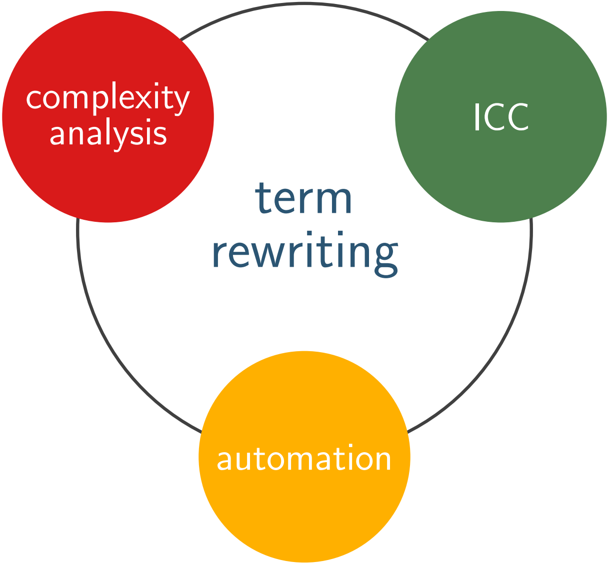
Verifying Polytime Computability Automatically

Martin Avanzini

Institute of Computer Science
University of Innsbruck, Austria

November 12, 2013

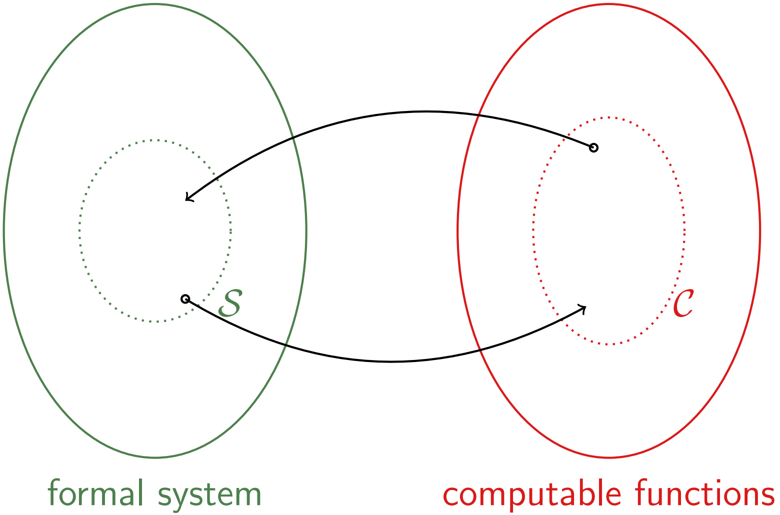

complexity analysis
ICC
term rewriting
automation

Outline

1. preliminaries
2. path orders
   - for runtime complexity analysis
   - applications in ICC
3. automation
4. invariance theorem

Implicit Computational Complexity

implicit characterisations of complexity classes



Term Rewriting

in a nutshell

Example

term rewrite system (TRS) $\mathcal{R}_{\mathsf{rev}}$ consists of rules

$$1: \quad [\,] \mathbin{@} ys \to ys \qquad\qquad 2: \quad \mathsf{rev}([\,]) \to [\,]$$

$$3: \quad (x : xs) \mathbin{@} ys \to x : (xs \mathbin{@} ys) \qquad\qquad 4: \quad \mathsf{rev}(x : xs) \to \mathsf{rev}(xs) \mathbin{@} (x : [\,])$$

Term Rewriting

in a nutshell

Example

term rewrite system (TRS) $\mathcal{R}_{\mathsf{rev}}$ consists of rules

$$1\colon\quad [\,] \mathbin{@} ys \to ys \qquad\qquad 2\colon\quad \mathsf{rev}([\,]) \to [\,]$$

$$3\colon\quad (x : xs) \mathbin{@} ys \to x : (xs \mathbin{@} ys) \qquad\qquad 4\colon\quad \mathsf{rev}(x : xs) \to \mathsf{rev}(xs) \mathbin{@} (x : [\,])$$

rewriting

$$\mathsf{rev}(1 : (2 : (3 : [\,]))) \to_{\mathcal{R}_{\mathsf{rev}}} \mathsf{rev}(2 : (3 : [\,])) \mathbin{@} (1 : [\,])$$

Term Rewriting

in a nutshell

Example

term rewrite system (TRS) $\mathcal{R}_{\mathsf{rev}}$ consists of rules

$$
\begin{array}{rrclrrcl}
1: & [\,] \mathbin{@} ys & \to & ys & 2: & \mathsf{rev}([\,]) & \to & [\,] \\
3: & (x : xs) \mathbin{@} ys & \to & x : (xs \mathbin{@} ys) & 4: & \mathsf{rev}(x : xs) & \to & \mathsf{rev}(xs) \mathbin{@} (x : [\,])
\end{array}
$$

rewriting

$$
\begin{aligned}
\mathsf{rev}(1 : (2 : (3 : [\,]))) &\to_{\mathcal{R}_{\mathsf{rev}}} \mathsf{rev}(2 : (3 : [\,])) \mathbin{@} (1 : [\,]) \\
&\to_{\mathcal{R}_{\mathsf{rev}}} \cdots \\
&\to_{\mathcal{R}_{\mathsf{rev}}} 3 : (2 : (1 : [\,]))
\end{aligned}
$$

Term Rewriting

in a nutshell

Example

term rewrite system (TRS) $\mathcal{R}_{\mathrm{rev}}$ consists of rules

$$1: \quad [\,] \mathbin{@} ys \to ys \qquad\qquad 2: \quad \mathrm{rev}([\,]) \to [\,]$$

$$3: \quad (x : xs) \mathbin{@} ys \to x : (xs \mathbin{@} ys) \qquad\qquad 4: \quad \mathrm{rev}(x : xs) \to \mathrm{rev}(xs) \mathbin{@} (x : [\,])$$

rewriting $\equiv$ computation

$$\begin{aligned}
\mathrm{rev}(1 : (2 : (3 : [\,]))) &\to_{\mathcal{R}_{\mathrm{rev}}} \mathrm{rev}(2 : (3 : [\,])) \mathbin{@} (1 : [\,]) \\
&\to_{\mathcal{R}_{\mathrm{rev}}} \cdots \\
&\to_{\mathcal{R}_{\mathrm{rev}}} 3 : (2 : (1 : [\,]))
\end{aligned}$$

