Complexity Analysis by Graph Rewriting

Martin Avanzini and Georg Moser

Computational Logic
Faculty of Computer Science, University of Innsbruck

FLOPS 2010

Term Rewriting

First Order 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	$e_1 - e_2$

Term Rewriting

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

Underlying Computation

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

Term Rewriting

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

Underlying Computation

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

Term Rewriting

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

Underlying Computation

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

Term Rewriting

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

Underlying Computation

$$\begin{aligned}\mathsf{d}(\mathsf{c} + (\mathsf{c} \times \mathsf{c})) &= \mathsf{d}(\mathsf{c}) + \mathsf{d}(\mathsf{c} \times \mathsf{c}) \\ &= 0 + \mathsf{d}(\mathsf{c} \times \mathsf{c})\end{aligned}$$

Term Rewriting

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

Underlying Computation

$$
\begin{aligned}
\mathsf{d}(\mathsf{c} + (\mathsf{c} \times \mathsf{c})) &= \mathsf{d}(\mathsf{c}) + \mathsf{d}(\mathsf{c} \times \mathsf{c}) \\
&= 0 + \mathsf{d}(\mathsf{c} \times \mathsf{c}) \\
&= 0 + \mathsf{d}(\mathsf{c}) \times \mathsf{c} + \mathsf{c} \times \mathsf{d}(\mathsf{c})
\end{aligned}
$$

Term Rewriting

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

Underlying Computation

$$
\begin{aligned}
\mathsf{d}(\mathsf{c} + (\mathsf{c} \times \mathsf{c})) &= \mathsf{d}(\mathsf{c}) + \mathsf{d}(\mathsf{c} \times \mathsf{c}) \\
&= 0 + \mathsf{d}(\mathsf{c} \times \mathsf{c}) \\
&= 0 + \mathsf{d}(\mathsf{c}) \times \mathsf{c} + \mathsf{c} \times \mathsf{d}(\mathsf{c}) \\
&= 0 + 0 \times \mathsf{c} + \mathsf{c} \times \mathsf{d}(\mathsf{c})
\end{aligned}
$$

Term Rewriting

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

Underlying Computation

$$
\begin{aligned}
\mathsf{d}(\mathsf{c} + (\mathsf{c} \times \mathsf{c})) &= \mathsf{d}(\mathsf{c}) + \mathsf{d}(\mathsf{c} \times \mathsf{c}) \\
&= 0 + \mathsf{d}(\mathsf{c} \times \mathsf{c}) \\
&= 0 + \mathsf{d}(\mathsf{c}) \times \mathsf{c} + \mathsf{c} \times \mathsf{d}(\mathsf{c}) \\
&= 0 + 0 \times \mathsf{c} + \mathsf{c} \times \mathsf{d}(\mathsf{c}) \\
&= 0 + 0 \times \mathsf{c} + \mathsf{c} \times 0
\end{aligned}
$$

Term Rewriting

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

Underlying Computation

$$
\begin{aligned}
\mathsf{d}(\mathsf{c} + (\mathsf{c} \times \mathsf{c})) &= \mathsf{d}(\mathsf{c}) + \mathsf{d}(\mathsf{c} \times \mathsf{c}) \\
&= 0 + \mathsf{d}(\mathsf{c} \times \mathsf{c}) \\
&= 0 + \mathsf{d}(\mathsf{c}) \times \mathsf{c} + \mathsf{c} \times \mathsf{d}(\mathsf{c}) \\
&= 0 + 0 \times \mathsf{c} + \mathsf{c} \times \mathsf{d}(\mathsf{c}) \\
&= 0 + 0 \times \mathsf{c} + \mathsf{c} \times 0
\end{aligned}
$$

Term Rewriting

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

Underlying Computation $\approx$ Rewriting

$$\begin{aligned}
\mathsf{d}(\mathsf{c} + (\mathsf{c} \times \mathsf{c})) &\to_{\mathcal{R}} \mathsf{d}(\mathsf{c}) + \mathsf{d}(\mathsf{c} \times \mathsf{c}) \\
&\to_{\mathcal{R}} 0 + \mathsf{d}(\mathsf{c} \times \mathsf{c}) \\
&\to_{\mathcal{R}} 0 + \mathsf{d}(\mathsf{c}) \times \mathsf{c} + \mathsf{c} \times \mathsf{d}(\mathsf{c}) \\
&\to_{\mathcal{R}} 0 + 0 \times \mathsf{c} + \mathsf{c} \times \mathsf{d}(\mathsf{c}) \\
&\to_{\mathcal{R}} 0 + 0 \times \mathsf{c} + \mathsf{c} \times 0
\end{aligned}$$

Term Rewriting

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

Underlying Computation $\approx$ Rewriting

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

Term Rewriting

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

Underlying Computation $\approx$ Rewriting

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

- above TRS computes differentiation of arithmetical expressions

Term Rewriting

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

Underlying Computation $\approx$ Rewriting

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

- above TRS computes differentiation of arithmetical expressions

Runtime Complexity

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

- innermost runtime complexity
  number of eager evaluation steps as function in the size of the initial terms

- innermost runtime complexity
  number of eager evaluation steps as function in the size of the initial terms

$$\mathsf{rc}^{\mathrm{i}}_{\mathcal{R}}(n) = \max\{\mathsf{dl}(t, \xrightarrow{\mathrm{i}}_{\mathcal{R}}) \mid \mathsf{size}(t) \leqslant n \}$$

  - $\xrightarrow{\mathrm{i}}_{\mathcal{R}} \subseteq \rightarrow_{\mathcal{R}}$ is restriction to eager evaluation

- derivation length

$$\mathsf{dl}(t, \xrightarrow{\mathrm{i}}_{\mathcal{R}}) = \max\{\ell \mid \exists(t_1, \dots, t_\ell).\ t \xrightarrow{\mathrm{i}}_{\mathcal{R}} t_1 \xrightarrow{\mathrm{i}}_{\mathcal{R}} \dots \xrightarrow{\mathrm{i}}_{\mathcal{R}} t_\ell\}$$

