl}(t, \to) \mid \exists t \in T \text{ and } |t| \leqslant n\}$$

Definition derivation length

$$\mathsf{dl}(t, \to) = \max\{n \mid \exists u\ t \to^n u\}$$
$$\mathsf{dl}(n, T, \to) = \max\{\mathsf{dl}(t, \to) \mid \exists t \in T \text{ and } |t| \leqslant n\}$$

Definition derivational complexity

$$\mathsf{dc}_{\mathcal{R}}(n) = \mathsf{dl}(n, \text{"all terms"}, \to_{\mathcal{R}})$$

Definition — derivation length

$$\mathsf{dl}(t, \to) = \max\{n \mid \exists u\ t \to^n u\}$$

$$\mathsf{dl}(n, T, \to) = \max\{\mathsf{dl}(t, \to) \mid \exists t \in T \text{ and } |t| \leqslant n\}$$

Definition — derivational complexity

$$\mathsf{dc}_{\mathcal{R}}(n) = \mathsf{dl}(n, \text{"all terms"}, \to_{\mathcal{R}})$$

Definition — runtime complexity

$$\mathsf{rc}_{\mathcal{R}}(n) = \mathsf{dl}(n, \text{"basic terms"}, \to_{\mathcal{R}})$$

Definition — derivation length

$$\mathsf{dl}(t, \to) = \max\{n \mid \exists u\ t \to^n u\}$$
$$\mathsf{dl}(n, T, \to) = \max\{\mathsf{dl}(t, \to) \mid \exists t \in T \text{ and } |t| \leqslant n\}$$

Definition — derivational complexity

$$\mathsf{dc}_{\mathcal{R}}(n) = \mathsf{dl}(n, \text{"all terms"}, \to_{\mathcal{R}})$$

Definition — runtime complexity

$$\mathsf{rc}_{\mathcal{R}}(n) = \mathsf{dl}(n, \text{"basic terms"}, \to_{\mathcal{R}})$$

term $f(t_1, \dots, t_n)$ is basic if

- f is defined
- $t_1, \dots, t_n$ contain no defined symbols

How To Analyse Complexity

$$t_1 \to_{\mathcal{R}} t_2 \to_{\mathcal{R}} t_3 \to_{\mathcal{R}} \dots$$

How To Analyse Complexity

$$t_1 \succ t_2 \succ t_3 \succ \dots \succ t_n$$

consider

1. $\exists$ reduction order $\succ$ such that
2. $\mathcal{R} \subseteq \succ$

How To Analyse Complexity

$$t_1 \to_{\mathcal{R}} t_2 \to_{\mathcal{R}} t_3 \to_{\mathcal{R}} \dots \to_{\mathcal{R}} t_n$$

consider

1. $\exists$ reduction order $\succ$ such that
2. $\mathcal{R} \subseteq \succ$

Observation

- $\succ$ can be used to measure the derivation length

How To Analyse Complexity

$$t_1 \to_{\mathcal{R}} t_2 \to_{\mathcal{R}} t_3 \to_{\mathcal{R}} \dots \to_{\mathcal{R}} t_n$$

consider

1. $\exists$ reduction order $\succ$ such that
2. $\mathcal{R} \subseteq \succ$

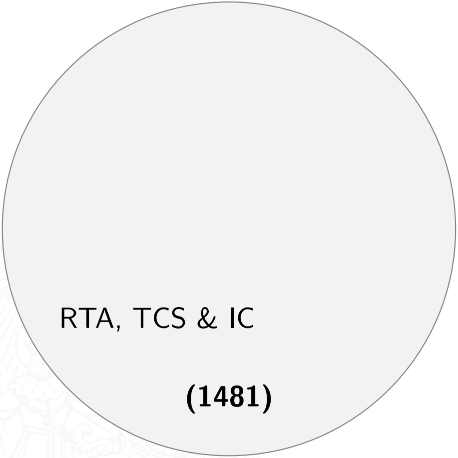
Observation

- $\succ$ can be used to measure the derivation length

The Past

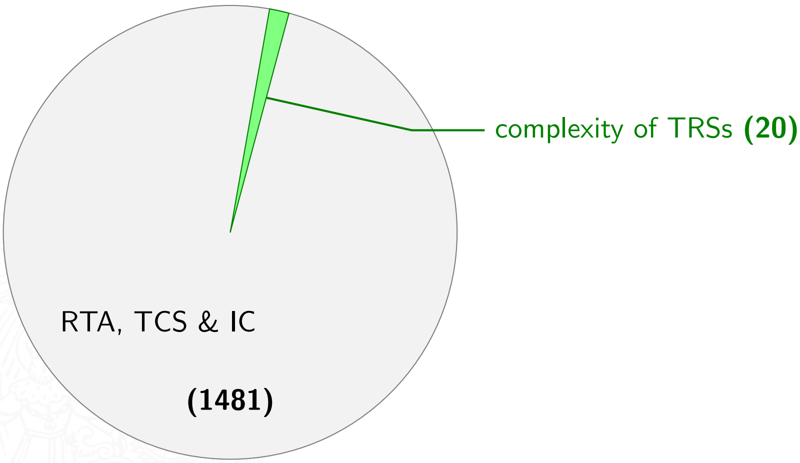
Complexity in Rewriting: A History

of papers



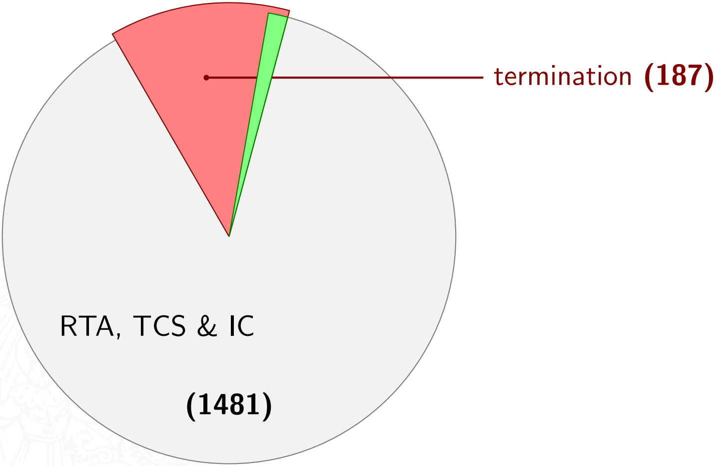
Complexity in Rewriting: A History

of papers



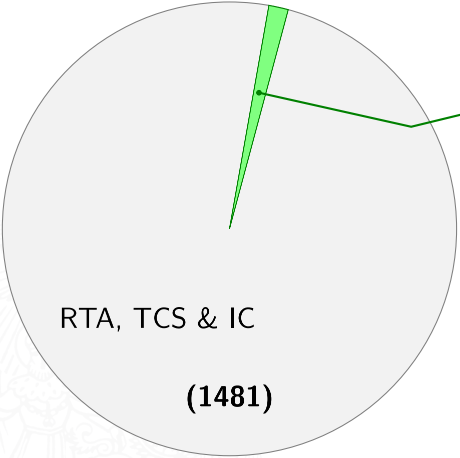
Complexity in Rewriting: A History

of papers



Complexity in Rewriting: A History

3 selected papers



ALGORITHMIC COMPLEXITY OF TERM REWRITING SYSTEMS

C. CHOPPY, S. KAPLAN, M. SORIA
Laboratoire de Recherche en Informatique
U.A. C.N.R.S. 410
Université Paris-Sud, Bât 490
F-91405 ORSAY Cédex, FRANCE
net : mcvax!inria!lri!cc, mcvax!inria!lri!kaplan, mcvax!inria!lri!soria

Introduction

Algebraic specifications are now widely used for data structuring and they turn out to be quite useful for various aspects of program development, such as prototyping, assisted program construction, proving properties, etc [BCV 85, FG 84, FGJM 84, GHW 85, Kap 86]. Some of these applications require to add a notion of computation to algebraic specifications, for instance by providing a (convergent) rewrite rule system that expresses the properties of the operators. In this context, it may be of first interest to define a notion of algorithmic complexity for an algebraic specification, or, more precisely, a notion of complexity for each operator defined in the specification. Computing operator complexity within a given specification helps understanding how evaluation costs are distributed ; it may point out "costly" operators, and motivate the search for an equivalent, but "cheaper", specification.

In [CLR 80], the cost of a term is defined as the number of rewriting steps for reducing it to its normal form, and the cost of an operator is defined as the general cost of a term obtained by applying this operator to terms in normal form. In this paper, we further formalize this notion of operator complexity and investigate its computation through analysis methods developed for instance in [Stey 84] and [Fla 87]. We show how these methods apply to the computation of the enumerative series over the terms of an algebraic specification. We define the notion of *regular* rewriting systems, and consider cost series of operators that are described by such systems. We show how these analysis methods apply to compute such costs and provide an asymptotic evaluation of the average cost of an operator. Our results allow costs to be computed without any explicit manipulation of series. We provide an eventual user with ready-to-use formulae, where the different parameters only depend on the "geometry" of the system, e.g. the number of constructors in the left handside of rules, number of occurrences of a derived operator in the right handside, etc.