Term Rewriting

in a nutshell

Example

term rewrite system (TRS) $\mathcal{R}_{\mathsf{rev}}$ consists of rules

constructor TRS

$$
\begin{array}{llll}
1: & [\,] \mathbin{@} ys \to ys & 2: & \mathsf{rev}([\,]) \to [\,] \\
3: & (x : xs) \mathbin{@} ys \to x : (xs \mathbin{@} ys) & 4: & \mathsf{rev}(x : xs) \to \mathsf{rev}(xs) \mathbin{@} (x : [\,])
\end{array}
$$

rewriting $\equiv$ computation

$$
\begin{aligned}
\mathsf{rev}(1 : (2 : (3 : [\,]))) &\to_{\mathcal{R}_{\mathsf{rev}}} \mathsf{rev}(2 : (3 : [\,])) \mathbin{@} (1 : [\,]) \\
&\to_{\mathcal{R}_{\mathsf{rev}}} \cdots \\
&\to_{\mathcal{R}_{\mathsf{rev}}} 3 : (2 : (1 : [\,]))
\end{aligned}
$$

constructor terms $\equiv$ values

Term Rewriting

in a nutshell

Example

term rewrite system (TRS) $\mathcal{R}_{\mathsf{rev}}$ consists of rules

constructor TRS

$$\begin{array}{llll}
1: & [\,] \mathbin{@} ys \to ys & 2: & \mathsf{rev}([\,]) \to [\,] \\
3: & (x : xs) \mathbin{@} ys \to x : (xs \mathbin{@} ys) & 4: & \mathsf{rev}(x : xs) \to \mathsf{rev}(xs) \mathbin{@} (x : [\,])
\end{array}$$

rewriting $\equiv$ computation

$$\begin{aligned}
\mathsf{rev}(1 : (2 : (3 : [\,]))) &\to_{\mathcal{R}_{\mathsf{rev}}} \mathsf{rev}(2 : (3 : [\,])) \mathbin{@} (1 : [\,]) \\
&\to_{\mathcal{R}_{\mathsf{rev}}} \cdots \\
&\to_{\mathcal{R}_{\mathsf{rev}}} 3 : (2 : (1 : [\,]))
\end{aligned}$$

constructor terms $\equiv$ values

$$\begin{tikzcd}
 & s \arrow[dl, "*"'] \arrow[dr, "*"] \arrow[d, phantom, "\forall"] & \\
t_1 \arrow[dr, dashed, "*"'] & \exists & t_2 \arrow[dl, dashed, "*"] \\
 & u &
\end{tikzcd}$$

confluence

Term Rewriting

in a nutshell

Example

term rewrite system (TRS) $\mathcal{R}_{\mathsf{rev}}$ consists of rules

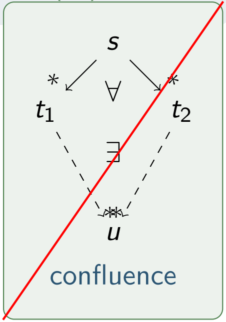
constructor TRS

$1{:}\quad [\,] \mathbin{@} ys \to ys$	$2{:}\quad \mathsf{rev}([\,]) \to [\,]$
$3{:}\quad (x : xs) \mathbin{@} ys \to x : (xs \mathbin{@} ys)$	$4{:}\quad \mathsf{rev}(x : xs) \to \mathsf{rev}(xs) \mathbin{@} (x : [\,])$
$5{:}\quad \mathsf{elem}(x : xs) \to x$	$6{:}\quad \mathsf{elem}(x : xs) \to \mathsf{elem}(xs)$

rewriting $\equiv$ computation

$$\begin{aligned}
\mathsf{rev}(1 : (2 : (3 : [\,]))) &\to_{\mathcal{R}_{\mathsf{rev}}} \mathsf{rev}(2 : (3 : [\,])) \mathbin{@} (1 : [\,]) \\
&\to_{\mathcal{R}_{\mathsf{rev}}} \cdots \\
&\to_{\mathcal{R}_{\mathsf{rev}}} 3 : (2 : (1 : [\,]))
\end{aligned}$$

constructor terms $\equiv$ values



confluence

Complexity of Term Rewrite Systems

- derivation height of term t with respect to relation $\to$

$$\mathsf{dh}(t, \to) = \max\{\ell \mid \exists (t_1, \ldots, t_\ell).\ t \to t_1 \to \cdots \to t_\ell\}$$

❶ runtime complexity of TRS $\mathcal{R}$

$$\mathsf{rc}_{\mathcal{R}}(n) = \max\{\, \mathsf{dh}(f(\vec{s}), \to_{\mathcal{R}}) \mid f(\vec{s}) \text{ and } \vec{s} \text{ are values of size up to } n \,\}$$

Complexity of Term Rewrite Systems

- derivation height of term t with respect to relation $\to$

$$\mathsf{dh}(t, \to) = \max\{\ell \mid \exists(t_1, \dots, t_\ell).\ t \to t_1 \to \dots \to t_\ell\}$$

❶ runtime complexity of TRS $\mathcal{R}$

$$\mathsf{rc}_{\mathcal{R}}(n) = \max\{\, \mathsf{dh}(f(\vec{s}), \to_{\mathcal{R}}) \mid f(\vec{s}) \text{ and } \vec{s} \text{ are values of size up to } n \,\}$$

❷ innermost runtime complexity of TRS $\mathcal{R}$

$$\mathsf{rc}^{\mathsf{i}}_{\mathcal{R}}(n) = \max\{\, \mathsf{dh}(f(\vec{s}), \xrightarrow{\mathsf{i}}_{\mathcal{R}}) \mid f(\vec{s}) \text{ and } \vec{s} \text{ are values of size up to } n \,\}$$

innermost rewriting $\approx$ eager reduction

Part I

Runtime Complexity Analysis

- Small Polynomial Path Order
- Exponential Polynomial Path Order

Presented at...

