Closing the Gap Between Runtime Complexity and Polytime Computability

Martin Avanzini and Georg Moser

Computational Logic
Faculty of Computer Science, University of Innsbruck

RTA '10

Simple "Functional Program"

① $\mathsf{d}(\mathsf{c}) = 0$

② $\mathsf{d}(x \times y) = \mathsf{d}(x) \times y + x \times \mathsf{d}(y)$

③ $\mathsf{d}(x + y) = \mathsf{d}(x) + \mathsf{d}(y)$

④ $\mathsf{d}(x - y) = \mathsf{d}(x) - \mathsf{d}(y)$

data Exp = Zero	0
| Const	c
| Times Exp Exp	$e_1 \times e_2$
| Plus Exp Exp	$e_1 + e_2$
| Minus Exp Exp	$e_1 - e_2$

Simple "Functional Program"

① $\mathsf{d}(\mathsf{c}) = 0$

② $\mathsf{d}(x \times y) = \mathsf{d}(x) \times y + x \times \mathsf{d}(y)$

③ $\mathsf{d}(x + y) = \mathsf{d}(x) + \mathsf{d}(y)$

④ $\mathsf{d}(x - y) = \mathsf{d}(x) - \mathsf{d}(y)$

Computation

$$\mathsf{d}(\mathsf{c} \times \mathsf{c})$$

Simple "Functional Program"

① $\mathsf{d}(\mathsf{c}) = 0$

② $\mathsf{d}(x \times y) = \mathsf{d}(x) \times y + x \times \mathsf{d}(y)$

③ $\mathsf{d}(x + y) = \mathsf{d}(x) + \mathsf{d}(y)$

④ $\mathsf{d}(x - y) = \mathsf{d}(x) - \mathsf{d}(y)$

Computation

$$\mathsf{d}(\mathsf{c} \times \mathsf{c}) = \mathsf{d}(\mathsf{c}) \times \mathsf{c} + \mathsf{c} \times \mathsf{d}(\mathsf{c})$$

Simple "Functional Program"

① $\mathsf{d}(\mathsf{c}) = 0$

② $\mathsf{d}(x \times y) = \mathsf{d}(x) \times y + x \times \mathsf{d}(y)$

③ $\mathsf{d}(x + y) = \mathsf{d}(x) + \mathsf{d}(y)$

④ $\mathsf{d}(x - y) = \mathsf{d}(x) - \mathsf{d}(y)$

Computation

$$\mathsf{d}(\mathsf{c} \times \mathsf{c}) = 0 \times \mathsf{c} + \mathsf{c} \times \mathsf{d}(\mathsf{c})$$

Simple "Functional Program"

① $\mathsf{d}(\mathsf{c}) = 0$

② $\mathsf{d}(x \times y) = \mathsf{d}(x) \times y + x \times \mathsf{d}(y)$

③ $\mathsf{d}(x + y) = \mathsf{d}(x) + \mathsf{d}(y)$

④ $\mathsf{d}(x - y) = \mathsf{d}(x) - \mathsf{d}(y)$

Computation

$$\mathsf{d}(\mathsf{c} \times \mathsf{c}) = 0 \times \mathsf{c} + \mathsf{c} \times 0$$

Simple "Functional Program" $\approx$ Term Rewrite System (TRS)

① $\mathsf{d}(\mathsf{c}) \to 0$

② $\mathsf{d}(x \times y) \to \mathsf{d}(x) \times y + x \times \mathsf{d}(y)$

③ $\mathsf{d}(x + y) \to \mathsf{d}(x) + \mathsf{d}(y)$

④ $\mathsf{d}(x - y) \to \mathsf{d}(x) - \mathsf{d}(y)$

Computation $\approx$ Rewriting

$$\mathsf{d}(\mathsf{c} \times \mathsf{c}) \to^{!}_{\mathcal{R}} 0 \times \mathsf{c} + \mathsf{c} \times 0$$

Simple “Functional Program” $\approx$ Term Rewrite System (TRS)

① $\mathsf{d}(\mathsf{c}) \to 0$

② $\mathsf{d}(x \times y) \to \mathsf{d}(x) \times y + x \times \mathsf{d}(y)$

③ $\mathsf{d}(x + y) \to \mathsf{d}(x) + \mathsf{d}(y)$

④ $\mathsf{d}(x - y) \to \mathsf{d}(x) - \mathsf{d}(y)$

Computation $\approx$ Rewriting

$$\mathsf{d}(\mathsf{c} \times \mathsf{c}) \to^{!}_{\mathcal{R}} 0 \times \mathsf{c} + \mathsf{c} \times 0$$

Runtime Complexity

“number of reduction steps as function in the size of the initial terms”

Simple "Functional Program" $\approx$ Term Rewrite System (TRS)

① $\mathsf{d}(\mathsf{c}) \to 0$

② $\mathsf{d}(x \times y) \to \mathsf{d}(x) \times y + x \times \mathsf{d}(y)$

③ $\mathsf{d}(x + y) \to \mathsf{d}(x) + \mathsf{d}(y)$

④ $\mathsf{d}(x - y) \to \mathsf{d}(x) - \mathsf{d}(y)$

Computation $\approx$ Rewriting

$$\mathsf{d}(\mathsf{c} \times \mathsf{c}) \to^{!}_{\mathcal{R}} 0 \times \mathsf{c} + \mathsf{c} \times 0$$

Runtime Complexity

"number of reduction steps as function in the size of the initial terms"

- initial terms are argument normalised

let $\mathcal{C}$ collect all constructor symbols and let $\mathcal{V}\mathsf{al}$ abbreviate $\mathcal{T}(\mathcal{C}, \mathcal{V})$

Definition (Computation)

Let $\mathcal{R}$ denote a confluent and terminating TRS.
$\mathcal{R}$ computes a function $f \colon \mathcal{V}\mathsf{al}^k \to \mathcal{V}\mathsf{al}$ if $\exists$ function symbol f

$$f(v_1, \ldots, v_k) = w \qquad \Longleftrightarrow \qquad \mathsf{f}(v_1, \ldots, v_k) \to^!_{\mathcal{R}} w$$

let $\mathcal{C}$ collect all constructor symbols and let $\mathcal{V}\mathsf{al}$ abbreviate $\mathcal{T}(\mathcal{C}, \mathcal{V})$

Definition (Computation)

Let $\mathcal{R}$ denote a ~~confluent~~ and terminating TRS.
$\mathcal{R}$ computes a relation $R \subseteq \mathcal{V}\mathsf{al}^k \times \mathcal{V}\mathsf{al}$ if $\exists$ function symbol f

$$(v_1, \ldots, v_k, w) \in R \qquad \Longleftrightarrow \qquad \mathsf{f}(v_1, \ldots, v_k) \to^!_{\mathcal{R}} w \text{ and } w \text{ accepting}$$

let $\mathcal{C}$ collect all constructor symbols and let $\mathcal{V}\mathrm{al}$ abbreviate $\mathcal{T}(\mathcal{C}, \mathcal{V})$

Definition (Computation)

Let $\mathcal{R}$ denote a ~~confluent~~ and terminating TRS.
$\mathcal{R}$ computes a relation $R \subseteq \mathcal{V}\mathrm{al}^k \times \mathcal{V}\mathrm{al}$ if $\exists$ function symbol f

$$(v_1, \dots, v_k, w) \in R \quad \Longleftrightarrow \quad \mathsf{f}(v_1, \dots, v_k) \to^!_{\mathcal{R}} w \text{ and } w \text{ accepting}$$