- innermost runtime complexity
  number of eager evaluation steps as function in the size of the initial terms

$$\mathsf{rc}^{\mathrm{i}}_{\mathcal{R}}(n) = \max\{\mathsf{dl}(t, \xrightarrow{\mathrm{i}}_{\mathcal{R}}) \mid \mathsf{size}(t) \leqslant n \text{ and arguments values}\}$$

  - $\xrightarrow{\mathrm{i}}_{\mathcal{R}} \subseteq \rightarrow_{\mathcal{R}}$ is restriction to eager evaluation
  - measure complexity of direct function calls

- derivation length

$$\mathsf{dl}(t, \xrightarrow{\mathrm{i}}_{\mathcal{R}}) = \max\{\ell \mid \exists (t_1, \dots, t_\ell).\ t \xrightarrow{\mathrm{i}}_{\mathcal{R}} t_1 \xrightarrow{\mathrm{i}}_{\mathcal{R}} \dots \xrightarrow{\mathrm{i}}_{\mathcal{R}} t_\ell\}$$

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

- runtime complexity of above TRS is linear

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

- runtime complexity of above TRS is linear
- this can be automatically verified

```
$ tct -a rc -p -s "wdp (matrix :kind triangular)" 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 $\mathsf{d}(x - y) \to \mathsf{d}(x) - \mathsf{d}(y)$

- runtime complexity of above TRS is linear
- this can be automatically verified

```
$ tct -a rc -p -s "wdp (matrix :kind trian
  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:
   ...
```

is this proof really meaningful?

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

- runtime complexity of above TRS is linear
- this can be automatically verified