M. Avanzini and N. Eguchi and G. Moser
A Path Order for Rewrite Systems that Compute Exponential Time Functions.
Proc. of 22nd RTA, LIPIcs, pages 123–138, 2011

M. Avanzini and N. Eguchi and G. Moser
A New Order-theoretic Characterisation of the Polytime Computable Functions.
Proc. of 10th APLAS, LNCS, pages 280–295, 2012

Small Polynomial Path Order

basics

- TRS $\mathcal{R}$ compatible with $>_{\mathsf{spop}*}$ if $\mathcal{R} \subseteq >_{\mathsf{spop}*}$

$$l >_{\mathsf{spop}*} r \quad \text{for all rules } l \to r \in \mathcal{R}$$

Small Polynomial Path Order

basics

- TRS $\mathcal{R}$ compatible with $>_{\mathsf{spop}*}$ if $\mathcal{R} \subseteq >_{\mathsf{spop}*}$

$$l >_{\mathsf{spop}*} r \quad \text{for all rules } l \to r \in \mathcal{R}$$

1. ensures termination of $\mathcal{R}$

Small Polynomial Path Order

basics

- TRS $\mathcal{R}$ compatible with $>_{\mathsf{spop}*}$ if $\quad \mathcal{R} \subseteq >_{\mathsf{spop}*}$

$$l >_{\mathsf{spop}*} r \quad \text{for all rules } l \to r \in \mathcal{R}$$

1. ensures termination of $\mathcal{R}$
2. embodies predicative recursion on $\mathcal{R}$

Stephen Bellantoni and Stephen A. Cook
A New Recursion-Theoretic Characterization of the Polytime Functions.
Computational Complexity, Vol. 2, pages 97–110, 1992

Small Polynomial Path Order

basics

- TRS $\mathcal{R}$ compatible with $>_{\mathsf{spop}*}$ if $\qquad \mathcal{R} \subseteq >_{\mathsf{spop}*}$

$$l >_{\mathsf{spop}*} r \quad \text{for all rules } l \to r \in \mathcal{R}$$

1. ensures termination of $\mathcal{R}$
2. embodies predicative recursion on $\mathcal{R}$

ingredients

1. precedence $>$ on function symbols *constructors are minimal*

$$f > g \qquad \approx \qquad \text{“}f \text{ uses } g\text{”}$$

2. separation of arguments

$$f(\underbrace{n_1, \ldots, n_l}_{\textit{normal}}; \underbrace{n_{l+1}, \ldots, n_{l+k}}_{\textit{safe}})$$

Small Polynomial Path Order

with parameter substitution

Definition

$s = f(s_1, \ldots, s_k; s_{k+1}, \ldots, s_{k+l}) >_{\mathsf{spop}^*} t$ if

❶ $s_i \geqslant_{\mathsf{spop}^*} t$ for some argument s_i

❷ $t = g(t_1, \ldots, t_m; t_{m+1}, \ldots, t_{m+n})$ where $f > g$

- $f(s_1, \ldots, s_k; s_{k+1}, \ldots, s_{k+l}) \rhd_{\mathsf{n}} t_j$ for all normal arguments t_j
- $f(s_1, \ldots, s_k; s_{k+1}, \ldots, s_{k+l}) >_{\mathsf{spop}^*} t_j$ for all safe arguments t_j
- t contains symbol f at most once

❸ $t = f(t_1, \ldots, t_k; t_{k+1}, \ldots, t_{k+l})$ where $f \in \mathcal{D}_{\mathsf{rec}}$

- $\langle s_1, \ldots, s_k \rangle >^{\mathsf{prod}}_{\mathsf{spop}^*} \langle t_1, \ldots, t_k \rangle$
- $s >_{\mathsf{spop}^*} t_{k+1}, \ldots, s >_{\mathsf{spop}^*} t_{k+l}$, f does not occur in $t_{k+1}, \ldots, t_{k+l}$.

Small Polynomial Path Order

main result

Theorem

Suppose $\mathcal{R}$ is *constructor* TRS compatible with or $>_{\mathsf{spop}^*}$.
Then

- the *innermost runtime* *complexity* of $\mathcal{R}$ is polynomially bounded.

Small Polynomial Path Order

main result

Theorem

Suppose $\mathcal{R}$ is *constructor* TRS compatible with or $>_{\mathsf{spop}^*}$.
Then

- the *innermost runtime complexity* of $\mathcal{R}$ is polynomially bounded.
- degree of the polynomial is given by the maximal depth of recursion.

"counts nestings of recursive definitions"

Small Polynomial Path Orders

applications in ICC

Definition

constructor TRS $\mathcal{R}$ is predicative recursive (of degree d) if

1. $\mathcal{R}$ is compatible with $>_{\mathsf{spop}^*}$
2. recursion depth of $\mathcal{R}$ is d

Small Polynomial Path Orders

applications in ICC

Definition

constructor TRS $\mathcal{R}$ is predicative recursive (of degree d) if

1. $\mathcal{R}$ is compatible with $>_{\mathsf{spop}^*}$
2. recursion depth of $\mathcal{R}$ is d

Theorem *characterisation of* FP

The following classes of functions are equivalent:

1. The class of functions computed by predicative recursive TRSs.
2. The class FP of functions computable in polynomial time on *deterministic Turing machines*.

Exponential Path Order

Definition

$s = f(s_1, \ldots, s_k; s_{k+1}, \ldots, s_{k+l}) >_{\mathsf{epo}\star} t$ if

1. $s_i \geqslant_{\mathsf{spop}*} t$ for some argument s_i
2. $t = g(t_1, \ldots, t_m; t_{m+1}, \ldots, t_{m+n})$ where $f > g$
   - $f(s_1, \ldots, s_k; s_{k+1}, \ldots, s_{k+l}) \rhd_{\mathsf{n}} t_j$ for all normal arguments t_j
   - $f(s_1, \ldots, s_k; s_{k+1}, \ldots, s_{k+l}) >_{\mathsf{epo}\star} t_j$ for all safe arguments t_j
   - ~~t contains symbol f at most once~~
3. $t = f(t_1, \ldots, t_k; t_{k+1}, \ldots, t_{k+l})$ where
   - $\langle s_1, \ldots, s_k \rangle >^{\mathsf{lex}'}_{\mathsf{epo}\star} \langle t_1, \ldots, t_k \rangle$
   - $s >_{\mathsf{epo}\star} t_{k+1}, \ldots, s >_{\mathsf{epo}\star} t_{k+l}$, ~~$f$ does not occur in $t_{k+1}, \ldots, t_{k+l}$~~.