Quantitative evaluation of rewriting systems had not yet been studied under such an approach (except in [CLR 80]), to our knowledge. From a different point of view, complexity of algebraic implementations has been studied in [BBWT 81, E&M 81, etc.] w.r.t. computability issues.

1. Introductory example

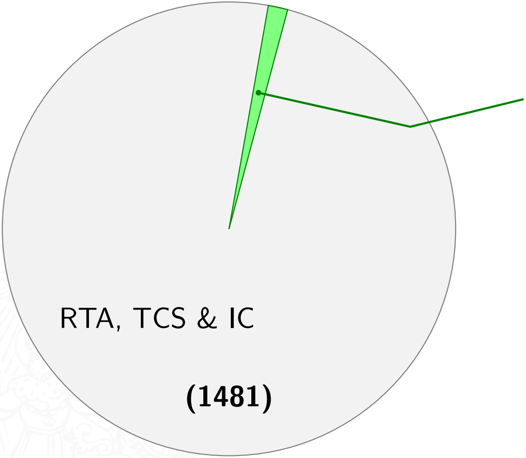
Let us assume that one wants to evaluate the average cost of a given computation on some data set. The data belong to a set of objects, a size can be computed for each object ; let D_n denote the set of

Algorithmic Complexity of Term Rewrite Systems; Choppy et al., RTA 1987

average cost analysis

Complexity in Rewriting: A History

3 selected papers



Termination proofs and the length of derivations
(Preliminary version)

Dieter Hofbauer, Technische Universität Berlin
Clemens Lautemann *, Universität Bremen

Abstract

The derivation height of a term t, relative to a set R of *rewrite rules*, $dh_R(t)$, is the length of a longest derivation from t. We investigate in which way certain termination proof methods impose bounds on dh_R. In particular we show that, if termination of R can be proved by *polynomial interpretation* then dh_R is bounded from above by a doubly exponential function[1], whereas termination proofs by *Knuth-Bendix ordering* are possible even for systems where dh_R cannot be bounded by any primitive recursive functions. For both methods, conditions are given which guarantee a singly exponential upper bound on dh_R. Moreover, all upper bounds are tight.

1 Introduction

The complexity of a term rewriting system R can be measured in different ways. In this paper, we consider the *derivation height*, expressed in the function Dh_R, where $Dh_R(n)$ is the length of a longest derivation which starts from a term of size n. Thus Dh_R describes the worst possible behaviour of R.

Obviously, Dh_R is partial recursive, and is total if and only if R is terminating. The rate of growth of Dh_R then quantifies termination of R in much the same way as the rate of convergence of a series (q_n) quantifies its convergence.

As termination is one of the basic concepts in term rewriting theory, crucial e.g. in Knuth-Bendix completion procedures or for inductive proofs, a lot of techniques have been developed to support or even automatically generate termination proofs (see [De87] for a survey).

Proving termination with one of these specific techniques in general proves more than just the absence of infinite derivation sequences. It turns out that such a proof in many cases implies an upper bound on Dh_R, so that the rate of growth of Dh_R can be used for measuring the power of termination proof methods.

A similar point of view is taken in [DO88], where Dershowitz and Okada investigate the order type of certain simplification orderings. However, the order type of such an ordering as such does not place a bound on its power for termation proofs, as it does not say anything about the intersection of the simplification ordering with the rewrite relation induced by a finite term rewriting system.

*in the early stages of this research supported by ESPRIT project PROSPECTRA, ref. #390

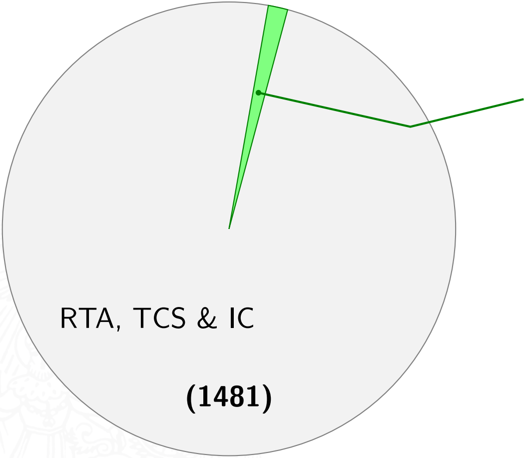
[1]this result was independently obtained by Oliver Geupel

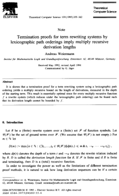
Termination Proofs and the Length of Derivations; Hofbauer, Lautemann, RTA 1989

to be continued

Complexity in Rewriting: A History

3 selected papers




ELSEVIER
Theoretical Computer Science 139 (1995) 355–362
Theoretical Computer Science

Note

Termination proofs for term rewriting systems by lexicographic path orderings imply multiply recursive derivation lengths

Andreas Weiermann
Institut für Mathematische Logik und Grundlagenforschung, Einsteinstr. 62, 48149 Münster, Germany

Received May 1992; revised April 1994
Communicated by G. Jäger

Abstract

It is shown that a termination proof for a term rewriting system using a lexicographic path ordering yields a multiply recursive bound on the length of derivations, measured in the depth of the starting term. This result is essentially optimal since for every multiply recursive function f a rewrite system (which reduces under the lexicographic path ordering) can be found such that its derivation length cannot be bounded by f.

0. Introduction

Let R be a (finite) rewrite system over a (finite) set $\mathscr{F}$ of function symbols. Let $\mathscr{G}(\mathscr{F})$ be the set of ground terms over $\mathscr{F}$. (We assume that $\mathscr{G}(\mathscr{F})$ is not empty.) For $m \in \mathbb{N}$ let

$$D(m) := \max\{n \in \mathbb{N} : (\exists t_1, \ldots, t_n \in \mathscr{G}(\mathscr{F}))[dp(t_1) \leqslant m \,\&\, t_1 \to_R \cdots \to_R t_n]\},$$

where $dp(t)$ denotes the depth of a term t and $\to_R$ denotes the rewrite relation induced by R. D is called the *derivation length function* for R. If $\mathscr{F}$ is finite and if R is finite and terminating, then D is a (total) recursive function.

In order to investigate the power as well as the limitations of different termination proof methods, it is natural to ask how long derivation sequences can be if a certain

Correspondence to: A. Weiermann, Institut für Mathematische Logik und Grundlagenforschung, Einsteinstr. 62, 48149 Münster, Germany. Email: weierma@math.uni-muenster.de.

Termination Proofs by Lexicographic Path Orders imply Multiply Recursive Derivation Lengths; Weiermann, TCS 1995

LPO simple, complexity wise

Termination Proofs and the Length of Derivations

- introduction of derivation length, derivational complexity
- derivational complexity as measure of a termination technique

Termination Proofs and the Length of Derivations

- introduction of derivation length, derivational complexity
- derivational complexity as measure of a termination technique

Theorem Hofbauer, Lautemann 1989

polynomial interpretations induce double-exponential derivational complexity

Termination Proofs and the Length of Derivations

- introduction of derivation length, derivational complexity
- derivational complexity as measure of a termination technique

Theorem Hofbauer, Lautemann 1989

polynomial interpretations induce double-exponential derivational complexity

Lemma ①

$\forall\ \mathcal{R}$ terminating via a polynomial interpretation
$\exists\ c \in \mathbb{R}$, $c > 0$ $\forall$ terms s: $\mathsf{dl}(s, \to_{\mathcal{R}}) \leqslant 2^{2^{c \cdot |s|}}$

Lemma ②

$\exists\ \mathcal{R}$ terminating via a polynomial interpretation
$\exists\ c \in \mathbb{R}$, $c > 0$ for infinitely many terms s: $\mathsf{dl}(s, \to_{\mathcal{R}}) \geqslant 2^{2^{c \cdot |s|}}$

Proof of Lemma ②

consider $\mathcal{R}_{\mathsf{hl}}$:

$$\begin{array}{lll} x+0 \to x & \mathsf{d}(0) \to 0 & \mathsf{d}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{s}(\mathsf{d}(x))) \\ x+\mathsf{s}(y) \to \mathsf{s}(x+y) & \mathsf{q}(0) \to 0 & \mathsf{q}(\mathsf{s}(x)) \to \mathsf{q}(x) + \mathsf{s}(\mathsf{d}(x)) \end{array}$$