Definition (Runtime Complexity)

$$\mathrm{rc}_{\mathcal{R}}(n) = \max\{ \mathrm{dl}(t, \to_{\mathcal{R}}) \mid |t| \leqslant n \qquad \}$$

where $\mathrm{dl}(t, \to_{\mathcal{R}}) = \max\{\ell \mid \exists (t_1, \dots, t_\ell).\ t \to_{\mathcal{R}} t_1 \to_{\mathcal{R}} \cdots \to_{\mathcal{R}} t_\ell\}$

let $\mathcal{C}$ collect all constructor symbols and let $\mathcal{V}\mathsf{al}$ abbreviate $\mathcal{T}(\mathcal{C},\mathcal{V})$

Definition (Computation)

Let $\mathcal{R}$ denote a ~~confluent~~ and terminating TRS.
$\mathcal{R}$ computes a relation $R \subseteq \mathcal{V}\mathsf{al}^k \times \mathcal{V}\mathsf{al}$ if $\exists$ function symbol f

$$(v_1,\dots,v_k,w) \in R \qquad \Longleftrightarrow \qquad \mathsf{f}(v_1,\dots,v_k) \to^{!}_{\mathcal{R}} w \text{ and } w \text{ accepting}$$

Definition (Runtime Complexity)

$$\mathsf{rc}_{\mathcal{R}}(n) = \max\{\, \mathsf{dl}(t,\to_{\mathcal{R}}) \mid |t| \leqslant n \text{ and arguments from } \mathcal{V}\mathsf{al} \,\}$$

where $\mathsf{dl}(t,\to_{\mathcal{R}}) = \max\{\ell \mid \exists (t_1,\dots,t_\ell).\ t \to_{\mathcal{R}} t_1 \to_{\mathcal{R}} \cdots \to_{\mathcal{R}} t_\ell\}$

Example

① $\mathsf{d}(\mathsf{c}) \to 0$

② $\mathsf{d}(x \times y) \to \mathsf{d}(x) \times y + x \times \mathsf{d}(y)$

③ $\mathsf{d}(x + y) \to \mathsf{d}(x) + \mathsf{d}(y)$

④ $\mathsf{d}(x - y) \to \mathsf{d}(x) - \mathsf{d}(y)$

Example

① $\mathsf{d}(\mathsf{c}) \to 0$

② $\mathsf{d}(x \times y) \to \mathsf{d}(x) \times y + x \times \mathsf{d}(y)$

③ $\mathsf{d}(x + y) \to \mathsf{d}(x) + \mathsf{d}(y)$

④ $\mathsf{d}(x - y) \to \mathsf{d}(x) - \mathsf{d}(y)$

- derivational complexity of above TRS is at least exponential

Example

① $\mathsf{d}(\mathsf{c}) \to 0$

② $\mathsf{d}(x \times y) \to \mathsf{d}(x) \times y + x \times \mathsf{d}(y)$

③ $\mathsf{d}(x + y) \to \mathsf{d}(x) + \mathsf{d}(y)$

④ $\mathsf{d}(x - y) \to \mathsf{d}(x) - \mathsf{d}(y)$

- derivational complexity of above TRS is at least exponential
- runtime complexity of above TRS is linear

```
$ tct -a rc -p dif.trs
  YES(?,O(n^1))

 'Weak Dependency Pairs'
  -----------------------
  Answer:           YES(?,O(n^1))
  Input Problem:    runtime-complexity with respect to
    Rules:
       { d(c) -> 0()
       , d(*(x, y)) -> +(*(y, d(x)), *(x, d(y)))
       , d(+(x, y)) -> +(d(x), d(y))
       , d(-(x, y)) -> -(d(x), d(y))}
  Proof Details:
   ...
```

Example

① $\mathsf{d}(\mathsf{c}) \to 0$

② $\mathsf{d}(x \times y) \to \mathsf{d}(x) \times y + x \times \mathsf{d}(y)$

③ $\mathsf{d}(x + y) \to \mathsf{d}(x) + \mathsf{d}(y)$

④ $\mathsf{d}(x - y) \to \mathsf{d}(x) - \mathsf{d}(y)$

- derivational complexity of above TRS is at least exponential
- runtime complexity of above TRS is linear

```
$ tct -a rc -p dif.trs
  YES(?,O(n^1))

 'Weak Dependency Pairs'
  -----------------------
  Answer:           YES(?,O(n^1))
  Input Problem:    runtime-complexity with re
    Rules:
       { d(c) -> 0()
       , d(*(x, y)) -> +(*(y, d(x)), *(x, d(y)))
       , d(+(x, y)) -> +(d(x), d(y))
       , d(-(x, y)) -> -(d(x), d(y))}
  Proof Details:
   ...
```

Our Question

what can we infer about the computational complexity from this proof?

Example

① $\mathsf{d}(\mathsf{c}) \to 0$

② $\mathsf{d}(x \times y) \to \mathsf{d}(x) \times y + x \times \mathsf{d}(y)$

③ $\mathsf{d}(x + y) \to \mathsf{d}(x) + \mathsf{d}(y)$

④ $\mathsf{d}(x - y) \to \mathsf{d}(x) - \mathsf{d}(y)$

- derivational complexity of above TRS is at least exponential
- runtime complexity of above TRS is linear

```
$ tct -a rc -p dif.trs
  YES(?,O(n^1))

 'Weak Dependency Pairs'
  -----------------------
  Answer:           YES(?,O(n^1))
  Input Problem:    runtime-complexity with re
    Rules:
       { d(c) -> 0()
       , d(*(x, y)) -> +(*(y, d(x)), *(x, d(y)))
       , d(+(x, y)) -> +(d(x), d(y))
       , d(-(x, y)) -> -(d(x), d(y))}
  Proof Details:
   ...
```

Our Question

In particular, does it certify polytime computability of the functions defined?

Main Result

Yes

Main Result

runtime complexity is a reasonable cost model for rewriting

Main Result

runtime complexity is a reasonable cost model for rewriting

1. runtime complexity naturally expresses the cost of computation

Main Result

runtime complexity is a reasonable cost model for rewriting

1. runtime complexity naturally expresses the cost of computation
2. polynomially related to actual cost of an implementation

   on a Turing machine

Main Result

Theorem

For any term s with $\mathrm{dl}_{\mathcal{R}}(s) \leqslant \ell$ $\qquad \ell = \Omega(|s|)$

1. *some normal-form is computable in deterministic time* $\mathrm{O}(\log(\ell)^3 \ell^7)$

Main Result

Theorem

For any term s with $\mathrm{dl}_{\mathcal{R}}(s) \leqslant \ell$ $\qquad \ell = \Omega(|s|)$

1. *some normal-form is computable in deterministic time* $\mathrm{O}(\log(\ell)^3 \ell^7)$

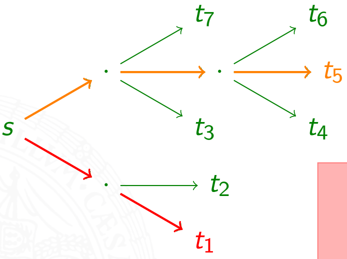
$$\begin{tikzcd}
 & & t_7 & t_6 \\
 & \cdot \arrow[ur] \arrow[r] \arrow[dr] & \cdot \arrow[ur] \arrow[r] \arrow[dr] & t_5 \\
s \arrow[ur] \arrow[dr] & & t_3 & t_4 \\
 & \cdot \arrow[r] \arrow[dr] & t_2 & \\
 & & t_1 &
\end{tikzcd}$$