Exponential Path Order

main result

Theorem

Suppose $\mathcal{R}$ is *constructor* TRS compatible with $>_{\mathsf{epo}\star}$.

Then the *innermost runtime complexity* of $\mathcal{R}$ is in $2^{\mathcal{O}(n^k)}$ $(k \in \mathbb{N})$.

Exponential Polynomial Path Orders

order-theoretic characterisation of FEXP

Definition

constructor TRS $\mathcal{R}$ is predicative nested recursive if $\mathcal{R}$ is compatible with $>_{\mathsf{epo}\star}$

Theorem

The following classes of functions are equivalent:

1. The class of functions computed by predicative nested recursive TRSs.
2. The class FEXP of functions computable in exponential time on deterministic Turing machines.

Part II

Automation

- TCT
- Complexity Framework
- Complexity Processors

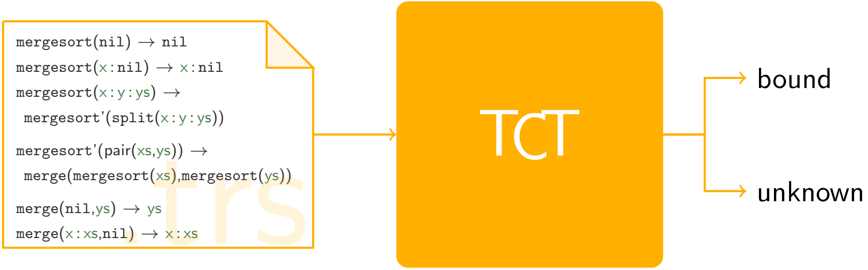
Presented at...

M. Avanzini and G. Moser and A. Schnabl
Automated Implicit Computational Complexity Analysis (System Description).
Proc. of 4$^{\text{th}}$ IJCAR, LNCS, pages 132–138, 2008

M. Avanzini and G. Moser
A Combination Framework for Complexity.
Proc. of 24$^{\text{th}}$ RTA, LIPIcs, pages 55–70, 2013

M. Avanzini and G. Moser
Tyrolean Complexity Tool: Features and Usage (System Description).
Proc. of 24$^{\text{th}}$ RTA, LIPIcs, pages 71–80, 2013

Tyrolean Complexity Tool



http://cl-informatik.uibk.ac.at/software/tct

open source™

Tyrolean Complexity Tool

history

2008 version 1.0 *extension to termination prover $\mathsf{T_TT_2}$*

- 3 dedicated complexity techniques

2009 version 1.5 *new implementation*

- in `Haskell`
- 9 methods implemented
- $\approx$ 3.400 lines of code

2013 version 2.0 *current version*

- 23 methods implemented
- $\approx$ 21.000 lines of code, of which 4.000 lines of comment

A Framework For Complexity Analysis of TRSs

complexity problem

- complexity problem $\mathcal{P}$ is tuple $\langle \to, \mathcal{T} \rangle$
  1. $\to$ is binary relation on terms
  2. $\mathcal{T}$ is set of starting terms
- complexity function of $\mathcal{P}$ is

$$\mathsf{cp}_{\mathcal{P}}(n) := \max\{\mathsf{dh}(t, \to) \mid t \in \mathcal{T} \text{ is term of size upto } n\}$$

A Framework For Complexity Analysis of TRSs

complexity problem

- complexity problem $\mathcal{P}$ is tuple $\langle \mathcal{S}/\mathcal{W}, \quad, \mathcal{T} \rangle$
  1. $\rightarrow$ is binary relation on terms

  $$\rightarrow_{\mathcal{S}/\mathcal{W}} := \rightarrow_{\mathcal{W}}^{*} \cdot \rightarrow_{\mathcal{S}} \cdot \rightarrow_{\mathcal{W}}^{*}$$

     - $\mathcal{S}$,$\mathcal{W}$ are TRSs

  2. $\mathcal{T}$ is set of starting terms

- complexity function of $\mathcal{P}$ is

$$\mathsf{cp}_{\mathcal{P}}(n) := \max\{\mathsf{dh}(t, \rightarrow) \mid t \in \mathcal{T} \text{ is term of size upto } n\}$$

A Framework For Complexity Analysis of TRSs

complexity problem

- complexity problem $\mathcal{P}$ is tuple $\langle \mathcal{S}/\mathcal{W}, \quad, \mathcal{T} \rangle$
  1. $\to$ is binary relation on terms

$$\to_{\mathcal{S}/\mathcal{W}} := \to^*_{\mathcal{W}} \cdot \to_{\mathcal{S}} \cdot \to^*_{\mathcal{W}}$$

  - $\mathcal{S}$,$\mathcal{W}$ are TRSs

  2. $\mathcal{T}$ is set of starting terms

- complexity function of $\mathcal{P}$ is

$$\mathsf{cp}_{\mathcal{P}}(n) := \max\{\mathsf{dh}(t, \to) \mid t \in \mathcal{T} \text{ is term of size upto } n\}$$

- runtime complexity problem if in terms of $\mathcal{T}$, arguments are values

A Framework For Complexity Analysis of TRSs

complexity problem

- complexity problem $\mathcal{P}$ is tuple $\langle \mathcal{S}/\mathcal{W}, \mathcal{Q}, \mathcal{T} \rangle$
  1. $\to$ is binary relation on terms

$$\xrightarrow{\mathcal{Q}}_{\mathcal{S}/\mathcal{W}} := \xrightarrow{\mathcal{Q}}^*_{\mathcal{W}} \cdot \xrightarrow{\mathcal{Q}}_{\mathcal{S}} \cdot \xrightarrow{\mathcal{Q}}^*_{\mathcal{W}}$$