Proof of Lemma ②

consider $\mathcal{R}_{\mathsf{hl}}$:

$$
\begin{array}{rll}
x+0 \to x & \mathsf{d}(0) \to 0 & \mathsf{d}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{s}(\mathsf{d}(x))) \\
x+\mathsf{s}(y) \to \mathsf{s}(x+y) & \mathsf{q}(0) \to 0 & \mathsf{q}(\mathsf{s}(x)) \to \mathsf{q}(x) + \mathsf{s}(\mathsf{d}(x))
\end{array}
$$

$$0_{\mathcal{A}} = 2 \quad \mathsf{s}_{\mathcal{A}}(n) = n + 1 \quad n +_{\mathcal{A}} m = n + 2m \quad \mathsf{d}_{\mathcal{A}}(n) = 3n \quad \mathsf{q}_{\mathcal{A}}(n) = n^3$$

Proof of Lemma ②

consider $\mathcal{R}_{\mathsf{hl}}$:

$$\begin{array}{lll} x + 0 \to x & \mathsf{d}(0) \to 0 & \mathsf{d}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{s}(\mathsf{d}(x))) \\ x + \mathsf{s}(y) \to \mathsf{s}(x + y) & \mathsf{q}(0) \to 0 & \mathsf{q}(\mathsf{s}(x)) \to \mathsf{q}(x) + \mathsf{s}(\mathsf{d}(x)) \end{array}$$

$$0_{\mathcal{A}} = 2 \quad \mathsf{s}_{\mathcal{A}}(n) = n + 1 \quad n +_{\mathcal{A}} m = n + 2m \quad \mathsf{d}_{\mathcal{A}}(n) = 3n \quad \mathsf{q}_{\mathcal{A}}(n) = n^3$$

1. s defines the successor function
2. d defines the doubling function, i.e., $\mathsf{d}(\mathsf{s}^n(0)) \xrightarrow{*} \mathsf{s}^{2n}(0)$
3. q defines the square function, i.e., $\mathsf{q}(\mathsf{s}^n(0)) \xrightarrow{*} \mathsf{s}^{n^2}(0)$

Proof of Lemma ②

consider $\mathcal{R}_{\mathsf{hl}}$:

$$x + 0 \to x \qquad \mathsf{d}(0) \to 0 \qquad \mathsf{d}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{s}(\mathsf{d}(x)))$$
$$x + \mathsf{s}(y) \to \mathsf{s}(x + y) \qquad \mathsf{q}(0) \to 0 \qquad \mathsf{q}(\mathsf{s}(x)) \to \mathsf{q}(x) + \mathsf{s}(\mathsf{d}(x))$$

$$0_{\mathcal{A}} = 2 \quad \mathsf{s}_{\mathcal{A}}(n) = n + 1 \quad n +_{\mathcal{A}} m = n + 2m \quad \mathsf{d}_{\mathcal{A}}(n) = 3n \quad \mathsf{q}_{\mathcal{A}}(n) = n^3$$

1. s defines the successor function
2. d defines the doubling function, i.e., $\mathsf{d}(\mathsf{s}^n(0)) \xrightarrow{*} \mathsf{s}^{2n}(0)$
3. q defines the square function, i.e., $\mathsf{q}(\mathsf{s}^n(0)) \xrightarrow{*} \mathsf{s}^{n^2}(0)$

from this we get:

$$s_m := \mathsf{q}^{m+1}(\mathsf{s}^2(0)) \xrightarrow{*} \mathsf{q}(\mathsf{s}^{2^{2^m}}(0)) \xrightarrow{\geqslant 2^{2^m}} \mathsf{s}^{2^{2^{m+1}}}(0)$$

Proof of Lemma ②

consider $\mathcal{R}_{\mathsf{hl}}$:

$$x+0 \to x \qquad \mathsf{d}(0) \to 0 \qquad \mathsf{d}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{s}(\mathsf{d}(x)))$$
$$x+\mathsf{s}(y) \to \mathsf{s}(x+y) \qquad \mathsf{q}(0) \to 0 \qquad \mathsf{q}(\mathsf{s}(x)) \to \mathsf{q}(x) + \mathsf{s}(\mathsf{d}(x))$$

$$0_{\mathcal{A}} = 2 \quad \mathsf{s}_{\mathcal{A}}(n) = n+1 \quad n +_{\mathcal{A}} m = n + 2m \quad \mathsf{d}_{\mathcal{A}}(n) = 3n \quad \mathsf{q}_{\mathcal{A}}(n) = n^3$$

1. s defines the successor function
2. d defines the doubling function, i.e., $\mathsf{d}(\mathsf{s}^n(0)) \xrightarrow{*} \mathsf{s}^{2n}(0)$
3. q defines the square function, i.e., $\mathsf{q}(\mathsf{s}^n(0)) \xrightarrow{*} \mathsf{s}^{n^2}(0)$

from this we get:

$$s_m := \mathsf{q}^{m+1}(\mathsf{s}^2(0)) \xrightarrow{*} \mathsf{q}(\mathsf{s}^{2^{2^m}}(0)) \xrightarrow{\geqslant 2^{2^m}} \mathsf{s}^{2^{2^{m+1}}}(0)$$

we conclude, for all $m \geqslant 1$

$$\mathsf{dl}(s_m, \to_{\mathcal{R}_{\mathsf{hl}}}) \geqslant 2^{2^m} = 2^{2^{|s_m|-4}} \geqslant 2^{2^{c \cdot |s_m|}}$$

where $c \leqslant \frac{1}{5}$ ∎

The Present

How can we improve this ?

How can we improve this ?

Goal ① modern

study the complexity induced by state-of-the-art termination techniques

How can we improve this ?

Goal ① modern

study the complexity induced by state-of-the-art termination techniques

Goal ② useful

induced complexity is bounded by functions of low computational complexity

How can we improve this ?

Goal ① modern

study the complexity induced by state-of-the-art termination techniques

... basic DP method based on LPO characterises the multiple recursive functions

Goal ② useful

induced complexity is bounded by functions of low computational complexity

How can we improve this ?

Goal ① modern

study the complexity induced by state-of-the-art termination techniques

... basic DP method based on LPO characterises the multiple recursive functions

Goal ② useful

induced complexity is bounded by functions of low computational complexity

... WDP method based on $\mathrm{POP}^*_{\mathsf{ps}}$ induces polytime computability

How can we improve this ?

Goal ① modern

study the complexity induced by state-of-the-art termination techniques

. . . basic DP method based on LPO characterises the multiple recursive functions

Goal ② useful

induced complexity is bounded by functions of low computational complexity

. . . WDP method based on $\mathrm{POP}^*_{\mathsf{ps}}$ induces polytime computability

Goal ③ automated

automated complexity analysis

How can we improve this ?

Goal ① modern

study the complexity induced by state-of-the-art termination techniques

... basic DP method based on LPO characterises the multiple recursive functions

Goal ② useful

induced complexity is bounded by functions of low computational complexity

... WDP method based on $\mathrm{POP}^*_{\mathsf{ps}}$ induces polytime computability

Goal ③ automated

automated complexity analysis

✓ `Tyrolean Complexity Tool`

How can we improve this ?

Goal ① modern

study the complexity induced by state-of-the-art termination techniques

... basic DP method based on LPO characterises the multiple recursive functions

Goal ② useful

induced complexity is bounded by functions of low computational complexity

... WDP method based on $\mathrm{POP}^*_{\mathsf{ps}}$ induces polytime computability

Goal ③ automated

automated complexity analysis

✓ `Tyrolean Complexity Tool`

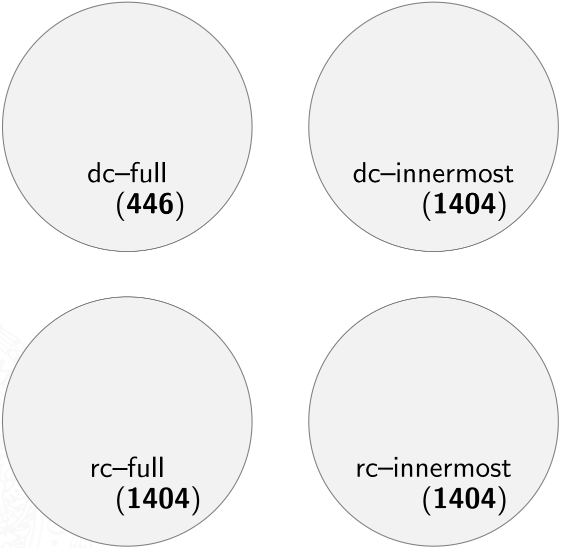
✓ `Complexity And Termination`

Automated Complexity Analysis: A Snapshot

polynomial derivation length on TPDB

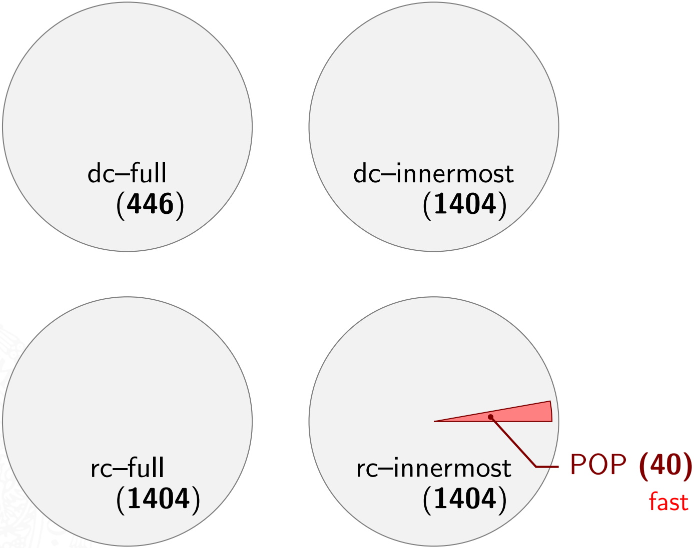
Automated Complexity Analysis: A Snapshot