Main Result

Theorem

For any term s with $\mathrm{dl}_{\mathcal{R}}(s) \leqslant \ell$ $\qquad \ell = \Omega(|s|)$

1. *some normal-form is computable in deterministic time* $\mathrm{O}(\log(\ell)^3 \ell^7)$



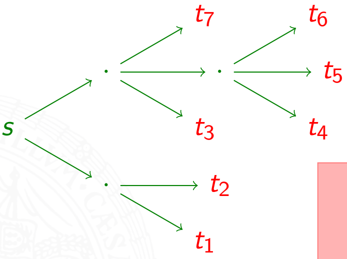
computed NF depends on employed strategy

Main Result

Theorem

For any term s with $\mathrm{dl}_{\mathcal{R}}(s) \leqslant \ell$ $\qquad \ell = \Omega(|s|)$

1. *some normal-form is computable in deterministic time* $\mathrm{O}(\log(\ell)^3 \ell^7)$
2. *any normal-form is computable in nondeterministic time* $\mathrm{O}(\log(\ell)^2 \ell^5)$



choice of redex nondeterministically

Main Result

Difficulty

a single rewrite step may copy arbitrarily large terms

- terms may grow exponential in the length of derivations

Main Result

Difficulty

a single rewrite step may copy arbitrarily large terms

- terms may grow exponential in the length of derivations

Example

① $\mathsf{d}(\mathsf{c}) \to 0$

② $\mathsf{d}(x \times y) \to \mathsf{d}(x) \times y + x \times \mathsf{d}(y)$

③ $\mathsf{d}(x + y) \to \mathsf{d}(x) + \mathsf{d}(y)$

④ $\mathsf{d}(x - y) \to \mathsf{d}(x) - \mathsf{d}(y)$

Main Result
Difficulty

a single rewrite step may copy arbitrarily large terms

- terms may grow exponential in the length of derivations

Example

① $\mathsf{d}(\mathsf{c}) \to 0$

② $\mathsf{d}(x \times y) \to \mathsf{d}(x) \times y + x \times \mathsf{d}(y)$

③ $\mathsf{d}(x + y) \to \mathsf{d}(x) + \mathsf{d}(y)$

④ $\mathsf{d}(x - y) \to \mathsf{d}(x) - \mathsf{d}(y)$

$$\mathsf{d}(\mathsf{c}) \to^{!}_{\mathcal{R}} 0$$

Main Result

Difficulty

a single rewrite step may copy arbitrarily large terms

- terms may grow exponential in the length of derivations

Example

① $\mathsf{d}(\mathsf{c}) \to 0$

② $\mathsf{d}(x \times y) \to \mathsf{d}(x) \times y + x \times \mathsf{d}(y)$

③ $\mathsf{d}(x + y) \to \mathsf{d}(x) + \mathsf{d}(y)$

④ $\mathsf{d}(x - y) \to \mathsf{d}(x) - \mathsf{d}(y)$

$$\mathsf{d}(\mathsf{c}) \to^{!}_{\mathcal{R}} 0$$
$$\mathsf{d}(\mathsf{c} \times \mathsf{c}) \to^{!}_{\mathcal{R}} 0 \times \mathsf{c} + \mathsf{c} \times 0$$

a single rewrite step may copy arbitrarily large terms

- terms may grow exponential in the length of derivations

Example

① $\mathsf{d}(\mathsf{c}) \to 0$

② $\mathsf{d}(x \times y) \to \mathsf{d}(x) \times y + x \times \mathsf{d}(y)$

③ $\mathsf{d}(x + y) \to \mathsf{d}(x) + \mathsf{d}(y)$

④ $\mathsf{d}(x - y) \to \mathsf{d}(x) - \mathsf{d}(y)$

$$\begin{aligned} \mathsf{d}(\mathsf{c}) &\to^!_{\mathcal{R}} 0 \\ \mathsf{d}(\mathsf{c} \times \mathsf{c}) &\to^!_{\mathcal{R}} 0 \times \mathsf{c} + \mathsf{c} \times 0 \\ \mathsf{d}((\mathsf{c} \times \mathsf{c}) \times \mathsf{c}) &\to^!_{\mathcal{R}} (0 \times \mathsf{c} + \mathsf{c} \times 0) \times \mathsf{c} + (\mathsf{c} \times \mathsf{c}) \times 0 \end{aligned}$$

a single rewrite step may copy arbitrarily large terms

- terms may grow exponential in the length of derivations

Example

① $\mathsf{d}(\mathsf{c}) \to 0$

② $\mathsf{d}(x \times y) \to \mathsf{d}(x) \times y + x \times \mathsf{d}(y)$

③ $\mathsf{d}(x + y) \to \mathsf{d}(x) + \mathsf{d}(y)$

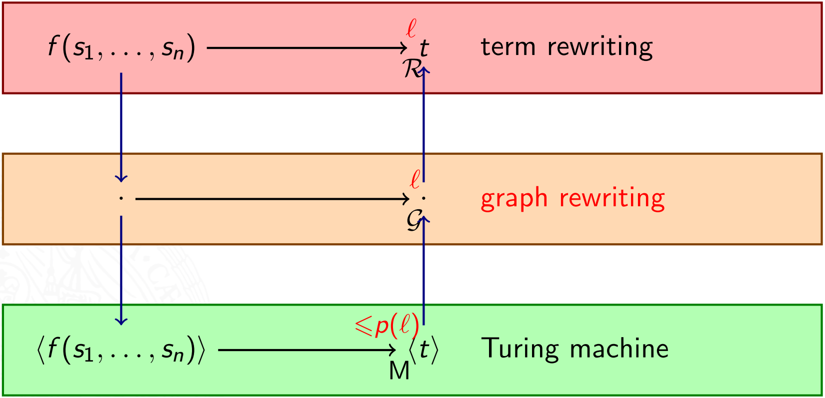
④ $\mathsf{d}(x - y) \to \mathsf{d}(x) - \mathsf{d}(y)$

$$\begin{aligned}
\mathsf{d}(\mathsf{c}) &\to^{!}_{\mathcal{R}} 0 \\
\mathsf{d}(\mathsf{c} \times \mathsf{c}) &\to^{!}_{\mathcal{R}} 0 \times \mathsf{c} + \mathsf{c} \times 0 \\
\mathsf{d}((\mathsf{c} \times \mathsf{c}) \times \mathsf{c}) &\to^{!}_{\mathcal{R}} (0 \times \mathsf{c} + \mathsf{c} \times 0) \times \mathsf{c} + (\mathsf{c} \times \mathsf{c}) \times 0 \\
\mathsf{d}((\mathsf{c} \times \mathsf{c}) \times (\mathsf{c} \times \mathsf{c})) &\to^{!}_{\mathcal{R}} ((0 \times \mathsf{c} + \mathsf{c} \times 0) \times \mathsf{c} + (\mathsf{c} \times \mathsf{c}) \times 0) \times (\mathsf{c} \times \mathsf{c}) \\
&\quad + (\mathsf{c} \times \mathsf{c}) \times ((0 \times \mathsf{c} + \mathsf{c} \times 0) \times \mathsf{c} + (\mathsf{c} \times \mathsf{c}) \times 0)
\end{aligned}$$

$$
\begin{tikzcd}
f(s_1,\dots,s_n) \arrow[r, "\ell", "\mathcal{R}"'] \arrow[d] & t & \text{term rewriting} \\
\langle f(s_1,\dots,s_n)\rangle \arrow[r, "\leqslant p(\ell)", "\mathsf{M}"'] & \langle t\rangle \arrow[u] & \text{Turing machine}
\end{tikzcd}
$$