     - $\mathcal{S}$,$\mathcal{W}$ are TRSs
     - $s \xrightarrow{\mathcal{Q}}_{\mathcal{R}} t$ if $s \to_{\mathcal{R}} t$ and arguments of redex in s are $\mathcal{Q}$ normal forms

  2. $\mathcal{T}$ is set of starting terms

- complexity function of $\mathcal{P}$ is

$$\mathsf{cp}_{\mathcal{P}}(n) := \max\{\mathsf{dh}(t, \to) \mid t \in \mathcal{T} \text{ is term of size upto } n\}$$

- runtime complexity problem if in terms of $\mathcal{T}$, arguments are values
- innermost complexity problem if normal forms of $\mathcal{Q}$ are normal forms of $\mathcal{S} \cup \mathcal{W}$

A Framework For Complexity Analysis of TRSs

processors and proofs

- complexity processor is inference rule

$$\frac{\vdash \mathcal{P}_1 \colon f_1 \qquad \cdots \qquad \vdash \mathcal{P}_n \colon f_n}{\vdash \mathcal{P} \colon f}$$

- complexity judgement is statement $\vdash \mathcal{P} \colon f$
  - $\mathcal{P}$ is a complexity problem
  - $f \colon \mathbb{N} \to \mathbb{N}$ is bounding function
  - valid if $\mathrm{cp}_{\mathcal{P}}(n) \in \mathcal{O}(f(n))$

A Framework For Complexity Analysis of TRSs

processors and proofs

- complexity processor is inference rule

$$\frac{\vdash \mathcal{P}_1 \colon f_1 \quad \cdots \quad \vdash \mathcal{P}_n \colon f_n}{\vdash \mathcal{P} \colon f}$$

- complexity judgement is statement $\vdash \mathcal{P} \colon f$
  - $\mathcal{P}$ is a complexity problem
  - $f \colon \mathbb{N} \to \mathbb{N}$ is bounding function
  - valid if $\mathsf{cp}_{\mathcal{P}}(n) \in \mathcal{O}(f(n))$

- complexity proof of $\vdash \mathcal{P} \colon f$ is a deduction using sound processors

Small Polynomial Path Order

as complexity processor

$$\frac{\mathcal{S} \subseteq >_{\mathsf{spop}^*} \quad \mathcal{W} \subseteq \geqslant_{\mathsf{spop}^*}}{\vdash \langle \mathcal{S}/\mathcal{W}, \mathcal{Q}, \mathcal{T} \rangle \colon n^d}$$

for innermost runtime complexity problem $\langle \mathcal{S}/\mathcal{W}, \mathcal{Q}, \mathcal{T} \rangle$, where

- d is maximal depth of recursion
- $\mathcal{S} \cup \mathcal{W}$ is a *constructor* TRS

Small Polynomial Path Order

as complexity processor

$$\frac{\mathcal{S} \subseteq >_{\mathsf{spop}^*} \quad \mathcal{W} \subseteq \geqslant_{\mathsf{spop}^*}}{\vdash \langle \mathcal{S}/\mathcal{W}, \mathcal{Q}, \mathcal{T} \rangle \colon n^d}$$

for innermost runtime complexity problem $\langle \mathcal{S}/\mathcal{W}, \mathcal{Q}, \mathcal{T} \rangle$, where

- d is maximal depth of recursion
- $\mathcal{S} \cup \mathcal{W}$ is a *constructor* TRS

Extensions. . .

- permutation of argument positions
- quasi-precedences
- argument filterings

Further Processors

- complexity pairs
  $\mathcal{P}$-monotone complexity pairs
- dependency pairs
  weak dependency pairs, dependency tuples
- simplifications
  usable rules, rule simplifications, predecessor estimation, relative decomposition
- call graph analysis
  dependency graph decomposition

generalised
novel

Experimental Evaluation

testsuite

- RC: set of runtime complexity examples
- RaML: straight forward translations of first-order ML programs

setup

- 16 Intel® Core™ i7-3930K (3.20GHz), 32Gb RAM
- 300 secs timeout

tools

- AProVE *for innermost rewriting*

  http://aprove.informatik.rwth-aachen.de/
- CaT *for full rewriting*

  http://cl-informatik.uibk.ac.at/software/cat/

Experimental Results

	RC				RaML	
	full		innermost		innermost	
Answer	TCT	CaT	TCT	AProVE	TCT	AProVE
$\mathcal{O}(1)$	1/0.90	–	1/0.09	1/1.06	–	–
$\mathcal{O}(n^1)$	7/9.22	6/0.42	8/0.91	8/2.05	3/10.21	2/2.89
$\mathcal{O}(n^2)$	2/4.61	–	11/13.02	11/2.53	13/17.45	6/11.01
$\mathcal{O}(n^3)$	1/44.64	–	3/22.59	3/6.20	3/63.08	1/11.95
$\mathcal{O}(n^4)$	1/52.85	–	2/77.99	–	1/159.03	–
$\mathcal{O}(n^5)$	–	–	2/84.33	–	1/149.30	–
Success	12/14.35	6/0.42	27/20.11	23/2.78	21/34.32	8/9.31
Maybe	–	–	–	1/168.07	–	–
Timeout	18	–	3	6	1	13

Table : Overview experimental evaluation on data set RC and RaML.

Experimental Results

	RC				RaML	
	full		innermost		innermost	
Answer	TCT	CaT	TCT	`AProVE`	TCT	`AProVE`
$\mathcal{O}(1)$	1/0.90	–	1/0.09	1/1.06	–	–
$\mathcal{O}(n^1)$	7/9.22	6/0.42	8/0.91	8/2.05	3/10.21	2/2.89
$\mathcal{O}(n^2)$	2/4.61	–	11/13.02	11/2.53	13/17.45	6/11.01
$\mathcal{O}(n^3)$	1/44.64	–	3/22.59	3/6.20	3/63.08	1/11.95
$\mathcal{O}(n^4)$	1/52.85	–	2/77.99	–	1/159.03	–
$\mathcal{O}(n^5)$	–	–	2/84.33	–	1/149.30	–
Success	12/14.35	6/0.42	27/20.11	23/2.78	21/34.32	8/9.31
Maybe	–	–	–	1/168.07	–	–
Timeout	18	–	3	6	1	13
due to `DGD`	25%	–	44%	–	57%	–

Table : Overview experimental evaluation on data set `RC` and `RaML`.