polynomial derivation length on TPDB



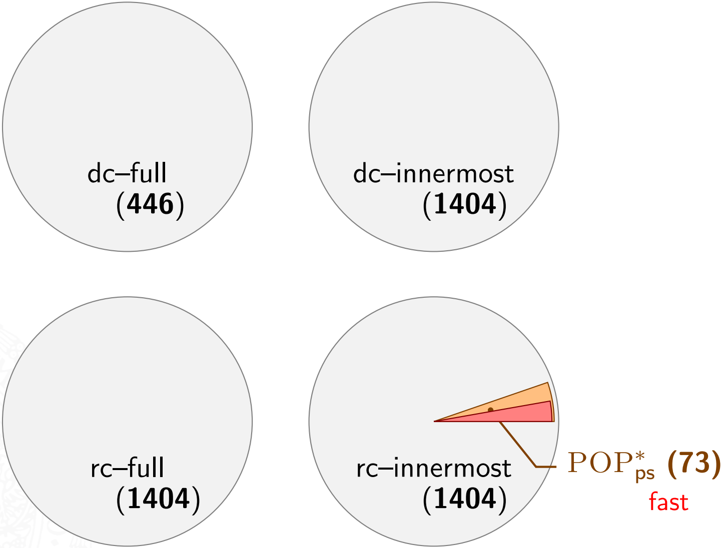
Automated Complexity Analysis: A Snapshot

polynomial derivation length on TPDB



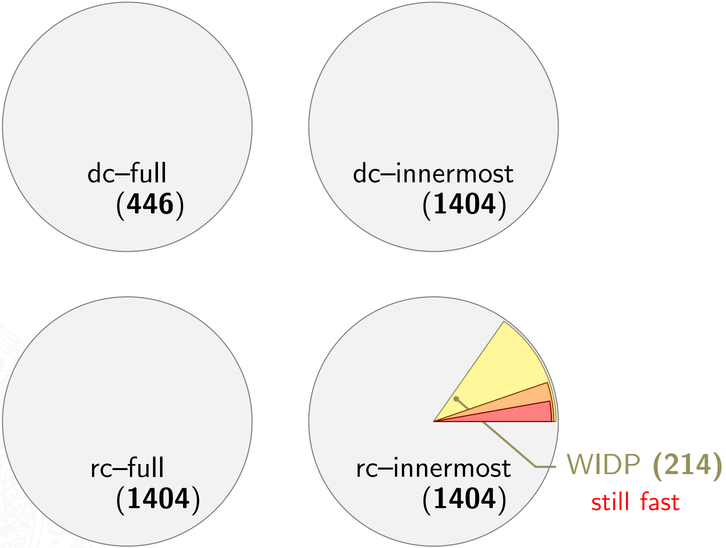
Automated Complexity Analysis: A Snapshot

polynomial derivation length on TPDB



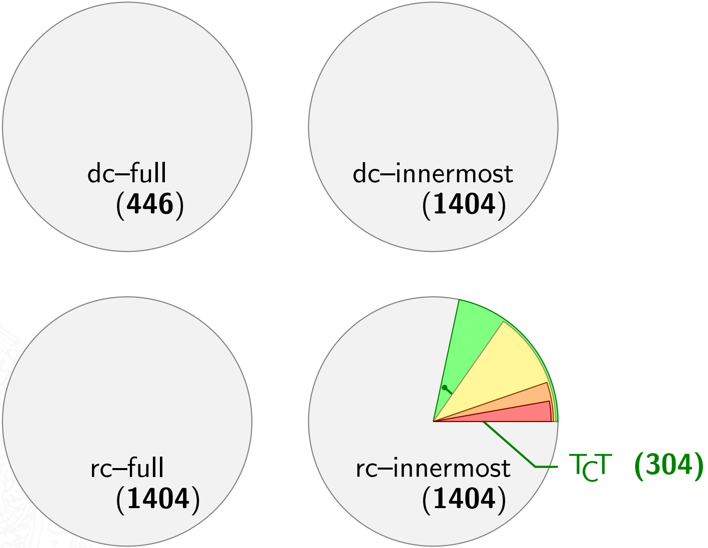
Automated Complexity Analysis: A Snapshot

polynomial derivation length on TPDB



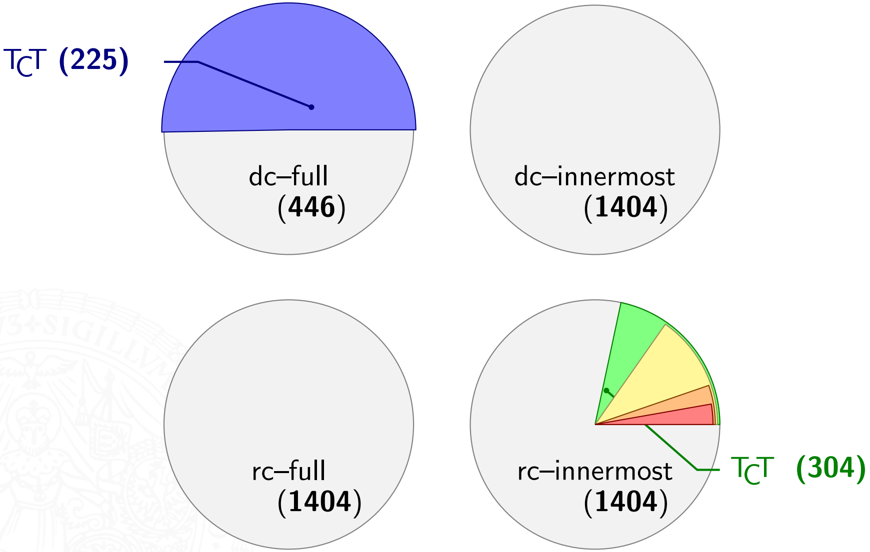
Automated Complexity Analysis: A Snapshot

polynomial derivation length on TPDB



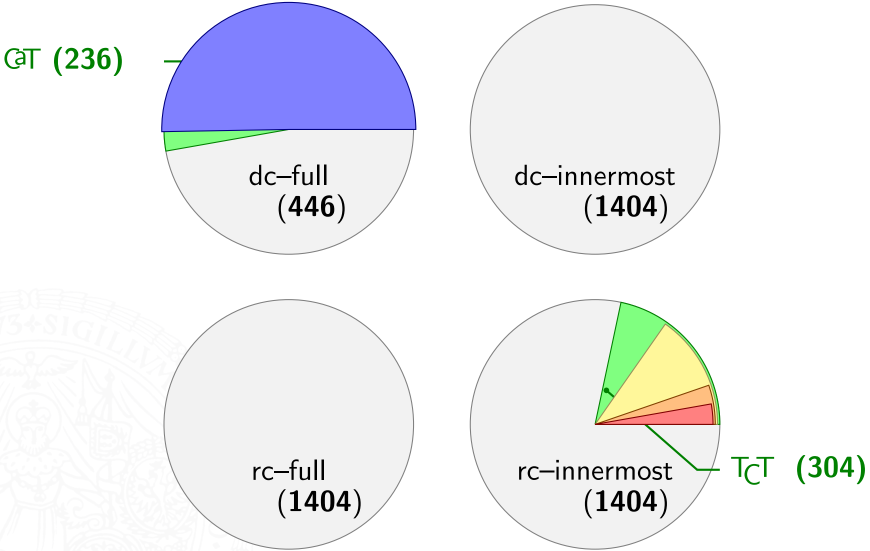
Automated Complexity Analysis: A Snapshot