Main Result

Proof Outline



Main Result

Proof Outline

Adequacy of Graph Rewriting – "Complexity Version"

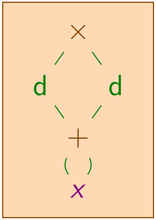
$$
\begin{tikzcd}
f(s_1,\ldots,s_n) \arrow[r, "\ell", "\mathcal{R}"'] \arrow[d] & t & \text{term rewriting} \\
\cdot \arrow[r, "\ell", "\mathcal{G}"'] \arrow[d] & \cdot \arrow[u] & \text{graph rewriting} \\
\langle f(s_1,\ldots,s_n)\rangle \arrow[r, "\leqslant p(\ell)", "\mathsf{M}"'] & \langle t\rangle \arrow[u] & \text{Turing machine}
\end{tikzcd}
$$

Graph Rewriting in a Nutshell

❶ term rewriting on graphs

Example

term $t = \mathsf{d}(x + x) \times \mathsf{d}(x + x)$ represented by



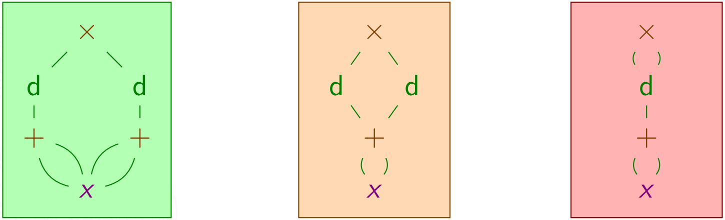
- same variable represented by unique node

Graph Rewriting in a Nutshell

❶ term rewriting on graphs

Example

term $t = \mathsf{d}(x + x) \times \mathsf{d}(x + x)$ represented by



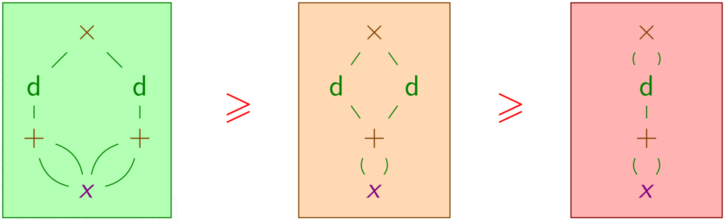
- same variable represented by unique node

Graph Rewriting in a Nutshell

❶ term rewriting on graphs

Example

term $t = \mathsf{d}(x + x) \times \mathsf{d}(x + x)$ represented by



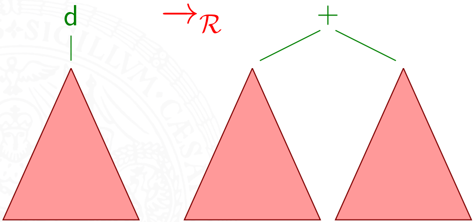
- same variable represented by unique node

Graph Rewriting in a Nutshell

1. term rewriting on graphs
2. copying $\rightsquigarrow$ sharing

Example

Term Rewriting



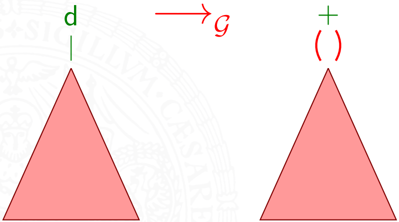
$$\boxed{\mathsf{d}(x) \to x + x}$$

Graph Rewriting in a Nutshell

1. term rewriting on graphs
2. copying $\rightsquigarrow$ sharing

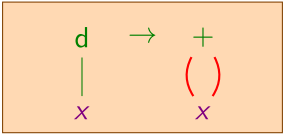
Example

Graph Rewriting



$$\boxed{\mathsf{d}(x) \to x + x}$$

$$\Downarrow$$

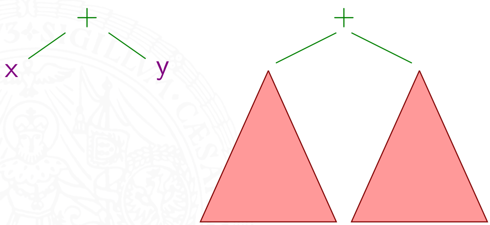


Graph Rewriting in a Nutshell

1. term rewriting on graphs
2. copying ⇝ sharing
3. structural equality ⇝ "pointer equality"

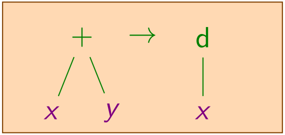
Example

Matching



$$x + y \to \mathsf{d}(x)$$

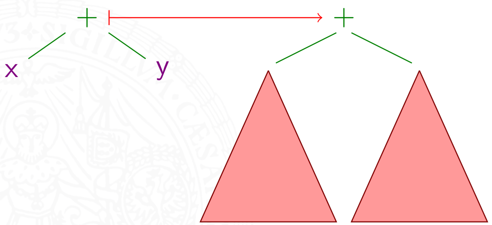
$$\Downarrow$$



1. term rewriting on graphs
2. copying $\rightsquigarrow$ sharing
3. structural equality $\rightsquigarrow$ “pointer equality”

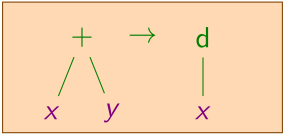
Example

Matching



$$x + y \to \mathsf{d}(x)$$

$$\Downarrow$$

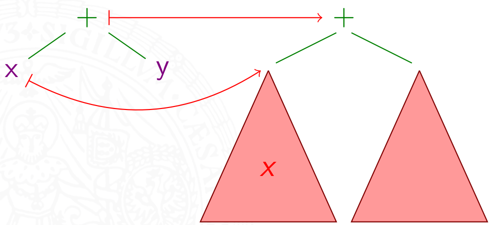


Graph Rewriting in a Nutshell

1. term rewriting on graphs
2. copying $\rightsquigarrow$ sharing
3. structural equality $\rightsquigarrow$ "pointer equality"

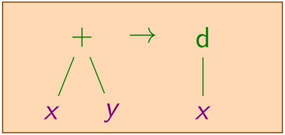
Example

Matching



$$x + y \to \mathsf{d}(x)$$

$$\Downarrow$$

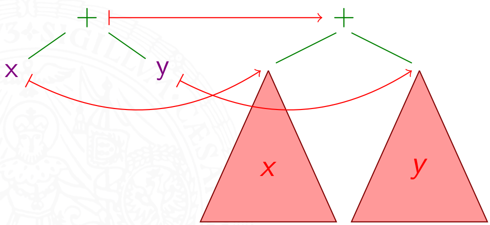


Graph Rewriting in a Nutshell

1. term rewriting on graphs
2. copying $\rightsquigarrow$ sharing
3. structural equality $\rightsquigarrow$ "pointer equality"