```
$ tct -a rc -p -s "wdp (matrix :kind trian
  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:
   ...
```

is this proof really
a certificate for
polytime computability?

Main Result

Yes

Main Result

Theorem

If the (innermost) runtime-complexity of $\mathcal{R}$ is polynomially bounded, then each function f computed by $\mathcal{R}$ is polytime computable.

$$\mathsf{rc}^{\mathrm{i}}_{\mathcal{R}}(n) \leqslant n^k \;\Rightarrow\; f \in \mathsf{TIME}(\mathsf{O}(n^{5\cdot(k+1)})) \qquad f \text{ computed by } \mathcal{R}$$

Main Result

Theorem

If the (innermost) runtime-complexity of $\mathcal{R}$ is polynomially bounded, then each function f computed by $\mathcal{R}$ is polytime computable.

$$\mathsf{rc}^{\mathrm{i}}_{\mathcal{R}}(n) \leqslant n^k \Rightarrow f \in \mathsf{FP}$$

f computed by $\mathcal{R}$

Main Result

Theorem

If the (innermost) runtime-complexity of $\mathcal{R}$ is polynomially bounded, then each function f computed by $\mathcal{R}$ is polytime computable.

$\mathsf{rc}^{\mathrm{i}}_{\mathcal{R}}(n) \leqslant n^k \;\Rightarrow\; f \in \mathsf{FP}$ *f computed by $\mathcal{R}$*

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

1. polynomial runtime complexity can be automatically verified
2. polytime computability of above given function can be verified automatically

Main Result

Theorem

If the (innermost) runtime-complexity of $\mathcal{R}$ is polynomially bounded, then each function f computed by $\mathcal{R}$ is polytime computable.

$\mathsf{rc}^{\mathrm{i}}_{\mathcal{R}}(n) \leqslant n^k \;\Rightarrow\; f \in \mathsf{FP}$ *f computed by $\mathcal{R}$*

Proof Idea

- *implement rewriting efficiently*

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

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

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

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

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

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

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

Main Result
Difficulty

a single rewrite step may copy arbitrarily large terms

☞ terms may grow exponential in the length of derivations

Example

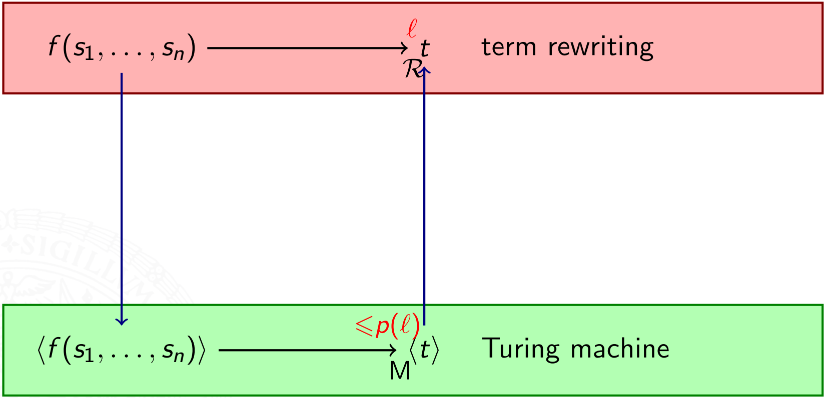
① $\mathsf{d}(\mathsf{c}) \to 0$

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

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

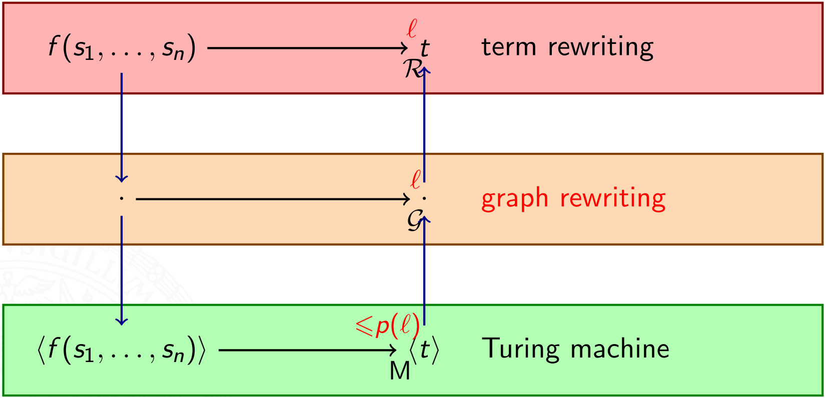
④ $\mathsf{d}(x - y) \to \mathsf{d}(x) - \mathsf{d}(y)$

$$
\begin{aligned}
\mathsf{d}(\mathsf{c}) &= 0 \\
\mathsf{d}(\mathsf{c} \times \mathsf{c}) &= 0 \times \mathsf{c} + \mathsf{c} \times 0 \\
\mathsf{d}((\mathsf{c} \times \mathsf{c}) \times \mathsf{c}) &= (0 \times \mathsf{c} + \mathsf{c} \times 0) \times \mathsf{c} + (\mathsf{c} \times \mathsf{c}) \times 0 \\
\mathsf{d}((\mathsf{c} \times \mathsf{c}) \times (\mathsf{c} \times \mathsf{c})) &= ((0 \times \mathsf{c} + \mathsf{c} \times 0) \times \mathsf{c} + (\mathsf{c} \times \mathsf{c}) \times 0) \times (\mathsf{c} \times \mathsf{c}) \\
&\quad + (\mathsf{c} \times \mathsf{c}) \times ((0 \times \mathsf{c} + \mathsf{c} \times 0) \times \mathsf{c} + (\mathsf{c} \times \mathsf{c}) \times 0)
\end{aligned}
$$


$f(s_1, \ldots, s_n)$
ℓ
$\mathcal{R}$
t
term rewriting
$\langle f(s_1, \ldots, s_n) \rangle$
$\leqslant p(\ell)$
M
$\langle t \rangle$
Turing machine

Main Result

Proof Outline



Outline

- Graph Rewriting in a Nutshell
- Adequacy of Graph Rewriting
- Conclusion

Graph Rewriting in a Nutshell

Graph Rewriting in a Nutshell

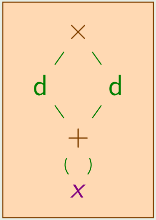
- term rewriting on graphs
- copying $\rightsquigarrow$ sharing
- structural equality $\rightsquigarrow$ pointer equality

Graph Rewriting in a Nutshell

- term rewriting on graphs
- copying $\rightsquigarrow$ sharing