Part III

Invariance Theorem

Presented at...

M. Avanzini and G. Moser
Complexity Analysis by Graph Rewriting.
Proc. of 10th FLOPS, LNCS, pages 257–271, 2010

M. Avanzini and G. Moser
Closing the Gap Between Runtime Complexity and Polytime Computability.
Proc. of 21st RTA, LIPIcs, pages 33-48, 2010

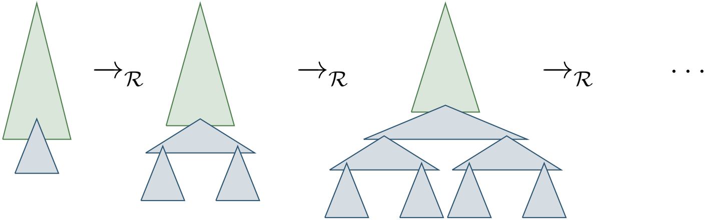
Invariance Thesis

"... reasonable universal machines can simulate each other within a polynomially bounded overhead in time and a constant-factor overhead in space."

P. Van Emde Boas
Machine Models and Simulation.
Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity (A), pp. 1-66, 1990

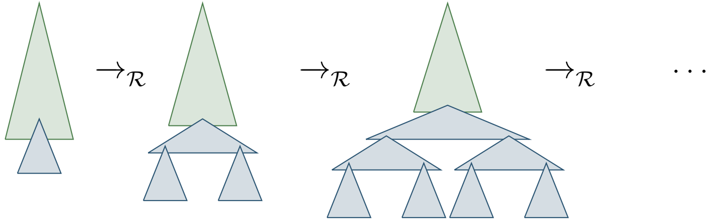
Invariance Thesis

unitary cost model a priory not invariant

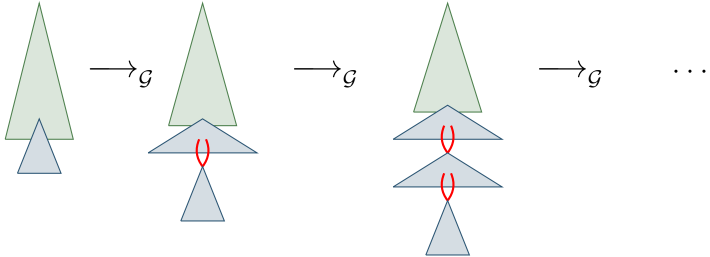


Invariance Thesis

unitary cost model a priory not invariant



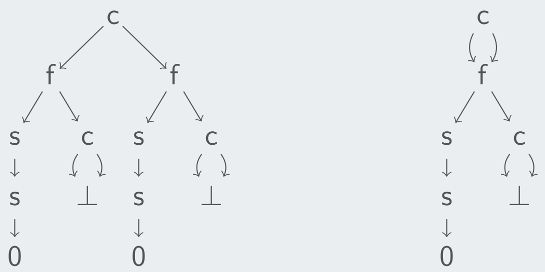
solution



Term Graph Rewriting

in a nutshell

- allow sharing in terms *term graphs*



$\mathsf{c}\big(\mathsf{f}(\mathsf{s}(\mathsf{s}(0)), \mathsf{c}(\perp, \perp)), \mathsf{f}(\mathsf{s}(\mathsf{s}(0)), \mathsf{c}(\perp, \perp))\big)$ term$(\cdot)$

Term Graph Rewriting

in a nutshell

- allow sharing in terms *term graphs*
- never duplicate, introduce sharing instead

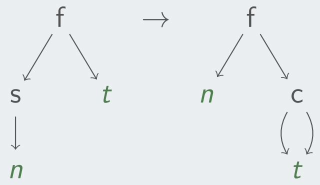
$$
\begin{tikzcd}
& \mathsf{f} \arrow[dl] \arrow[dr] & & \longrightarrow & & \mathsf{f} \arrow[dl] \arrow[dr] & \\
\mathsf{s} \arrow[d] & & t & & n & & \mathsf{c} \arrow[d, bend right] \arrow[d, bend left] \\
n & & & & & & t
\end{tikzcd}
$$

$$\mathsf{f}(\mathsf{s}(n), t) \quad \rightarrow \quad \mathsf{f}(n, \mathsf{c}(t, t))$$

Term Graph Rewriting

in a nutshell

- allow sharing in terms *term graphs*
- never duplicate, introduce sharing instead
- runtime complexity is an invariant cost model for term graph rewriting



$$\mathsf{f}(\mathsf{s}(n), t) \quad \rightarrow \quad \mathsf{f}(n, \mathsf{c}(t, t))$$

Adequacy

connecting term rewriting and term graph rewriting

Definition

Let G be a set of term graphs.
Relation $\longrightarrow_{\mathcal{G}}$ on graphs is adequate (wrt. G) for relation $\rightarrow_{\mathcal{R}}$ on terms if

1. *Surjectivity of unfolding on G:*

   every term t has a graph representation in G

2. *Closure under reductions of G:*

$$S \in G \text{ and } S \longrightarrow_{\mathcal{G}} T \quad \Longrightarrow \quad T \in G$$

3. *Preservation of reductions:*

$$S \in G \text{ and } S \longrightarrow_{\mathcal{G}} T \quad \Longrightarrow \quad \mathsf{term}(S) \rightarrow_{\mathcal{R}} \mathsf{term}(T)$$

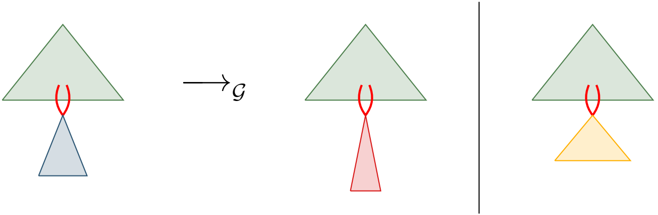
4. *Simulation of reductions:*

$$S \in G \text{ and } \mathsf{term}(S) \rightarrow_{\mathcal{R}} t \quad \Longrightarrow \quad S \longrightarrow_{\mathcal{G}} T \text{ where } \mathsf{term}(T) = t$$

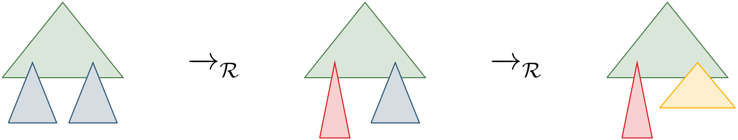
Sources of Inadequacy

(i) shared redexes

on term graphs ...