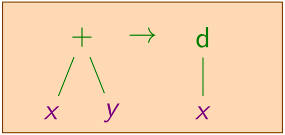
Example

Matching



$$x + y \to \mathsf{d}(x)$$

$$\Downarrow$$

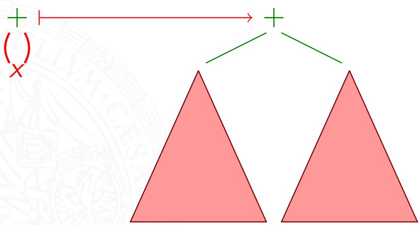


Graph Rewriting in a Nutshell

1. term rewriting on graphs
2. copying $\rightsquigarrow$ sharing
3. structural equality $\rightsquigarrow$ "pointer equality"

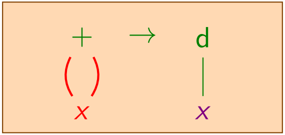
Example

Matching



$$x + x \to \mathsf{d}(x)$$

$$\Downarrow$$

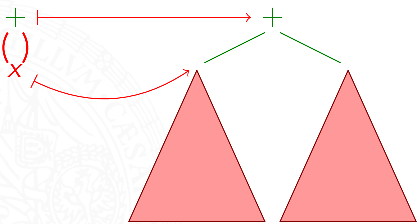


Graph Rewriting in a Nutshell

1. term rewriting on graphs
2. copying $\rightsquigarrow$ sharing
3. structural equality $\rightsquigarrow$ “pointer equality”

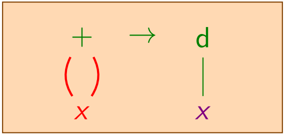
Example

Matching



$$x + x \to \mathsf{d}(x)$$

$$\Downarrow$$

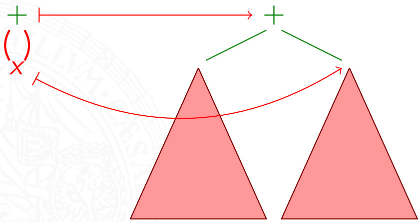


Graph Rewriting in a Nutshell

1. term rewriting on graphs
2. copying $\rightsquigarrow$ sharing
3. structural equality $\rightsquigarrow$ "pointer equality"

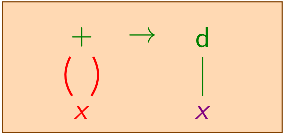
Example

Matching



$$x + x \to \mathsf{d}(x)$$

$$\Downarrow$$

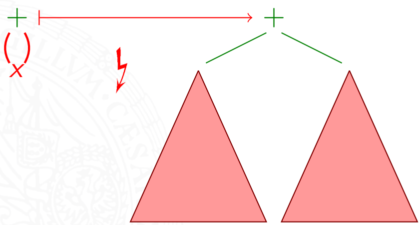


Graph Rewriting in a Nutshell

1. term rewriting on graphs
2. copying $\rightsquigarrow$ sharing
3. structural equality $\rightsquigarrow$ “pointer equality”

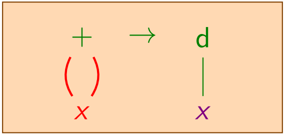
Example

Matching



$$x + x \to \mathsf{d}(x)$$

$$\Downarrow$$

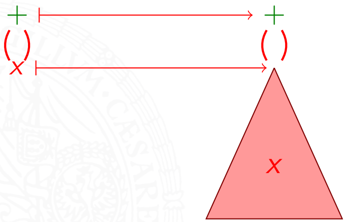


Graph Rewriting in a Nutshell

1. term rewriting on graphs
2. copying $\rightsquigarrow$ sharing
3. structural equality $\rightsquigarrow$ "pointer equality"

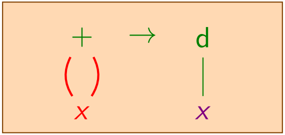
Example

Matching



$$x + x \to \mathsf{d}(x)$$

$$\Downarrow$$



Adequacy of Graph Rewriting

Problems

$$\boxed{x + x \to \mathsf{d}(x)}$$

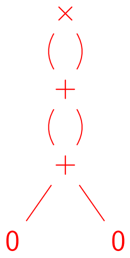
$$\big((0+0)+(0+0)\big) \times \big((0+0)+(0+0)\big)$$

$$\to_{\mathcal{R}} \big((0+0)+(0+0)\big) \times \big((0+0)+\mathsf{d}(0)\big)$$

$$x + x \to \mathsf{d}(x)$$

$$\big((0+0)+(0+0)\big) \times \big((0+0)+(0+0)\big)$$

$$\to_{\mathcal{R}} \big((0+0)+(0+0)\big) \times \big((0+0)+\mathsf{d}(0)\big)$$



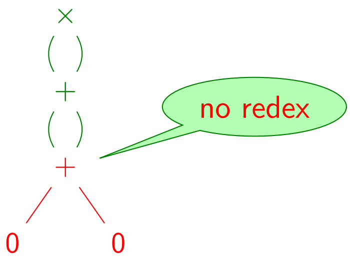
Adequacy of Graph Rewriting

Problems

$$x + x \to \mathsf{d}(x)$$

$$\big((0+0)+(0+0)\big) \times \big((0+0)+(0+0)\big)$$

$$\to_{\mathcal{R}} \big((0+0)+(0+0)\big) \times \big((0+0)+\mathsf{d}(0)\big)$$



$$x + x \to \mathsf{d}(x)$$

$$((0+0)+(0+0)) \times ((0+0)+(0+0))$$

$$\to_{\mathcal{R}} ((0+0)+(0+0)) \times ((0+0)+\mathsf{d}(0))$$

Problem ①

below redex

maximal sharing

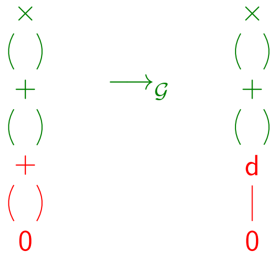
required

$$x + x \to \mathsf{d}(x)$$

$$((0+0)+(0+0)) \times ((0+0)+(0+0))$$

$$\to_{\mathcal{R}} ((0+0)+(0+0)) \times ((0+0)+\mathsf{d}(0))$$

Problem ①
below redex
maximal sharing
required



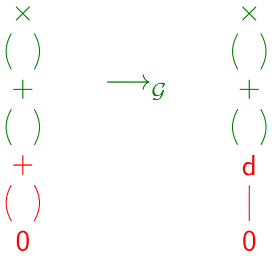
$$x + x \to \mathsf{d}(x)$$

$$((0+0)+(0+0)) \times ((0+0)+(0+0))$$

$$\to_{\mathcal{R}} ((0+0)+(0+0)) \times ((0+0)+\mathsf{d}(0))$$

$$\to^{3}_{\mathcal{R}} (\mathsf{d}(0)+\mathsf{d}(0)) \times (\mathsf{d}(0)+\mathsf{d}(0))$$

Problem ①
below redex
maximal sharing
required



Problem ②
both arguments
of $+\backslash\times$ rewritten

Adequacy of Graph Rewriting

Theorem

suppose S *is a term graph such that*

1. *node corresponding to* p *is unshared*
2. *subgraph* $S \upharpoonright p$ *is maximally shared*

Then

$$S \longrightarrow_{\mathcal{G},p} T \qquad \iff \qquad \mathsf{term}(S) \to_{\mathcal{R},p} \mathsf{term}(T)$$

Adequacy of Graph Rewriting

Theorem

suppose S is a term graph such that

1. *node corresponding to p is unshared*
2. *subgraph $S \upharpoonright p$ is maximally shared*