- structural equality $\rightsquigarrow$ pointer equality

Example

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



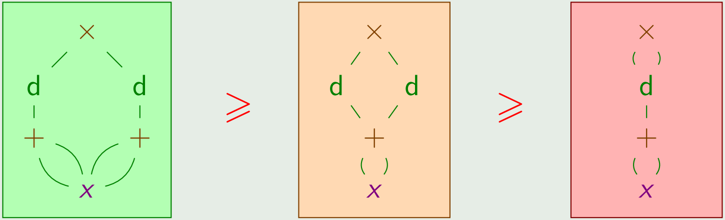
- variables always represented by unique node

Graph Rewriting in a Nutshell

- term rewriting on graphs
- copying ⇝ sharing
- structural equality ⇝ pointer equality

Example

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



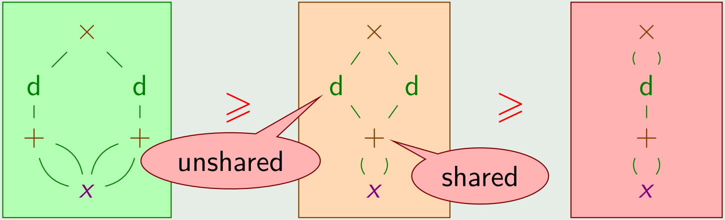
- variables always represented by unique node

Graph Rewriting in a Nutshell

- term rewriting on graphs
- copying $\rightsquigarrow$ sharing
- structural equality $\rightsquigarrow$ pointer equality

Example

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



- variables always represented by unique node

Graph Rewriting in a Nutshell

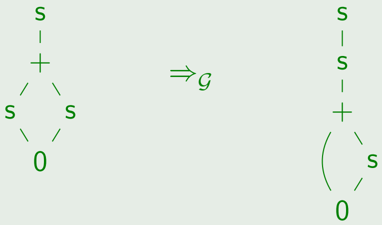
Example

applying rule $\mathsf{s}(x) + y \to \mathsf{s}(x + y)$ on $\mathsf{s}(\mathsf{s}(0) + \mathsf{s}(0))$ . . .

Term Rewriting

$$\mathsf{s}(\mathsf{s}(0) + \mathsf{s}(0)) \quad \to_{\mathcal{R}} \quad \mathsf{s}(\mathsf{s}(0 + \mathsf{s}(0)))$$

Graph Rewriting



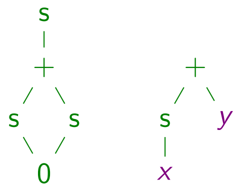
Graph Rewriting in a Nutshell

Rewriting $\mathsf{s}(\mathsf{s}(0) + \mathsf{s}(0))$ using rule $\mathsf{s}(x) + y \to \mathsf{s}(x + y)$

term rewriting	graph rewriting
1. identifying matching subterm $\mathsf{s}(\mathsf{s}(0) + \mathsf{s}(0))\vert_1 = \sigma(\mathsf{s}(x) + y)$ $\sigma = \begin{cases} x \mapsto 0 \\ y \mapsto \mathsf{s}(0) \end{cases}$	

Graph Rewriting in a Nutshell

Rewriting $\mathsf{s}(\mathsf{s}(0) + \mathsf{s}(0))$ using rule $\mathsf{s}(x) + y \to \mathsf{s}(x + y)$

term rewriting	graph rewriting
1. identifying matching subterm $\mathsf{s}(\mathsf{s}(0) + \mathsf{s}(0))\|_1 = \sigma(\mathsf{s}(x) + y)$ $\sigma = \begin{cases} x \mapsto 0 \\ y \mapsto \mathsf{s}(0) \end{cases}$	1. finding term graph morphism 

Graph Rewriting in a Nutshell

Rewriting $\mathsf{s}(\mathsf{s}(0)+\mathsf{s}(0))$ using rule $\mathsf{s}(x)+y \to \mathsf{s}(x+y)$

term rewriting

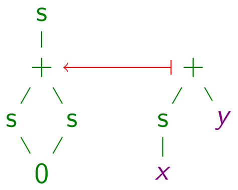
1. identifying matching subterm

$$\mathsf{s}(\mathsf{s}(0)+\mathsf{s}(0))|_1 = \sigma(\mathsf{s}(x)+y)$$

$$\sigma = \begin{cases} x \mapsto 0 \\ y \mapsto \mathsf{s}(0) \end{cases}$$

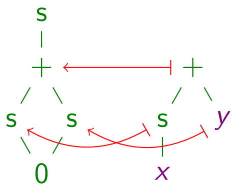
graph rewriting

1. finding term graph morphism



Graph Rewriting in a Nutshell

Rewriting $\mathsf{s}(\mathsf{s}(0)+\mathsf{s}(0))$ using rule $\mathsf{s}(x)+y \to \mathsf{s}(x+y)$

term rewriting	graph rewriting
1. identifying matching subterm $\mathsf{s}(\mathsf{s}(0)+\mathsf{s}(0))\vert_1 = \sigma(\mathsf{s}(x)+y)$ $\sigma = \begin{cases} x \mapsto 0 \\ y \mapsto \mathsf{s}(0) \end{cases}$	1. finding term graph morphism 

Graph Rewriting in a Nutshell

Rewriting $\mathsf{s}(\mathsf{s}(0)+\mathsf{s}(0))$ using rule $\mathsf{s}(x)+y \to \mathsf{s}(x+y)$

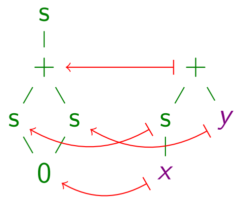
term rewriting	graph rewriting
1. identifying matching subterm	1. finding term graph morphism

Term rewriting:

$$\mathsf{s}(\mathsf{s}(0)+\mathsf{s}(0))|_1 = \sigma(\mathsf{s}(x)+y)$$

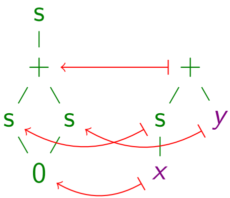
$$\sigma = \begin{cases} x \mapsto 0 \\ y \mapsto \mathsf{s}(0) \end{cases}$$

Graph rewriting:



Graph Rewriting in a Nutshell

Rewriting $\mathsf{s}(\mathsf{s}(0) + \mathsf{s}(0))$ using rule $\mathsf{s}(x) + y \to \mathsf{s}(x + y)$

term rewriting	graph rewriting
1. identifying matching subterm $\mathsf{s}(\mathsf{s}(0) + \mathsf{s}(0))\|_1 = \sigma(\mathsf{s}(x) + y)$ $\sigma = \begin{cases} x \mapsto 0 \\ y \mapsto \mathsf{s}(0) \end{cases}$	1. finding term graph morphism 
2. replace matched subterm $\mathsf{s}(\mathsf{s}(\mathsf{s}(0) + 0))$	

Graph Rewriting in a Nutshell

Rewriting $\mathsf{s}(\mathsf{s}(0)+\mathsf{s}(0))$ using rule $\mathsf{s}(x)+y \to \mathsf{s}(x+y)$