... on terms

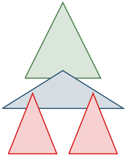


Sources of Inadequacy

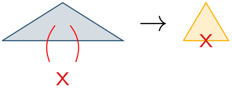
(ii) graph matching

on terms . . .

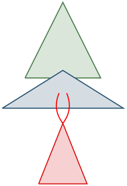
matches



. . . on term graphs



matches only



Adequacy

by folklore

for every TRS $\mathcal{R}$, there exists a GRS $\mathcal{G}_{\mathcal{R}}$ such that the relation

$$\longrightarrow_{\mathcal{G}_{\mathcal{R}}} + \text{ sharing } + \text{ unsharing}$$

is adequate for $\rightarrow_{\mathcal{R}}$

Adequacy

by folklore

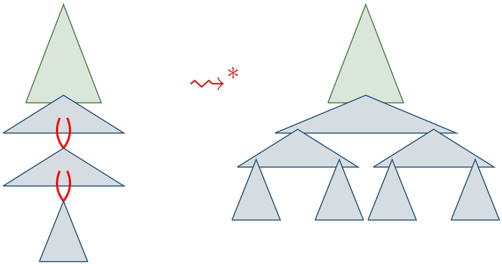
for every TRS $\mathcal{R}$, there exists a GRS $\mathcal{G}_{\mathcal{R}}$ such that the relation

$$\longrightarrow_{\mathcal{G}_{\mathcal{R}}} + \text{sharing} + \text{unsharing}$$

is adequate for $\rightarrow_{\mathcal{R}}$

however...

- uncontrolled unsharing might blow up graph sizes exponentially

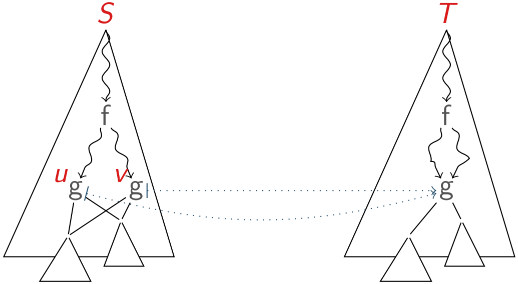


Recovering Adequacy

restricted sharing and unsharing

define for term graphs S, T

- $S \sqsupset^u_v T$:⇔ "T obtained from S by identifying nodes u and v"

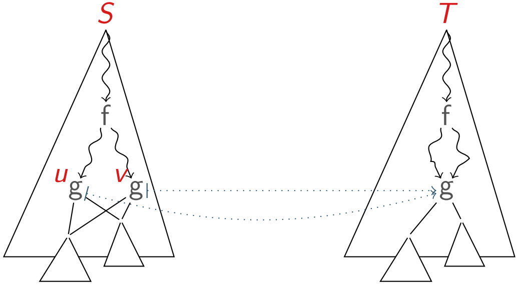


Recovering Adequacy

restricted sharing and unsharing

define for term graphs S, T

- $S \sqsupset^u_v T :\Leftrightarrow$ "T obtained from S by identifying nodes u and v"



❶ $S \blacktriangleright_p T :\Leftrightarrow S \sqsupset^u_v T$ for nodes $u, v \in S$ strictly below position p

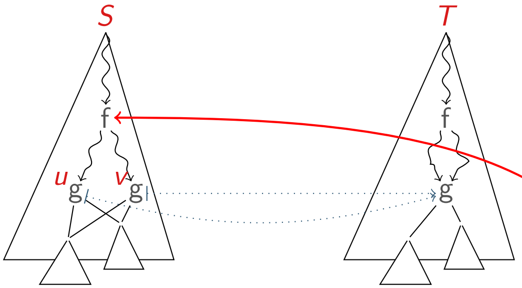
❷ $T \lhd_p S :\Leftrightarrow T \,{}^u_v\!\sqsubset S$ and $u \in S$ unshared node above position p

Recovering Adequacy

restricted sharing and unsharing

define for term graphs S, T

- $S \sqsupset^u_v T :\Leftrightarrow$ "T obtained from S by identifying nodes u and v"



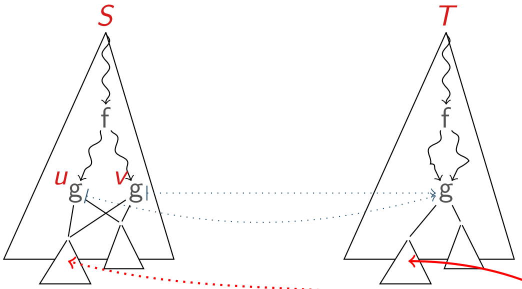
1. $S \blacktriangleright_p T :\Leftrightarrow S \sqsupset^u_v T$ for nodes $u, v \in S$ strictly below position p
2. $T \lhd_p S :\Leftrightarrow T \,{}^u_v\!\sqsubset S$ and $u \in S$ unshared node above position p

Recovering Adequacy

restricted sharing and unsharing

define for term graphs S, T

- $S \sqsupset^u_v T :\Leftrightarrow$ "T obtained from S by identifying nodes u and v"



❶ $S \blacktriangleright_p T :\Leftrightarrow S \sqsupset^u_v T$ for nodes $u, v \in S$ strictly below position p

❷ $T \lhd_p S :\Leftrightarrow T\ {}^u_v{\sqsubset}\ S$ and $u \in S$ unshared node above position p