Then

$$S \longrightarrow_{\mathcal{G},p} T \qquad \Longleftrightarrow \qquad \mathsf{term}(S) \rightarrow_{\mathcal{R},p} \mathsf{term}(T)$$

Idea

- extend rewrite relation $\longrightarrow_{\mathcal{G}}$ with folding and unfolding steps that recover condition ❶ and ❷

$$S \leqslant \cdot \geqslant \cdot \longrightarrow_{\mathcal{G}} T \qquad \Longleftrightarrow \qquad \mathsf{term}(S) \rightarrow_{\mathcal{R}} \mathsf{term}(T)$$

Adequacy of Graph Rewriting

Theorem

suppose S is a term graph such that

1. *node corresponding to p is unshared*
2. *subgraph $S \upharpoonright p$ is maximally shared*

Then

$$S \longrightarrow_{\mathcal{G},p} T \qquad \Longleftrightarrow \qquad \mathsf{term}(S) \rightarrow_{\mathcal{R},p} \mathsf{term}(T)$$

Idea

- extend rewrite relation $\longrightarrow_{\mathcal{G}}$ with folding and unfolding steps that recover condition ❶ and ❷

$$S \leqslant \cdot \geqslant \cdot \longrightarrow_{\mathcal{G}} T \qquad \Longleftrightarrow \qquad \mathsf{term}(S) \rightarrow_{\mathcal{R}} \mathsf{term}(T)$$

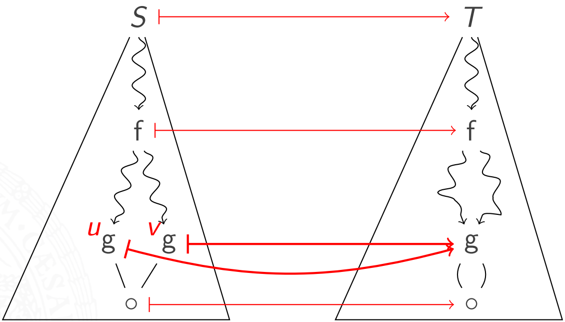
Observation

- unfolding may lead to exponential blowup

Restricted Folding and Unfolding

define for term graphs S, T

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

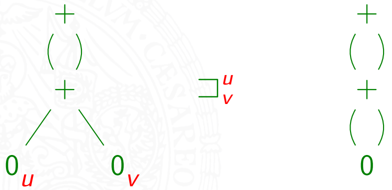


Restricted Folding and Unfolding

define for term graphs S, T

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

Example



Restricted Folding and Unfolding

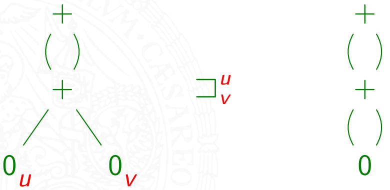
define for term graphs S, T

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

and position p

- $S \blacktriangleright_p T \;:\Longleftrightarrow\; S \sqsupset^u_v T$ for nodes $u, v \in S$ strictly below position p

Example



Restricted Folding and Unfolding

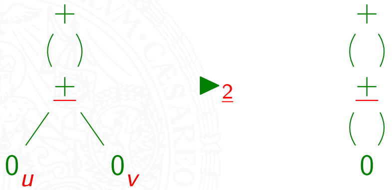
define for term graphs S, T

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

and position p

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

Example



Restricted Folding and Unfolding

define for term graphs S, T

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

and position p

- $S \blacktriangleright_p T :\Longleftrightarrow S \sqsupset^u_v T$ for nodes $u, v \in S$ strictly below position p
- $S \lhd_p T :\Longleftrightarrow S \sqsubset^u_v T$ and $u \in T$ node above position p

Restricted Folding and Unfolding

define for term graphs S, T

- $S \sqsupset_v^u T :\Longleftrightarrow$ "T obtained from S by identifying nodes u and v"

and position p

- $S \blacktriangleright_p T :\Longleftrightarrow S \sqsupset_v^u T$ for nodes $u, v \in S$ strictly below position p
- $S \lhd_p T :\Longleftrightarrow S \sqsubset_v^u T$ and $u \in T$ unshared node above position p

Restricted Folding and Unfolding

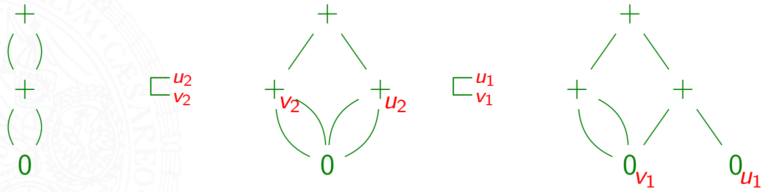
define for term graphs S, T

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

and position p

- $S \blacktriangleright_p T :\Longleftrightarrow S \sqsupset^u_v T$ for nodes $u, v \in S$ strictly below position p
- $S \lhd_p T :\Longleftrightarrow S \sqsubset^u_v T$ and $u \in T$ unshared node above position p

Example



Restricted Folding and Unfolding

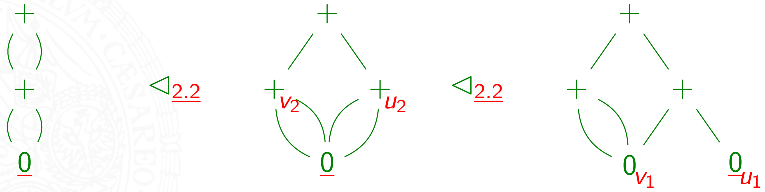
define for term graphs S, T

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

and position p

- $S \blacktriangleright_p T :\Longleftrightarrow S \sqsupset^u_v T$ for nodes $u, v \in S$ strictly below position p
- $S \lhd_p T :\Longleftrightarrow S \sqsubset^u_v T$ and $u \in T$ unshared node above position p

Example



Adequacy Theorem

"Complexity Version"

Lemma

1. *if* S *is* $\blacktriangleright_p$*-minimal then* $S \upharpoonright p$ *is maximally sharing*
2. *if* S *is* $\lhd_p$*-minimal then the node at position* p *is unshared*

Adequacy Theorem

"Complexity Version"

Lemma

1. *if S is $\blacktriangleright_p$-minimal then $S \upharpoonright p$ is maximally sharing*
2. *if S is $\lhd_p$-minimal then the node at position p is unshared*

Theorem

$$S \lhd_p^! \cdot \blacktriangleright_p^! \cdot \longrightarrow_{\mathcal{G},p} T \qquad \Longleftrightarrow \qquad \mathrm{term}(S) \to_{\mathcal{R},p} \mathrm{term}(T)$$

Adequacy Theorem

"Complexity Version"

Lemma

1. *if S is $\blacktriangleright_p$-minimal then $S \restriction p$ is maximally sharing*
2. *if S is $\lhd_p$-minimal then the node at position p is unshared*

Theorem

$$S \lhd_p^! \cdot \blacktriangleright_p^! \cdot \longrightarrow_{\mathcal{G},p} T \qquad \iff \qquad \mathrm{term}(S) \to_{\mathcal{R},p} \mathrm{term}(T)$$

- set $\twoheadrightarrow_{\mathcal{G},p} := \lhd_p^! \cdot \blacktriangleright_p^! \cdot \longrightarrow_{\mathcal{G},p}$