term rewriting

1. identifying matching subterm

$$\mathsf{s}(\mathsf{s}(0)+\mathsf{s}(0))|_1 = \sigma(\mathsf{s}(x)+y)$$

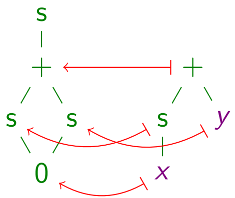
$$\sigma = \begin{cases} x \mapsto 0 \\ y \mapsto \mathsf{s}(0) \end{cases}$$

2. replace matched subterm

$$\mathsf{s}(\sigma(\mathsf{s}(x)+y))$$

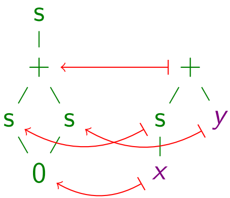
graph rewriting

1. finding term graph morphism



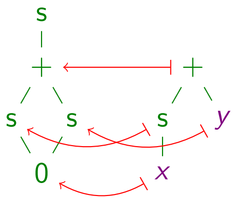
Graph Rewriting in a Nutshell

Rewriting $\mathsf{s}(\mathsf{s}(0) + \mathsf{s}(0))$ using rule $\mathsf{s}(x) + y \to \mathsf{s}(x + y)$

term rewriting	graph rewriting
1. identifying matching subterm $\mathsf{s}(\mathsf{s}(0) + \mathsf{s}(0))\|_1 = \sigma(\mathsf{s}(x) + y)$ $\sigma = \begin{cases} x \mapsto 0 \\ y \mapsto \mathsf{s}(0) \end{cases}$	1. finding term graph morphism 
2. replace matched subterm $\mathsf{s}(\sigma(\mathsf{s}(x) + y)) \to_{\mathcal{R}} \mathsf{s}(\sigma(\mathsf{s}(x + y)))$	

Graph Rewriting in a Nutshell

Rewriting $\mathsf{s}(\mathsf{s}(0) + \mathsf{s}(0))$ using rule $\mathsf{s}(x) + y \to \mathsf{s}(x + y)$

term rewriting	graph rewriting
1. identifying matching subterm $\mathsf{s}(\mathsf{s}(0) + \mathsf{s}(0))\|_1 = \sigma(\mathsf{s}(x) + y)$ $\sigma = \begin{cases} x \mapsto 0 \\ y \mapsto \mathsf{s}(0) \end{cases}$	1. finding term graph morphism 
2. replace matched subterm $\mathsf{s}(\sigma(\mathsf{s}(x) + y)) \to_{\mathcal{R}} \mathsf{s}(\mathsf{s}(0 + \mathsf{s}(0)))$	

Graph Rewriting in a Nutshell

Rewriting $\mathsf{s}(\mathsf{s}(0) + \mathsf{s}(0))$ using rule $\mathsf{s}(x) + y \to \mathsf{s}(x + y)$

term rewriting	graph rewriting
1. identifying matching subterm	1. finding term graph morphism
2. replace matched subterm	2. replace matched subgraph

term rewriting

1. identifying matching subterm

$$\mathsf{s}(\mathsf{s}(0) + \mathsf{s}(0))|_1 = \sigma(\mathsf{s}(x) + y)$$

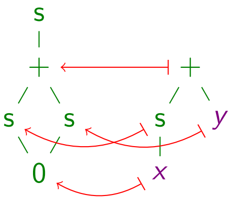
$$\sigma = \begin{cases} x \mapsto 0 \\ y \mapsto \mathsf{s}(0) \end{cases}$$

2. replace matched subterm

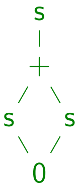
$$\mathsf{s}(\sigma(\mathsf{s}(x) + y)) \to_{\mathcal{R}} \mathsf{s}(\mathsf{s}(0 + \mathsf{s}(0)))$$

graph rewriting

1. finding term graph morphism

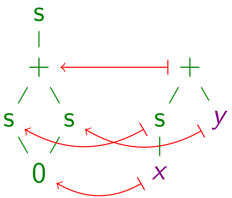
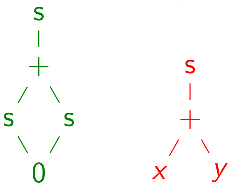


2. replace matched subgraph



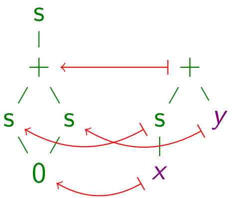
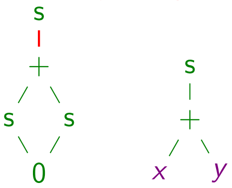
Graph Rewriting in a Nutshell

Rewriting $\mathsf{s}(\mathsf{s}(0)+\mathsf{s}(0))$ using rule $\mathsf{s}(x)+y \to \mathsf{s}(x+y)$

term rewriting	graph rewriting
1. identifying matching subterm $\mathsf{s}(\mathsf{s}(0)+\mathsf{s}(0))\|_1 = \sigma(\mathsf{s}(x)+y)$ $\sigma = \begin{cases} x \mapsto 0 \\ y \mapsto \mathsf{s}(0) \end{cases}$	1. finding term graph morphism 
2. replace matched subterm $\mathsf{s}(\sigma(\mathsf{s}(x)+y)) \to_{\mathcal{R}} \mathsf{s}(\mathsf{s}(0+\mathsf{s}(0)))$	2a. add copy of right-hand side 

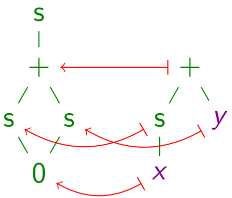
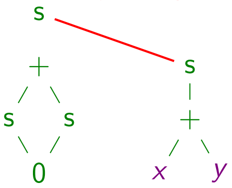
Graph Rewriting in a Nutshell

Rewriting $\mathsf{s}(\mathsf{s}(0) + \mathsf{s}(0))$ using rule $\mathsf{s}(x) + y \to \mathsf{s}(x + y)$

term rewriting	graph rewriting
1. identifying matching subterm $\mathsf{s}(\mathsf{s}(0) + \mathsf{s}(0))\|_1 = \sigma(\mathsf{s}(x) + y)$ $\sigma = \begin{cases} x \mapsto 0 \\ y \mapsto \mathsf{s}(0) \end{cases}$	1. finding term graph morphism 
2. replace matched subterm $\mathsf{s}(\sigma(\mathsf{s}(x) + y)) \to_{\mathcal{R}} \mathsf{s}(\mathsf{s}(0 + \mathsf{s}(0)))$	2a. add copy of right-hand side 

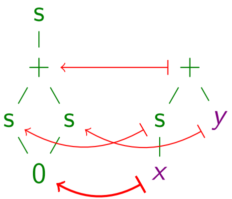
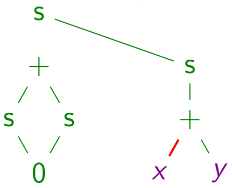
Graph Rewriting in a Nutshell

Rewriting $\mathsf{s}(\mathsf{s}(0) + \mathsf{s}(0))$ using rule $\mathsf{s}(x) + y \to \mathsf{s}(x + y)$