polynomial derivation length on TPDB



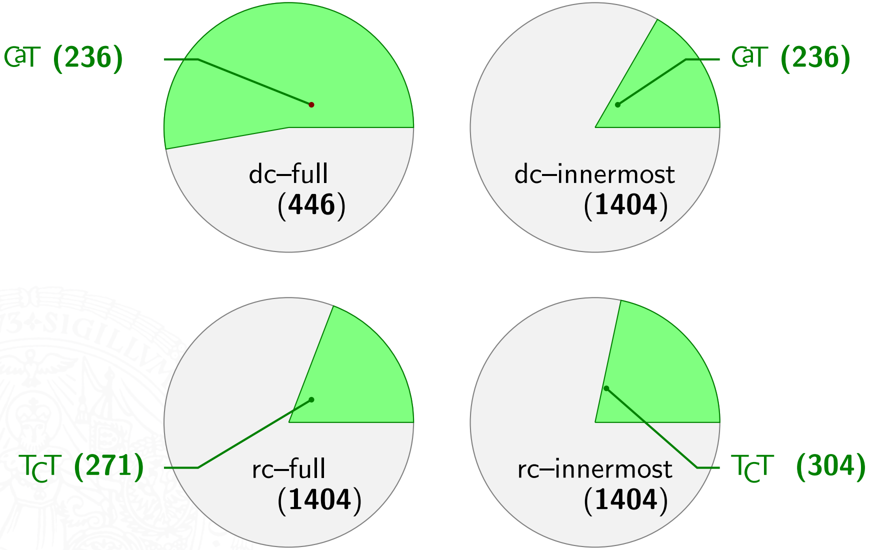
Automated Complexity Analysis: A Snapshot

polynomial derivation length on TPDB



Automated Complexity Analysis: A Snapshot

polynomial derivation length on TPDB



Complexity Analysis Based on DP Method

consider $\mathcal{R}_{\mathsf{div}}$

$$
\begin{aligned}
1: &\quad x - 0 \to x & \qquad 3: &\quad 0 \div \mathsf{s}(y) \to 0 \\
2: &\quad \mathsf{s}(x) - \mathsf{s}(y) \to x - y & \qquad 4: &\quad \mathsf{s}(x) \div \mathsf{s}(y) \to \mathsf{s}((x - y) \div \mathsf{s}(y))
\end{aligned}
$$

Complexity Analysis Based on DP Method

consider $\mathcal{R}_{\mathsf{div}}$

$$
\begin{array}{llll}
1: & x - 0 \to x & 3: & 0 \div \mathsf{s}(y) \to 0 \\
2: & \mathsf{s}(x) - \mathsf{s}(y) \to x - y & 4: & \mathsf{s}(x) \div \mathsf{s}(y) \to \mathsf{s}((x - y) \div \mathsf{s}(y))
\end{array}
$$

Question

what is the runtime complexity of $\mathcal{R}_{\mathsf{div}}$?

Complexity Analysis Based on DP Method

consider $\mathcal{R}_{\mathsf{div}}$

$$
\begin{aligned}
1: &\quad x - 0 \to x & 3: &\quad 0 \div \mathsf{s}(y) \to 0 \\
2: &\quad \mathsf{s}(x) - \mathsf{s}(y) \to x - y & 4: &\quad \mathsf{s}(x) \div \mathsf{s}(y) \to \mathsf{s}((x - y) \div \mathsf{s}(y))
\end{aligned}
$$

Question

what is the runtime complexity of $\mathcal{R}_{\mathsf{div}}$?

Answer

linear

Complexity Analysis Based on DP Method

consider $\mathcal{R}_{\mathsf{div}}$

$$1: \quad x - 0 \to x \qquad\qquad 3: \quad 0 \div \mathsf{s}(y) \to 0$$

$$2: \quad \mathsf{s}(x) - \mathsf{s}(y) \to x - y \qquad 4: \quad \mathsf{s}(x) \div \mathsf{s}(y) \to \mathsf{s}((x - y) \div \mathsf{s}(y))$$

Question

what is the derivational complexity of $\mathcal{R}_{\mathsf{div}}$?

Answer

Complexity Analysis Based on DP Method

consider $\mathcal{R}_{\mathsf{div}}$

$$
\begin{aligned}
1: &\quad x - 0 \to x & 3: &\quad 0 \div \mathsf{s}(y) \to 0 \\
2: &\quad \mathsf{s}(x) - \mathsf{s}(y) \to x - y & 4: &\quad \mathsf{s}(x) \div \mathsf{s}(y) \to \mathsf{s}((x - y) \div \mathsf{s}(y))
\end{aligned}
$$

Question

what is the derivational complexity of $\mathcal{R}_{\mathsf{div}}$?

Answer

at least exponential

Complexity Analysis Based on DP Method

consider $\mathcal{R}_{\mathsf{div}}$

$$
\begin{aligned}
1:&\quad x - 0 \to x & \qquad 3:&\quad 0 \div \mathsf{s}(y) \to 0 \\
2:&\quad \mathsf{s}(x) - \mathsf{s}(y) \to x - y & \qquad 4:&\quad \mathsf{s}(x) \div \mathsf{s}(y) \to \mathsf{s}((x - y) \div \mathsf{s}(y))
\end{aligned}
$$

Question

what is the innermost runtime complexity of $\mathcal{R}_{\mathsf{div}}$?

Answer

Complexity Analysis Based on DP Method

consider $\mathcal{R}_{\mathsf{div}}$

$$
\begin{aligned}
1:&\quad x - 0 \to x & \qquad 3:&\quad 0 \div \mathsf{s}(y) \to 0 \\
2:&\quad \mathsf{s}(x) - \mathsf{s}(y) \to x - y & \qquad 4:&\quad \mathsf{s}(x) \div \mathsf{s}(y) \to \mathsf{s}((x - y) \div \mathsf{s}(y))
\end{aligned}
$$

Question

what is the innermost runtime complexity of $\mathcal{R}_{\mathsf{div}}$?

Answer

linear

Complexity Analysis Based on DP Method

consider $\mathcal{R}_{\mathsf{div}}$

$$
\begin{aligned}
1:&\quad x - 0 \to x & 3:&\quad 0 \div \mathsf{s}(y) \to 0\\
2:&\quad \mathsf{s}(x) - \mathsf{s}(y) \to x - y & 4:&\quad \mathsf{s}(x) \div \mathsf{s}(y) \to \mathsf{s}((x - y) \div \mathsf{s}(y))
\end{aligned}
$$

Question

what is the innermost runtime complexity of $\mathcal{R}_{\mathsf{div}}$?

Answer

linear

Challenge

how to prove (at least) innermost polynomial runtime complexity automatically?

Weak Dependency Pairs

Definition weak dependency pairs

$$\mathsf{WDP}(\mathcal{R}) = \{\, l^\sharp \to \textsc{com}(u_1{}^\sharp, \dots, u_n{}^\sharp) \mid (l \to r) \in \mathcal{R},\ r = \underbrace{C[u_1, \dots, u_n]}_{\text{no defined symbols, no vars in } C}\}$$

$u_i \in \mathcal{V}$ or starts with defined functions symbols

Weak Dependency Pairs

Definition — weak dependency pairs

$$\text{WDP}(\mathcal{R}) = \{\, l^\sharp \to \textsc{com}({u_1}^\sharp, \dots, {u_n}^\sharp) \mid (l \to r) \in \mathcal{R},\ r = \underbrace{C[u_1, \dots, u_n]}_{\text{no defined symbols, no vars in } C} \}$$

$u_i \in \mathcal{V}$ or starts with defined functions symbols; $\textsc{com}(t_1, \dots, t_n)$ is t_1 if $n = 1$, and $c(t_1, \dots, t_n)$ otherwise

Weak Dependency Pairs

Definition — weak dependency pairs

$$\mathsf{WDP}(\mathcal{R}) = \{\, l^\sharp \to \textsc{com}({u_1}^\sharp, \dots, {u_n}^\sharp) \mid (l \to r) \in \mathcal{R},\ r = \underbrace{C[u_1, \dots, u_n]}_{\text{no defined symbols, no vars in } C} \}$$

$u_i \in \mathcal{V}$ or starts with defined functions symbols; $\textsc{com}(t_1, \dots, t_n)$ is t_1 if $n = 1$, and $c(t_1, \dots, t_n)$ otherwise

consider $\mathsf{WDP}(\mathcal{R}_{\mathsf{div}})$:

$$5: \quad x -^\sharp 0 \to x \qquad 7: \quad 0 \div^\sharp \mathsf{s}(y) \to \mathsf{c}$$

$$6: \quad \mathsf{s}(x) -^\sharp \mathsf{s}(y) \to x -^\sharp y \qquad 8: \quad \mathsf{s}(x) \div^\sharp \mathsf{s}(y) \to (x - y) \div^\sharp \mathsf{s}(y)$$