Adequacy Theorem

"Complexity Version"

Lemma

1. *if S is $\blacktriangleright_p$-minimal then $S \restriction p$ is maximally sharing*
2. *if S is $\lhd_p$-minimal then the node at position p is unshared*

Theorem

$$S \twoheadrightarrow_{\mathcal{G},p} T \qquad \iff \qquad \mathrm{term}(S) \rightarrow_{\mathcal{R},p} \mathrm{term}(T)$$

- set $\twoheadrightarrow_{\mathcal{G},p} := \lhd_p^! \cdot \blacktriangleright_p^! \cdot \longrightarrow_{\mathcal{G},p}$

Adequacy Theorem
"Complexity Version"

Lemma

1. *if S is $\blacktriangleright_p$-minimal then $S \restriction p$ is maximally sharing*
2. *if S is $\lhd_p$-minimal then the node at position p is unshared*

Theorem

$$S \twoheadrightarrow_{\mathcal{G},p} T \qquad \Longleftrightarrow \qquad \mathrm{term}(S) \to_{\mathcal{R},p} \mathrm{term}(T)$$

- set $\twoheadrightarrow_{\mathcal{G},p} := \lhd_p^! \cdot \blacktriangleright_p^! \cdot \longrightarrow_{\mathcal{G},p}$

Lemma

If $S \twoheadrightarrow_{\mathcal{G}}^{\ell} T$ then $|T| \leqslant (\ell + 1) \cdot |S| + \ell^2 \cdot \Delta$ for fixed $\Delta \in \mathbb{N}$

- polynomial size growth in $|S|$ and length ℓ

Main Result Revisited

Theorem

For any term s with $\mathsf{dl}_{\mathcal{R}}(s) \leqslant \ell$ $\qquad\qquad \ell = \Omega(|s|)$

1. *some normal-form is computable in deterministic time* $\mathrm{O}(\log(\ell)^3 \ell^7)$
2. *any normal-form is computable in nondeterministic time* $\mathrm{O}(\log(\ell)^2 \ell^5)$

Main Result Revisited

Theorem

For any term s with $\mathrm{dl}_{\mathcal{R}}(s) \leqslant \ell$ $\qquad\qquad$ $\ell = \Omega(|s|)$

1. *some normal-form is computable in deterministic time* $\mathrm{O}(\log(\ell)^3 \ell^7)$
2. *any normal-form is computable in nondeterministic time* $\mathrm{O}(\log(\ell)^2 \ell^5)$

Proof Idea.

$$s = s_0 \quad \to_{\mathcal{R}} \quad s_1 \quad \to_{\mathcal{R}} \quad \cdots \quad \to_{\mathcal{R}} \quad s_l$$

$\square$

Main Result Revisited

Theorem

For any term s with $\mathsf{dl}_{\mathcal{R}}(s) \leqslant \ell$ $\qquad\qquad \ell = \Omega(|s|)$

1. *some normal-form is computable in deterministic time* $\mathrm{O}(\log(\ell)^3 \ell^7)$
2. *any normal-form is computable in nondeterministic time* $\mathrm{O}(\log(\ell)^2 \ell^5)$

Proof Idea.

$$S = S_0 \quad \twoheadrightarrow_{\mathcal{G}} \quad S_1 \quad \twoheadrightarrow_{\mathcal{G}} \quad \cdots \quad \twoheadrightarrow_{\mathcal{G}} \quad S_l$$

1. rewrite graphs instead of terms $\qquad$ adequacy theorem

$\square$

Main Result Revisited

Theorem

For any term s with $\mathrm{dl}_{\mathcal{R}}(s) \leqslant \ell$ $\qquad\qquad$ $\ell = \Omega(|s|)$

1. *some normal-form is computable in deterministic time* $\mathrm{O}(\log(\ell)^3 \ell^7)$
2. *any normal-form is computable in nondeterministic time* $\mathrm{O}(\log(\ell)^2 \ell^5)$

Proof Idea.

$$S = S_0 \quad \twoheadrightarrow_{\mathcal{G}} \quad S_1 \quad \twoheadrightarrow_{\mathcal{G}} \quad \cdots \quad \twoheadrightarrow_{\mathcal{G}} \quad S_l$$

1. rewrite graphs instead of terms $\qquad$ adequacy theorem
2. size growth bound polynomially in ℓ and $|S|$ $\qquad$ restrictive unfolding

$\square$

Main Result Revisited

Theorem

For any term s with $\mathrm{dl}_{\mathcal{R}}(s) \leqslant \ell$ $\qquad \ell = \Omega(|s|) = \Omega(|S|)$

1. *some normal-form is computable in deterministic time* $\mathrm{O}(\log(\ell)^3 \ell^7)$
2. *any normal-form is computable in nondeterministic time* $\mathrm{O}(\log(\ell)^2 \ell^5)$

Proof Idea.

$$S = S_0 \quad \twoheadrightarrow_{\mathcal{G}} \quad S_1 \quad \twoheadrightarrow_{\mathcal{G}} \quad \cdots \quad \twoheadrightarrow_{\mathcal{G}} \quad S_l$$

1. rewrite graphs instead of terms — adequacy theorem
2. size growth bound polynomially in ℓ and $|S|$ — restrictive unfolding

$\square$

Main Result Revisited

Theorem

For any term s with $\mathrm{dl}_{\mathcal{R}}(s) \leqslant \ell$ $\qquad \ell = \Omega(|s|) = \Omega(|S|)$

1. *some normal-form is computable in deterministic time* $\mathrm{O}(\log(\ell)^3 \ell^7)$
2. *any normal-form is computable in nondeterministic time* $\mathrm{O}(\log(\ell)^2 \ell^5)$

Proof Idea.

$$S = S_0 \quad \twoheadrightarrow_{\mathcal{G}} \quad S_1 \quad \twoheadrightarrow_{\mathcal{G}} \quad \cdots \quad \twoheadrightarrow_{\mathcal{G}} \quad S_l$$

1. rewrite graphs instead of terms — adequacy theorem
2. size growth bound polynomially in ℓ — restrictive unfolding

□

Main Result Revisited

Theorem

For any term s with $\mathrm{dl}_{\mathcal{R}}(s) \leqslant \ell$ $\qquad \ell = \Omega(|s|) = \Omega(|S|)$

1. *some normal-form is computable in deterministic time* $\mathrm{O}(\log(\ell)^3 \ell^7)$
2. *any normal-form is computable in nondeterministic time* $\mathrm{O}(\log(\ell)^2 \ell^5)$

Proof Idea.

$$S = S_0 \twoheadrightarrow_{\mathcal{G}} S_1 \twoheadrightarrow_{\mathcal{G}} \cdots \twoheadrightarrow_{\mathcal{G}} S_l$$

1. rewrite graphs instead of terms — adequacy theorem
2. size growth bound polynomially in ℓ — restrictive unfolding
3. each step $S_i \twoheadrightarrow_{\mathcal{G}} S_{i+1}$ polytime computable in $|S_i|$ — tedious

$\square$

Main Result Revisited

Theorem

For any term s with $\mathrm{dl}_{\mathcal{R}}(s) \leqslant \ell$ $\qquad \ell = \Omega(|s|) = \Omega(|S|)$

1. *some normal-form is computable in deterministic time* $\mathrm{O}(\log(\ell)^3 \ell^7)$
2. *any normal-form is computable in nondeterministic time* $\mathrm{O}(\log(\ell)^2 \ell^5)$