term rewriting	graph rewriting
1. identifying matching subterm	1. finding term graph morphism
$\mathsf{s}(\mathsf{s}(0) + \mathsf{s}(0))\vert_1 = \sigma(\mathsf{s}(x) + y)$ $\sigma = \begin{cases} x \mapsto 0 \\ y \mapsto \mathsf{s}(0) \end{cases}$	
2. replace matched subterm	2a. add copy of right-hand side
$\mathsf{s}(\sigma(\mathsf{s}(x) + y)) \to_{\mathcal{R}} \mathsf{s}(\mathsf{s}(0 + \mathsf{s}(0)))$	

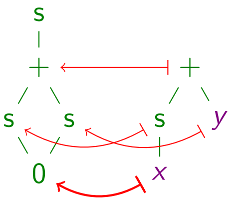
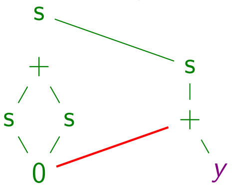
Graph Rewriting in a Nutshell

Rewriting $\mathsf{s}(\mathsf{s}(0) + \mathsf{s}(0))$ using rule $\mathsf{s}(x) + y \to \mathsf{s}(x + y)$

term rewriting	graph rewriting
1. identifying matching subterm $\mathsf{s}(\mathsf{s}(0) + \mathsf{s}(0))\|_1 = \sigma(\mathsf{s}(x) + y)$ $\sigma = \begin{cases} x \mapsto 0 \\ y \mapsto \mathsf{s}(0) \end{cases}$	1. finding term graph morphism 
2. replace matched subterm $\mathsf{s}(\sigma(\mathsf{s}(x) + y)) \to_{\mathcal{R}} \mathsf{s}(\mathsf{s}(0 + \mathsf{s}(0)))$	2b. redirect edges 

Graph Rewriting in a Nutshell

Rewriting $\mathsf{s}(\mathsf{s}(0) + \mathsf{s}(0))$ using rule $\mathsf{s}(x) + y \to \mathsf{s}(x + y)$

term rewriting	graph rewriting
1. identifying matching subterm	1. finding term graph morphism
$\mathsf{s}(\mathsf{s}(0) + \mathsf{s}(0))\vert_1 = \sigma(\mathsf{s}(x) + y)$ $\sigma = \begin{cases} x \mapsto 0 \\ y \mapsto \mathsf{s}(0) \end{cases}$	
2. replace matched subterm	2b. redirect edges
$\mathsf{s}(\sigma(\mathsf{s}(x) + y)) \to_{\mathcal{R}} \mathsf{s}(\mathsf{s}(0 + \mathsf{s}(0)))$	

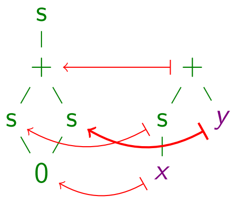
Graph Rewriting in a Nutshell

Rewriting $\mathsf{s}(\mathsf{s}(0) + \mathsf{s}(0))$ using rule $\mathsf{s}(x) + y \to \mathsf{s}(x + y)$

term rewriting	graph rewriting
1. identifying matching subterm	1. finding term graph morphism

$$\mathsf{s}(\mathsf{s}(0) + \mathsf{s}(0))|_1 = \sigma(\mathsf{s}(x) + y)$$

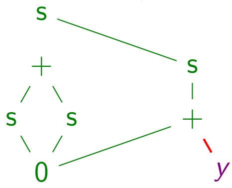
$$\sigma = \begin{cases} x \mapsto 0 \\ y \mapsto \mathsf{s}(0) \end{cases}$$



2. replace matched subterm

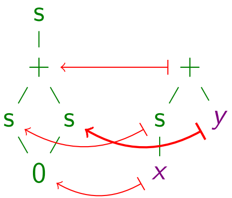
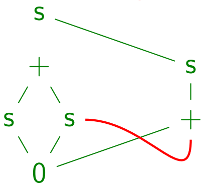
$$\mathsf{s}(\sigma(\mathsf{s}(x) + y)) \to_{\mathcal{R}} \mathsf{s}(\mathsf{s}(0 + \mathsf{s}(0)))$$

2b. redirect edges



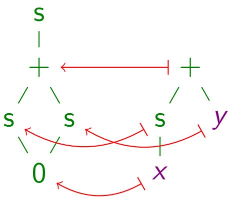
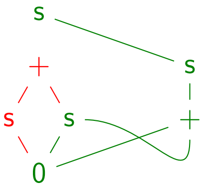
Graph Rewriting in a Nutshell

Rewriting $\mathsf{s}(\mathsf{s}(0) + \mathsf{s}(0))$ using rule $\mathsf{s}(x) + y \to \mathsf{s}(x + y)$

term rewriting	graph rewriting
1. identifying matching subterm	1. finding term graph morphism
$\mathsf{s}(\mathsf{s}(0) + \mathsf{s}(0))\|_1 = \sigma(\mathsf{s}(x) + y)$ $\sigma = \begin{cases} x \mapsto 0 \\ y \mapsto \mathsf{s}(0) \end{cases}$	
2. replace matched subterm	2b. redirect edges
$\mathsf{s}(\sigma(\mathsf{s}(x) + y)) \to_{\mathcal{R}} \mathsf{s}(\mathsf{s}(0 + \mathsf{s}(0)))$	

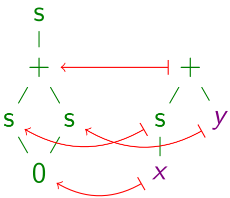
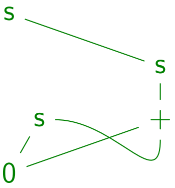
Graph Rewriting in a Nutshell

Rewriting $\mathsf{s}(\mathsf{s}(0) + \mathsf{s}(0))$ using rule $\mathsf{s}(x) + y \to \mathsf{s}(x + y)$

term rewriting	graph rewriting
1. identifying matching subterm $\mathsf{s}(\mathsf{s}(0) + \mathsf{s}(0))\vert_1 = \sigma(\mathsf{s}(x) + y)$ $\sigma = \begin{cases} x \mapsto 0 \\ y \mapsto \mathsf{s}(0) \end{cases}$	1. finding term graph morphism 
2. replace matched subterm $\mathsf{s}(\sigma(\mathsf{s}(x) + y)) \to_{\mathcal{R}} \mathsf{s}(\mathsf{s}(0 + \mathsf{s}(0)))$	2c. remove inaccessible nodes 

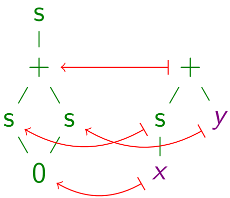
Graph Rewriting in a Nutshell

Rewriting $\mathsf{s}(\mathsf{s}(0)+\mathsf{s}(0))$ using rule $\mathsf{s}(x)+y \to \mathsf{s}(x+y)$

term rewriting	graph rewriting