Weak Dependency Pairs

Definition weak dependency pairs

$$\mathsf{WDP}(\mathcal{R}) = \{\, l^\sharp \to \textsc{com}({u_1}^\sharp, \dots, {u_n}^\sharp) \mid (l \to r) \in \mathcal{R},\ r = \underbrace{C[u_1, \dots, u_n]}_{\text{no defined symbols, no vars in } C} \}$$

$u_i \in \mathcal{V}$ or starts with defined functions symbols; $\textsc{com}(t_1, \dots, t_n)$ is t_1 if $n = 1$, and $c(t_1, \dots, t_n)$ otherwise

consider $\mathsf{WDP}(\mathcal{R}_{\mathsf{div}})$:

$$\begin{array}{llll} 5: & x -^\sharp 0 \to x & 7: & 0 \div^\sharp \mathsf{s}(y) \to \mathsf{c} \\ 6: & \mathsf{s}(x) -^\sharp \mathsf{s}(y) \to x -^\sharp y & 8: & \mathsf{s}(x) \div^\sharp \mathsf{s}(y) \to (x - y) \div^\sharp \mathsf{s}(y) \end{array}$$

consider the TRS $\mathcal{R}$: $\mathsf{f}(\mathsf{s}(x)) \to \mathsf{g}(\mathsf{f}(x), \mathsf{f}(x))$

1. set $t_n = \mathsf{f}(\mathsf{s}^n(0))$, i.e, $\mathsf{dl}(t_{n+1}, \to_{\mathcal{R}}) \geqslant 2^n$
2. but $\mathsf{dl}(t^\sharp_{n+1}, \to_{\mathcal{R} \cup \mathsf{DP}(\mathcal{R})}) = n$

TRS $\mathcal{R}_{\mathsf{div}}$:

$$\begin{array}{rrclrrcl}
1: & x - 0 & \to & x & 3: & 0 \div \mathsf{s}(y) & \to & 0 \\
2: & \mathsf{s}(x) - \mathsf{s}(y) & \to & x - y & 4: & \mathsf{s}(x) \div \mathsf{s}(y) & \to & \mathsf{s}((x - y) \div \mathsf{s}(y))
\end{array}$$

$\mathcal{P} := \mathsf{WDP}(\mathcal{R}_{\mathsf{div}})$:

$$\begin{array}{rrclrrcl}
5: & x -^\sharp 0 & \to & x & 7: & 0 \div^\sharp \mathsf{s}(y) & \to & \mathsf{c} \\
6: & \mathsf{s}(x) -^\sharp \mathsf{s}(y) & \to & x -^\sharp y & 8: & \mathsf{s}(x) \div^\sharp \mathsf{s}(y) & \to & (x - y) \div^\sharp \mathsf{s}(y)
\end{array}$$

TRS $\mathcal{R}_{\mathsf{div}}$:

$$1: \quad x - 0 \to x \qquad 3: \quad 0 \div \mathsf{s}(y) \to 0$$

$$2: \quad \mathsf{s}(x) - \mathsf{s}(y) \to x - y \qquad 4: \quad \mathsf{s}(x) \div \mathsf{s}(y) \to \mathsf{s}((x - y) \div \mathsf{s}(y))$$

$\mathcal{P} := \mathsf{WDP}(\mathcal{R}_{\mathsf{div}})$:

$$5: \quad x -^\sharp 0 \to x \qquad 7: \quad 0 \div^\sharp \mathsf{s}(y) \to \mathsf{c}$$

$$6: \quad \mathsf{s}(x) -^\sharp \mathsf{s}(y) \to x -^\sharp y \qquad 8: \quad \mathsf{s}(x) \div^\sharp \mathsf{s}(y) \to (x - y) \div^\sharp \mathsf{s}(y)$$

$$\mathsf{s}(0) \div \mathsf{s}(0) \to_{\mathcal{R}_{\mathsf{div}}} \mathsf{s}((0 - 0) \div \mathsf{s}(0)) \to_{\mathcal{R}_{\mathsf{div}}} \mathsf{s}(0 \div \mathsf{s}(0)) \to_{\mathcal{R}_{\mathsf{div}}} \mathsf{s}(0)$$

TRS $\mathcal{R}_{\mathsf{div}}$:

$$1: \quad x - 0 \to x$$
$$2: \quad \mathsf{s}(x) - \mathsf{s}(y) \to x - y$$

$\mathcal{P} := \mathsf{WDP}(\mathcal{R}_{\mathsf{div}})$:

$$5: \quad x -^{\sharp} 0 \to x \qquad 7: \quad 0 \div^{\sharp} \mathsf{s}(y) \to \mathsf{c}$$
$$6: \quad \mathsf{s}(x) -^{\sharp} \mathsf{s}(y) \to x -^{\sharp} y \qquad 8: \quad \mathsf{s}(x) \div^{\sharp} \mathsf{s}(y) \to (x - y) \div^{\sharp} \mathsf{s}(y)$$

$$\mathsf{s}(0) \div^{\sharp} \mathsf{s}(0) \to_{\mathcal{U}(\mathcal{P}) \cup \mathcal{P}} (0 - 0) \div^{\sharp} \mathsf{s}(0) \to_{\mathcal{U}(\mathcal{P}) \cup \mathcal{P}} 0 \div^{\sharp} \mathsf{s}(0) \to_{\mathcal{U}(\mathcal{P}) \cup \mathcal{P}} \mathsf{c}$$

TRS $\mathcal{R}_{\mathsf{div}}$:

$$1: \quad x - 0 \to x$$
$$2: \quad \mathsf{s}(x) - \mathsf{s}(y) \to x - y$$

$\mathcal{P} := \mathsf{WDP}(\mathcal{R}_{\mathsf{div}})$:

$$5: \quad x -^\sharp 0 \to x \qquad 7: \quad 0 \div^\sharp \mathsf{s}(y) \to \mathsf{c}$$
$$6: \quad \mathsf{s}(x) -^\sharp \mathsf{s}(y) \to x -^\sharp y \qquad 8: \quad \mathsf{s}(x) \div^\sharp \mathsf{s}(y) \to (x - y) \div^\sharp \mathsf{s}(y)$$

Lemma

$\mathsf{dl}(t, \to_{\mathcal{R}}) = \mathsf{dl}(t^\sharp, \to_{\mathcal{U}(\mathsf{WDP}(\mathcal{R})) \cup \mathsf{WDP}(\mathcal{R})})$

$$\mathsf{s}(0) \div^\sharp \mathsf{s}(0) \to_{\mathcal{U}(\mathcal{P}) \cup \mathcal{P}} (0 - 0) \div^\sharp \mathsf{s}(0) \to_{\mathcal{U}(\mathcal{P}) \cup \mathcal{P}} 0 \div^\sharp \mathsf{s}(0) \to_{\mathcal{U}(\mathcal{P}) \cup \mathcal{P}} \mathsf{c}$$

TRS $\mathcal{R}_{\mathsf{div}}$:

$$1: \quad x - 0 \to x$$
$$2: \quad \mathsf{s}(x) - \mathsf{s}(y) \to x - y$$

$\mathcal{P} := \mathsf{WDP}(\mathcal{R}_{\mathsf{div}})$:

$$5: \quad x -^\sharp 0 \to x \qquad 7: \quad 0 \div^\sharp \mathsf{s}(y) \to \mathsf{c}$$
$$6: \quad \mathsf{s}(x) -^\sharp \mathsf{s}(y) \to x -^\sharp y \qquad 8: \quad \mathsf{s}(x) \div^\sharp \mathsf{s}(y) \to (x - y) \div^\sharp \mathsf{s}(y)$$

Lemma

$\mathsf{dl}(t, \to_{\mathcal{R}}) = \mathsf{dl}(t^\sharp, \to_{\mathcal{U}(\mathsf{WDP}(\mathcal{R})) \cup \mathsf{WDP}(\mathcal{R})})$

$$\mathsf{s}(0) \div^\sharp \mathsf{s}(0) \to_{\mathcal{U}(\mathcal{P}) \cup \mathcal{P}} (0 - 0) \div^\sharp \mathsf{s}(0) \to_{\mathcal{U}(\mathcal{P}) \cup \mathcal{P}} 0 \div^\sharp \mathsf{s}(0) \to_{\mathcal{U}(\mathcal{P}) \cup \mathcal{P}} \mathsf{c}$$

TRS $\mathcal{R}_{\mathsf{div}}$:

$$1: \quad x - 0 \to x$$
$$2: \quad \mathsf{s}(x) - \mathsf{s}(y) \to x - y$$

$\mathcal{P} := \mathsf{WDP}(\mathcal{R}_{\mathsf{div}})$:

$$5: \quad x -^\sharp 0 \to x \qquad 7: \quad 0 \div^\sharp \mathsf{s}(y) \to \mathsf{c}$$
$$6: \quad \mathsf{s}(x) -^\sharp \mathsf{s}(y) \to x -^\sharp y \qquad 8: \quad \mathsf{s}(x) \div^\sharp \mathsf{s}(y) \to (x - y) \div^\sharp \mathsf{s}(y)$$