Proof Idea.

$$S = S_0 \;\twoheadrightarrow_{\mathcal{G}}\; S_1 \;\twoheadrightarrow_{\mathcal{G}}\; \cdots \;\twoheadrightarrow_{\mathcal{G}}\; S_l$$

1. rewrite graphs instead of terms — adequacy theorem
2. size growth bound polynomially in ℓ — restrictive unfolding
3. each step $S_i \twoheadrightarrow_{\mathcal{G}} S_{i+1}$ polytime computable in $|S_i|$ hence ℓ — tedious

□

Main Result Revisited

Theorem

For any term s with $\mathsf{dl}_{\mathcal{R}}(s) \leqslant \ell$ $\qquad \ell = \Omega(|s|) = \Omega(|S|)$

1. *some normal-form is computable in deterministic time* $\mathrm{O}(\log(\ell)^3 \ell^7)$
2. *any normal-form is computable in nondeterministic time* $\mathrm{O}(\log(\ell)^2 \ell^5)$

Proof Idea.

$$S = S_0 \quad \twoheadrightarrow_{\mathcal{G}} \quad S_1 \quad \twoheadrightarrow_{\mathcal{G}} \quad \cdots \quad \twoheadrightarrow_{\mathcal{G}} \quad S_l$$

1. rewrite graphs instead of terms — adequacy theorem
2. size growth bound polynomially in ℓ — restrictive unfolding
3. each step $S_i \twoheadrightarrow_{\mathcal{G}} S_{i+1}$ polytime computable in $|S_i|$ hence ℓ — tedious
4. at most ℓ steps have to be performed — assumption

$\square$

Main Result Revisited

Theorem

For any term s *with* $\mathrm{dl}_{\mathcal{R}}(s) \leqslant \ell$ $\qquad \ell = \Omega(|s|) = \Omega(|S|)$

1. *some normal-form is computable in deterministic time* $\mathrm{O}(\log(\ell)^3 \ell^7)$
2. *any normal-form is computable in nondeterministic time* $\mathrm{O}(\log(\ell)^2 \ell^5)$

Main Result Revisited

Theorem

For any term s with $\mathsf{dl}_{\mathcal{R}}(s) \leqslant \ell$ $\qquad \ell = \Omega(|s|) = \Omega(|S|)$

1. *some normal-form is computable in deterministic time* $\mathrm{O}(\log(\ell)^3 \ell^7)$
2. *any normal-form is computable in nondeterministic time* $\mathrm{O}(\log(\ell)^2 \ell^5)$

Corollary

Let $\mathcal{R}$ *be a terminating TRS with* $\mathsf{rc}_{\mathcal{R}}(n) \in \mathrm{O}(n^k)$ $\qquad k \geqslant 1$

1. *if* $\mathcal{R}$ *computes the function* f *then* $f \in \mathrm{FP}$

Main Result Revisited

Theorem

For any term s with $\mathsf{dl}_{\mathcal{R}}(s) \leqslant \ell$ $\qquad \ell = \Omega(|s|) = \Omega(|S|)$

1. *some normal-form is computable in deterministic time* $\mathsf{O}(\log(\ell)^3 \ell^7)$
2. *any normal-form is computable in nondeterministic time* $\mathsf{O}(\log(\ell)^2 \ell^5)$

Corollary

Let $\mathcal{R}$ be a terminating TRS with $\mathsf{rc}_{\mathcal{R}}(n) \in \mathsf{O}(n^k)$ $\qquad k \geqslant 1$

1. *if $\mathcal{R}$ computes the function f then* $f \in \mathsf{FP}$
2. *if $\mathcal{R}$ computes the relation R then the function problem associated with R is in* FNP. $\qquad$ *given* $\vec{v} \in \mathcal{V}\mathsf{al}^k$, *find* $w \in \mathcal{V}\mathsf{al}$ *with* $(\vec{v}, w) \in R$

Main Result Revisited

Theorem

For any term s with $\mathsf{dl}_{\mathcal{R}}(s) \leqslant \ell$ $\qquad\qquad \ell = \Omega(|s|) = \Omega(|S|)$

1. *some normal-form is computable in deterministic time* $\mathsf{O}(\log(\ell)^3\ell^7)$
2. *any normal-form is computable in nondeterministic time* $\mathsf{O}(\log(\ell)^2\ell^5)$

Corollary

Let $\mathcal{R}$ be a terminating TRS with $\mathsf{rc}_{\mathcal{R}}(n) \in \mathsf{O}(n^k)$ $\qquad\qquad k \geqslant 1$

1. *if $\mathcal{R}$ computes the function f then* $f \in \mathsf{FP}$
2. *if $\mathcal{R}$ computes the relation R then the function problem associated with R is in* FNP. $\qquad$ *given* $\vec{v} \in \mathcal{V}\mathsf{al}^k$, *find* $w \in \mathcal{V}\mathsf{al}$ *with* $(\vec{v}, w) \in R$

FNP "is class of function problems associated with $\mathcal{L} \in \mathsf{NP}$"

$F_{\mathsf{SAT}} =$ given formula ϕ, find satisfying assignment α

Conclusion and Related Work

notion of runtime-complexity is a reasonable cost model for rewriting

1. cost of computation naturally expressed
2. polynomially related to actual cost on Turing machines

Conclusion and Related Work

notion of runtime-complexity is a reasonable cost model for rewriting

1. cost of computation naturally expressed
2. polynomially related to actual cost on Turing machines

U. Dal Lago and S. Martini
On Constructor Rewrite Systems and the Lambda-Calculus.
In ICALP, pages 163–174, 2009.

Results
polynomially bounded innermost runtime complexity induces polytime computability on orthogonal constructor TRSs

Conclusion and Related Work

notion of runtime-complexity is a reasonable cost model for rewriting

1. cost of computation naturally expressed
2. polynomially related to actual cost on Turing machines

U. Dal Lago and S. Martini
On Constructor Rewrite Systems and the Lambda-Calculus.
In ICALP, pages 163–174, 2009.

U. Dal Lago and S. Martini
Derivational Complexity is an Invariant Cost Model.
In FOPARA, 2009.

Results

polynomially bounded innermost (outermost) runtime complexity induces polytime computability on orthogonal TRSs

Conclusion and Related Work

notion of runtime-complexity is a reasonable cost model for rewriting

1. cost of computation naturally expressed
2. polynomially related to actual cost on Turing machines

following tools verify polynomial (i)RC fully automatically

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

Conclusion and Related Work

notion of runtime-complexity is a reasonable cost model for rewriting

1. cost of computation naturally expressed
2. polynomially related to actual cost on Turing machines

following tools verify polynomial (i)RC + CC fully automatically

- $\mathsf{T_CT}$
  http://cl-informatik.uibk.ac.at/research/software/tct

Conclusion and Related Work

notion of runtime-complexity is a reasonable cost model for rewriting

1. cost of computation naturally expressed
2. polynomially related to actual cost on Turing machines

following tools verify polynomial (i)RC + CC fully automatically

- AProVE — innermost runtime complexity
  http://aprove.informatik.rwth-aachen.de/
- CaT
  http://cl-informatik.uibk.ac.at/research/software/ttt2
- T_CT
  http://cl-informatik.uibk.ac.at/research/software/tct
- Matchbox/Poly — derivational complexity
  http://dfa.imn.htwk-leipzig.de/matchbox/poly