1. identifying matching subterm	1. finding term graph morphism
$\mathsf{s}(\mathsf{s}(0)+\mathsf{s}(0))\|_1 = \sigma(\mathsf{s}(x)+y)$ $\sigma = \begin{cases} x \mapsto 0 \\ y \mapsto \mathsf{s}(0) \end{cases}$	
2. replace matched subterm	2c. remove inaccessible nodes
$\mathsf{s}(\sigma(\mathsf{s}(x)+y)) \to_{\mathcal{R}} \mathsf{s}(\mathsf{s}(0+\mathsf{s}(0)))$	

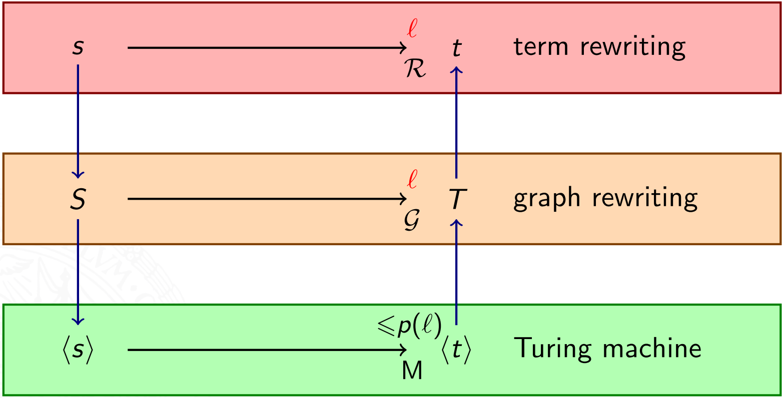
Graph Rewriting in a Nutshell

Rewriting $\mathsf{s}(\mathsf{s}(0) + \mathsf{s}(0))$ using rule $\mathsf{s}(x) + y \to \mathsf{s}(x + y)$

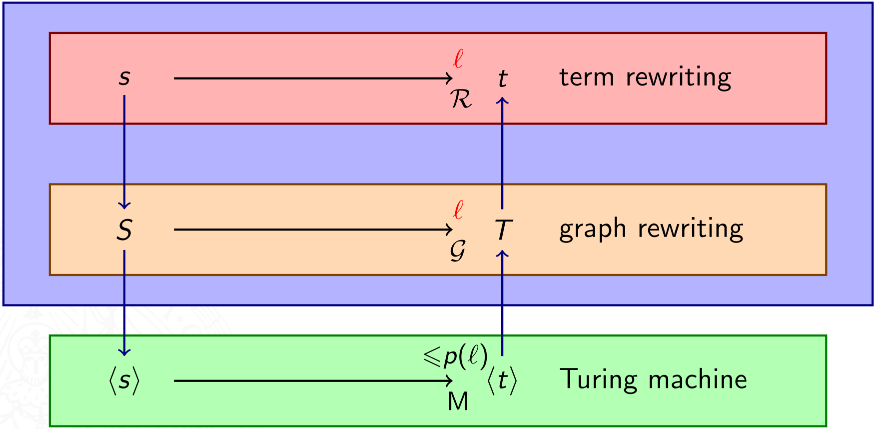
term rewriting	graph rewriting
1. identifying matching subterm	1. finding term graph morphism
$\mathsf{s}(\mathsf{s}(0) + \mathsf{s}(0))\|_1 = \sigma(\mathsf{s}(x) + y)$ $\sigma = \begin{cases} x \mapsto 0 \\ y \mapsto \mathsf{s}(0) \end{cases}$	
2. replace matched subterm	2c. remove inaccessible nodes
$\mathsf{s}(\sigma(\mathsf{s}(x) + y)) \to_{\mathcal{R}} \mathsf{s}(\mathsf{s}(0 + \mathsf{s}(0)))$	

Adequacy of Graph Rewriting

Adequacy of Graph Rewriting for Term Rewriting



Adequacy of Graph Rewriting for Term Rewriting



Simulating Graph Rewrite System

Definition

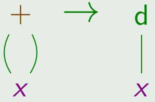
- simulating graph rewrite system $\mathcal{G}(\mathcal{R})$ of TRS $\mathcal{R}$

$$\mathcal{G}(\mathcal{R}) := \{\triangle(l) \to \triangle(r) \mid (l \to r) \in \mathcal{R}\}$$

- $\triangle(s)$ is minimally sharing graph representing s

Example

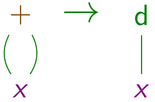
$x + x \to \mathsf{d}(x)$ $\Longrightarrow$



Simulating Graph Rewrite System

Problems

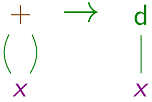
$x + x \to \mathsf{d}(x)$ $\quad\Longrightarrow\quad$



Simulating Graph Rewrite System

Problems

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



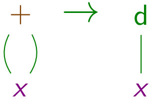
$$((0+0)+(0+0)) \times ((0+0)+(0+0))$$

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

Simulating Graph Rewrite System

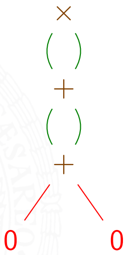
Problems

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



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

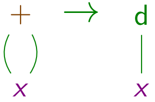
$$\to_{\mathcal{R}} \big((0+0)+(0+0)\big) \times \big((0+0)+\mathsf{d}(0)\big)$$



Simulating Graph Rewrite System

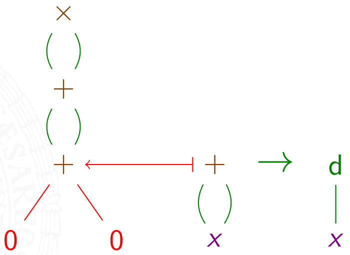
Problems

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



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

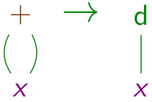
$$\to_{\mathcal{R}} \big((0+0)+(0+0)\big) \times \big((0+0)+\mathsf{d}(0)\big)$$



Simulating Graph Rewrite System

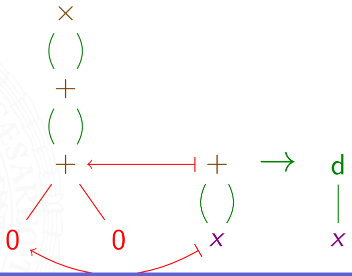
Problems

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



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

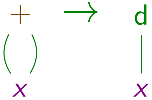
$$\to_{\mathcal{R}} ((0+0)+(0+0)) \times ((0+0)+\mathsf{d}(0))$$



Simulating Graph Rewrite System

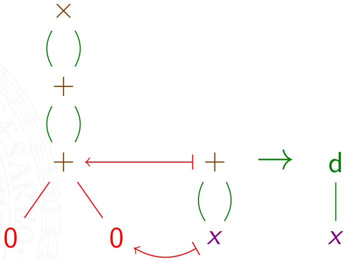
Problems

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



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

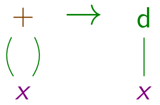
$$\to_{\mathcal{R}} \big((0+0)+(0+0)\big) \times \big((0+0)+\mathsf{d}(0)\big)$$



Simulating Graph Rewrite System

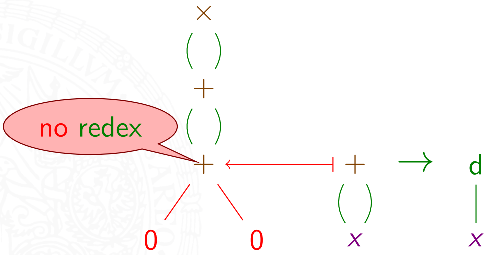
Problems

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



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

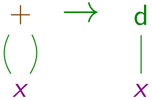
$$\to_{\mathcal{R}} \big((0 + 0) + (0 + 0)\big) \times \big((0 + 0) + \mathsf{d}(0)\big)$$



Simulating Graph Rewrite System

Problems

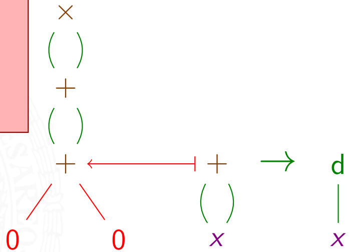
$x + x \to \mathsf{d}(x) \quad \Rightarrow$



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

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

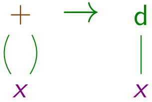
Problem ①
below redex