Lemma

$\mathsf{dl}(t, \to_{\mathcal{R}}) = \mathsf{dl}(t^\sharp, \to_{\mathcal{U}(\mathsf{WDP}(\mathcal{R})) \cup \mathsf{WDP}(\mathcal{R})})$

$$\mathsf{s}(0) \div^\sharp \mathsf{s}(0) \to_{\mathcal{U}(\mathcal{P}) \cup \mathcal{P}} (0 - 0) \div^\sharp \mathsf{s}(0) \to_{\mathcal{U}(\mathcal{P}) \cup \mathcal{P}} 0 \div^\sharp \mathsf{s}(0) \to_{\mathcal{U}(\mathcal{P}) \cup \mathcal{P}} \mathsf{c}$$

Definition Fernández 2005

$\mathcal{UA}(f, \mathcal{R})$ collects the argument positions of f that are used

Theorem Hirokawa-M 2008

$\forall$ TRS $\mathcal{R}$, assume

- $\exists$ rewrite order $\succ$ that induces linear runtime complexity
- $\mathcal{U}(\mathsf{W(I)DP}(\mathcal{R})) \cup \mathsf{W(I)DP}(\mathcal{R}) \subseteq \succ$

then the (innermost) runtime complexity of $\mathcal{R}$ is linear

Theorem

Hirokawa-M 2009

$\forall$ TRS $\mathcal{R}$, assume

- $\exists$ rewrite order $\succ$ that induces linear runtime complexity
- $\mathcal{U}(\mathsf{WIDP}(\mathcal{R})) \cup \mathsf{WIDP}(\mathcal{R}) \subseteq \succ$ and

then the innermost runtime complexity of $\mathcal{R}$ is linear

Theorem Hirokawa-M 2009

$\forall$ TRS $\mathcal{R}$, assume

- $\exists$ stable order $\succ$ that induces linear runtime complexity
- $\mathcal{U}(\mathsf{WIDP}(\mathcal{R})) \cup \mathsf{WIDP}(\mathcal{R}) \subseteq \succ$ and
- $\succ$ is monotone on $\mathcal{UA}(f, \mathcal{R})$ for any $f \in \mathcal{F}^\sharp$

then the innermost runtime complexity of $\mathcal{R}$ is linear

Theorem Hirokawa-M 2009

$\forall$ TRS $\mathcal{R}$, assume

- $\exists$ stable order $\succ$ that induces linear runtime complexity
- $\mathcal{U}(\mathsf{WIDP}(\mathcal{R})) \cup \mathsf{WIDP}(\mathcal{R}) \subseteq \succ$ and
- $\succ$ is monotone on $\mathcal{UA}(f, \mathcal{R})$ for any $f \in \mathcal{F}^\sharp$
- $\mathsf{WIDP}(\mathcal{R})$ is constructor

then the innermost runtime complexity of $\mathcal{R}$ is linear

Theorem

Hirokawa-M 2009

$\forall$ TRS $\mathcal{R}$, assume

- $\exists$ stable order $\succ$ that induces linear runtime complexity
- $\mathcal{U}(\text{WIDP}(\mathcal{R})) \cup \text{WIDP}(\mathcal{R}) \subseteq \succ$ and
- $\succ$ is monotone on $\mathcal{UA}(f, \mathcal{R})$ for any $f \in \mathcal{F}^\sharp$
- WIDP($\mathcal{R}$) is constructor

then the innermost runtime complexity of $\mathcal{R}$ is linear

consider

$\mathcal{U}(\mathcal{P}) \cup \mathcal{P}$ and the WMA $\mathcal{A}$:

$$0_\mathcal{A} = \mathsf{c}_\mathcal{A} = 0 \quad \mathsf{s}_\mathcal{A}(x) = x + 2 \quad -_\mathcal{A}(x, y) = -^\sharp_\mathcal{A}(x, y) = \div^\sharp_\mathcal{A}(x, y) = x + 1$$

1. $\mathcal{A}$ is monotone on usable arguments
2. $\mathcal{P} \subseteq >_\mathcal{A}$ and $\mathcal{U}(\mathcal{P}) \subseteq >_\mathcal{A}$

we conclude linear innermost runtime complexity

The Future

Open Problems and Challenges

Open Problems and Challenges

Goal ① modern

... complexity-wise it is the removal of rules that adds real power to the DP method

Open Problems and Challenges

Goal ① modern

... complexity-wise it is the removal of rules that adds real power to the DP method

Goal ② useful

... how to remove all the extra conditions from the WDP method

Open Problems and Challenges

Goal ① modern

... complexity-wise it is the removal of rules that adds real power to the DP method

Goal ② useful

... how to remove all the extra conditions from the WDP method

Goal ④ applications

program analysis, completion, automated deduction, implicit computational complexity theory, proof theory ...

Open Problems and Challenges

Goal ① modern

… complexity-wise it is the removal of rules that adds real power to the DP method

Goal ② useful

… how to remove all the extra conditions from the WDP method

Goal ④ applications

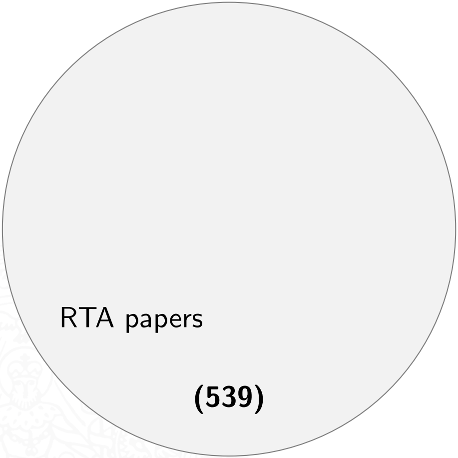
program analysis, completion, automated deduction, implicit computational complexity theory, proof theory …

… complexity preserving transformations from Scheme, Haskell, Logic Programs, JAVA bytecode …

Thank you for Your Attention!