Adequacy Theorem

full rewriting

Theorem

the following relations are adequate for $\rightarrow_{\mathcal{R}}$:

- in general

$$S \mathrel{\triangleleft\!\!\blacktriangleright\!\!\rightarrow}_{\mathcal{G}_{\mathcal{R}}} T \;:\Longleftrightarrow\; S \triangleleft_p^! \cdot \blacktriangleright_p^! \cdot \longrightarrow_{\mathcal{G}_{\mathcal{R}},p} T$$

- when $\mathcal{R}$ is left-linear

$$S \mathrel{\triangleleft\!\!\rightarrow}_{\mathcal{G}_{\mathcal{R}}} T \;:\Longleftrightarrow\; S \triangleleft_p^! \cdot \longrightarrow_{\mathcal{G}_{\mathcal{R}},p} T$$

Adequacy Theorem

full rewriting

Theorem

the following relations are adequate for $\rightarrow_{\mathcal{R}}$:

- in general

$$S \mathrel{\triangleleft\!\!\blacktriangleright\!\!\rightarrow}_{\mathcal{G}_{\mathcal{R}}} T :\Longleftrightarrow S \triangleleft^{!}_{p} \cdot \blacktriangleright^{!}_{p} \cdot \longrightarrow_{\mathcal{G}_{\mathcal{R}},p} T$$

- when $\mathcal{R}$ is left-linear

$$S \mathrel{\triangleleft\!\!\longrightarrow}_{\mathcal{G}_{\mathcal{R}}} T :\Longleftrightarrow S \triangleleft^{!}_{p} \cdot \longrightarrow_{\mathcal{G}_{\mathcal{R}},p} T$$

Lemma *space lemma*

$$S \mathrel{\triangleleft\!\!\blacktriangleright\!\!\rightarrow}^{\ell}_{\mathcal{G}_{\mathcal{R}}} T \text{ or } S \mathrel{\triangleleft\!\!\longrightarrow}^{\ell}_{\mathcal{G}_{\mathcal{R}}} T \implies \mathsf{size}(T) \in \mathcal{O}(\ell \cdot \mathsf{size}(S) + \ell^2)$$

Adequacy Theorem

innermost rewriting

Theorem

the following relations are adequate for $\xrightarrow{\mathsf{i}}_{\mathcal{R}}$ on NF-sharing graphs:

- in general

$$S \blacktriangleright\!\!\xrightarrow{\mathsf{i}}_{\mathcal{G}_{\mathcal{R}}} T :\Longleftrightarrow \quad S \blacktriangleright^{!}_{p} \cdot \xrightarrow{\mathsf{i}}_{\mathcal{G}_{\mathcal{R}},p} T$$

- when $\mathcal{R}$ is left-linear

$$S \xrightarrow{\mathsf{i}}_{\mathcal{G}_{\mathcal{R}}} T$$

Lemma *space lemma*

$$S \blacktriangleright\!\!\xrightarrow{\mathsf{i}}{}^{\ell}_{\mathcal{G}_{\mathcal{R}}} T \;\text{ or }\; S \xrightarrow{\mathsf{i}}{}^{\ell}_{\mathcal{G}_{\mathcal{R}}} T \quad \Longrightarrow \quad \mathsf{size}(T) \in \mathcal{O}(\mathsf{size}(S) + \ell)$$

Invariance Theorems

deterministic case

Theorem *deterministic invariance theorem*

Let $\mathcal{R}$ be a confluent TRS with $\mathsf{rc}_{\mathcal{R}}(n) \in \mathcal{O}(g(n))$.

Any function computed by $\mathcal{R}$ is computable in time $p(n, g(n))$ on a *deterministic Turing machine*, where

$$p(n, \ell) \in \mathcal{O}\big(\log(\ell + n)^3 \cdot (\ell \cdot n^3 + \ell^4)\big)$$

Invariance Theorems

deterministic case

Theorem *deterministic invariance theorem*

Let $\mathcal{R}$ be a confluent TRS with $\mathsf{rc}_{\mathcal{R}}(n) \in \mathcal{O}(g(n))$.

Any function computed by $\mathcal{R}$ is computable in time $p(n, g(n))$ on a *deterministic Turing machine*, where

$$p(n, \ell) \in \mathcal{O}\big(\log(\ell + n)^3 \cdot (\ell \cdot n^3 + \ell^4)\big)$$

Corollary *Polynomial Time*

Let $\mathcal{R}$ be a confluent TRS with $\mathsf{rc}_{\mathcal{R}}(n) \in \mathcal{O}(n^k)$ for some $k \geqslant 0$.

Then all functions computed by $\mathcal{R}$ are computable in polynomial time, i.e, are in FP.

Invariance Theorems

non-deterministic case

Theorem *non-deterministic invariance theorem*

Let $\mathcal{R}$ be a TRS with $\mathsf{rc}_{\mathcal{R}}(n) \in \mathcal{O}(g(n))$.

Any relation computed by $\mathcal{R}$ is computable in time $p(n, g(n))$ on a *non-deterministic Turing machine*, where

$$p(n, \ell) \in \mathcal{O}\big(\log(\ell \cdot n)^2 \cdot (\ell^3 \cdot n^2 + \ell^5)\big)$$

Invariance Theorems

non-deterministic case

Theorem *non-deterministic invariance theorem*

Let $\mathcal{R}$ be a TRS with $\mathsf{rc}_{\mathcal{R}}(n) \in \mathcal{O}(g(n))$.

Any relation computed by $\mathcal{R}$ is computable in time $p(n, g(n))$ on a *non-deterministic Turing machine*, where

$$p(n, \ell) \in \mathcal{O}\big(\log(\ell \cdot n)^2 \cdot (\ell^3 \cdot n^2 + \ell^5)\big)$$

Corollary *function problems computable in non-deterministic time*

Let $\mathcal{R}$ be a TRS with $\mathsf{rc}_{\mathcal{R}}(n) \in \mathcal{O}(n^k)$ for some $k \geqslant 0$.

Then the function problem associated with any relation computed by $\mathcal{R}$ is in FNP.

Thanks!