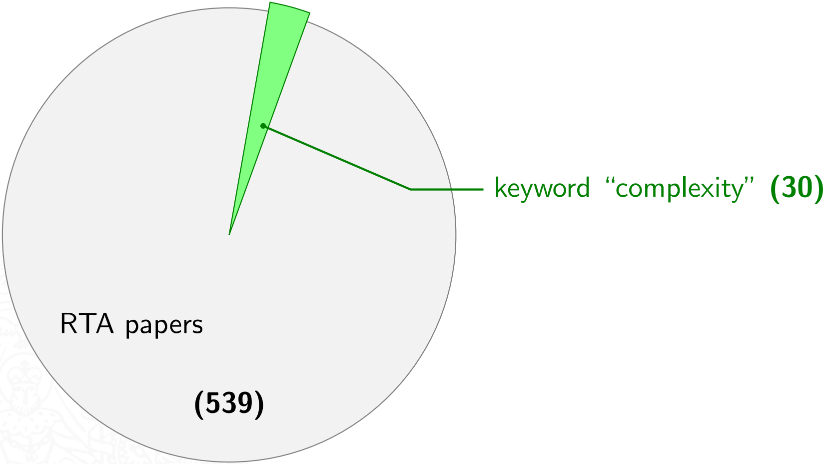
Complexity in Rewriting: A History

number of complexity related papers



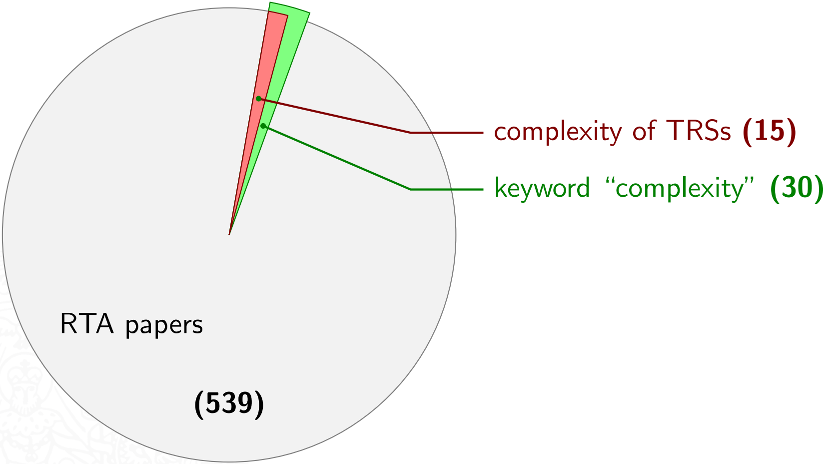
Complexity in Rewriting: A History

number of complexity related papers



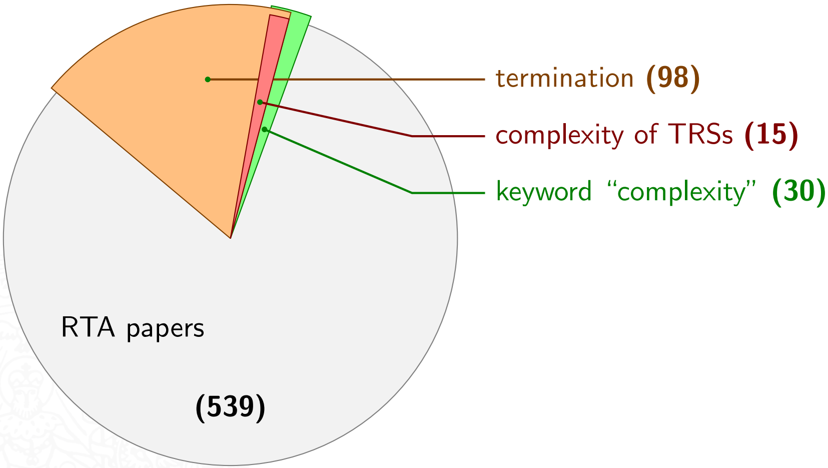
Complexity in Rewriting: A History

number of complexity related papers



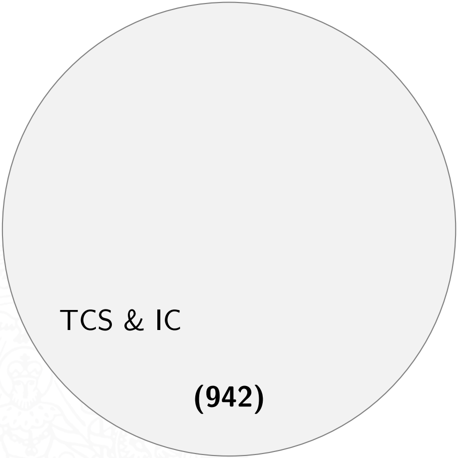
Complexity in Rewriting: A History

number of complexity related papers



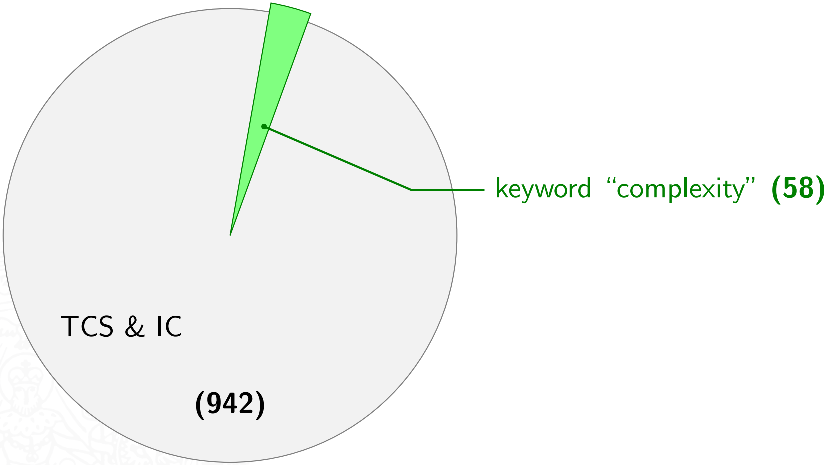
Complexity in Rewriting: A History

number of complexity related papers



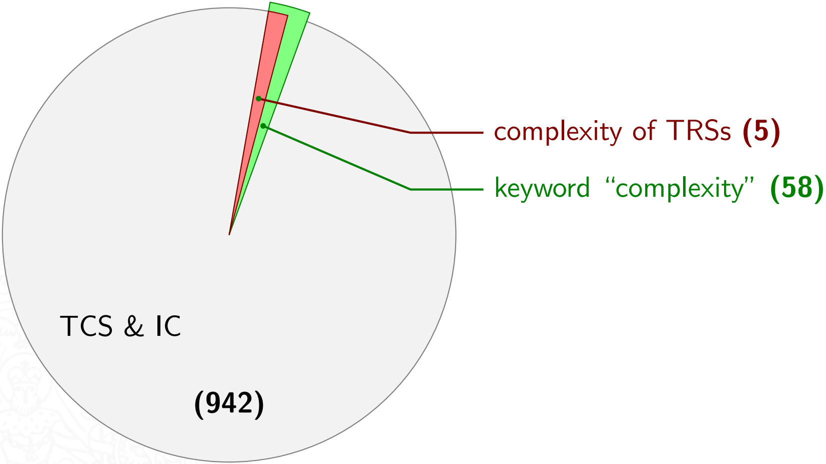
Complexity in Rewriting: A History

number of complexity related papers



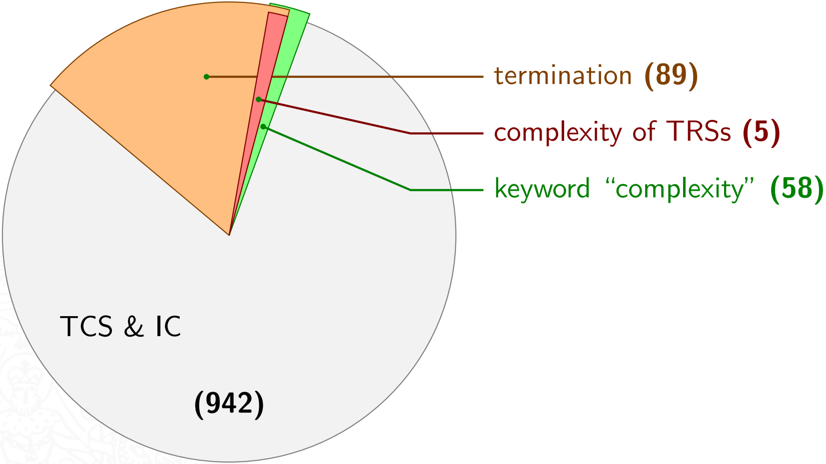
Complexity in Rewriting: A History

number of complexity related papers



Complexity in Rewriting: A History

number of complexity related papers



Usable Argument Positions

Definition

let $\mathcal{R}$, $\mathcal{P}$ be a pair of TRSs based on signature $\mathcal{F}$, such that $\mathcal{P}$ is a constructur TRS with respect to $\mathcal{F}$. The set of usable arguments of $f \in \mathcal{F}$ with respect to $\mathcal{R}$, $\mathcal{P}$ is defined as follows.

$$\mathcal{UA}(f, \mathcal{R}, \mathcal{P}) := \{1 \leqslant i \leqslant n \mid \exists l \to r \in \mathcal{R} \cup \mathcal{P}, \exists p, p' \in \mathcal{P}\text{os}(r) \text{ such that } p'.i \leqslant p, \operatorname{root}(r|_p) \text{ is defined in } \mathcal{R}, \operatorname{root}(s|_{p'}) = f, \text{ and } l \not\rhd r|_p\}$$

Here the arity of f is n.

Usable Argument Positions

Definition

let $\mathcal{R}$, $\mathcal{P}$ be a pair of TRSs based on signature $\mathcal{F}$, such that $\mathcal{P}$ is a constructur TRS with respect to $\mathcal{F}$. The set of usable arguments of $f \in \mathcal{F}$ with respect to $\mathcal{R}$, $\mathcal{P}$ is defined as follows.

$$\mathcal{UA}(f, \mathcal{R}, \mathcal{P}) := \{1 \leqslant i \leqslant n \mid \exists l \to r \in \mathcal{R} \cup \mathcal{P}, \exists p, p' \in \mathcal{P}\mathrm{os}(r) \text{ such that } p'.i \leqslant p, \mathrm{root}(r|_p) \text{ is defined in } \mathcal{R}, \mathrm{root}(s|_{p'}) = f, \text{ and } l \not\rhd r|_p\}$$

Here the arity of f is n.

$$1\colon \quad x - 0 \to x \qquad 3\colon \quad 0 \div \mathsf{s}(y) \to 0$$

$$2\colon \quad \mathsf{s}(x) - \mathsf{s}(y) \to x - y \qquad 4\colon \quad \mathsf{s}(x) \div \mathsf{s}(y) \to \mathsf{s}((x - y) \div \mathsf{s}(y))$$

usable argument positions are as follows

$$\mathcal{UA}_{\mathcal{R}}(0) = \mathcal{UA}_{\mathcal{R}}(-) = \varnothing \qquad \mathcal{UA}_{\mathcal{R}}(\mathsf{s}) = \mathcal{UA}_{\mathcal{R}}(\div) = \{1\}$$