maximal sharing
required



Simulating Graph Rewrite System

Problems

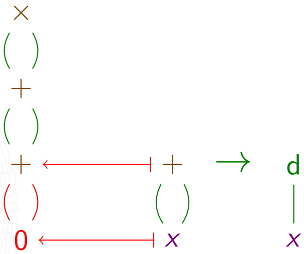
$$x + x \to \mathsf{d}(x) \quad \Rightarrow$$



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

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

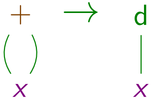
Problem ①
below redex
maximal sharing
required



Simulating Graph Rewrite System

Problems

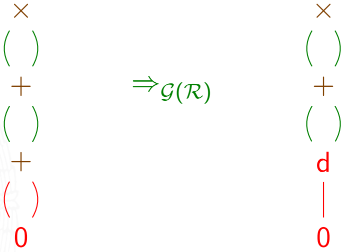
$x + x \to \mathsf{d}(x)$ $\Rrightarrow$



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

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

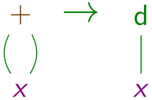
Problem ①
below redex
maximal sharing
required



Simulating Graph Rewrite System

Problems

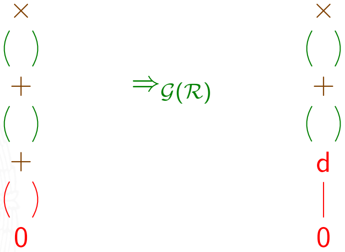
$x + x \to \mathsf{d}(x)$ $\Rightarrow$



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

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

Problem ①
below redex
maximal sharing
required



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

Adequacy of Graph Rewriting

Theorem

suppose s *is a term and* S *is a term graph representing* s *such that for redex position* p *in* s

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

Then

$$s \to_{\mathcal{R},p} t \qquad \Longleftrightarrow \qquad S \Rightarrow_{\mathcal{G}(\mathcal{R}),p} T$$

where T *represents* t

Adequacy of Graph Rewriting

Theorem

suppose s is a term and S is a term graph representing s such that for redex position p in s

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

Then

$$s \to_{\mathcal{R},p} t \qquad \iff \qquad S \Rightarrow_{\mathcal{G}(\mathcal{R}),p} T$$

where T represents t

Idea

- recover condition ❷ by extending rewrite relation with sharing

$$\overset{\geq}{\Rightarrow}_{\mathcal{G}} := \overset{\mathsf{i}}{\Rightarrow}_{\mathcal{G}} \cdot \geq$$

Adequacy of Graph Rewriting

Theorem

suppose s *is a term and* S *is a term graph representing* s *such that for redex position* p *in* s

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

Then

$$s \rightarrow_{\mathcal{R},p} t \qquad \Longleftrightarrow \qquad S \Rightarrow_{\mathcal{G}(\mathcal{R}),p} T$$

where T *represents* t

Idea

- condition ❶ is invariant on innermost $\mathcal{G}(\mathcal{R})$ reductions
- recover condition ❷ by extending rewrite relation with sharing

$$\stackrel{\geq}{\Rightarrow}_{\mathcal{G}} := \stackrel{\mathsf{i}}{\Rightarrow}_{\mathcal{G}} \cdot \geq$$

Adequacy of Graph Rewriting

Theorem

suppose s *is a term and* S *is a term graph representing* s *such that for redex position* p *in* s

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

Then

$$s \to_{\mathcal{R},p} t \quad\Longleftrightarrow\quad S \Rightarrow_{\mathcal{G}(\mathcal{R}),p} T$$

where T *represents* t

Theorem

$$s \xrightarrow{\mathsf{i}}{}^{\ell}_{\mathcal{R}} t \quad\Longleftrightarrow\quad S \overset{\geq}{\Rightarrow}{}^{\ell}_{\mathcal{G}(\mathcal{R})} T$$

for suitable graph representations S *and* T *of terms* s *and* t

Main Result Revisited

Theorem

If the (innermost) runtime-complexity of $\mathcal{R}$ is polynomially bounded, then each function f computed by $\mathcal{R}$ is polytime computable.

$$\mathrm{rc}^{\mathrm{i}}_{\mathcal{R}}(n) \leqslant n^k \;\Rightarrow\; f \in \mathsf{FP}$$

f computed by $\mathcal{R}$

Main Result Revisited

Theorem

If the (innermost) runtime-complexity of $\mathcal{R}$ is polynomially bounded, then each function f computed by $\mathcal{R}$ is polytime computable.

$$\mathrm{rc}^{\mathrm{i}}_{\mathcal{R}}(n) \leqslant n^k \ \Rightarrow\ f \in \mathrm{FP} \qquad f \text{ computed by } \mathcal{R}$$

Proof Idea

1. *employ innermost rewriting for computation of results*

Main Result Revisited

Theorem

If the (innermost) runtime-complexity of $\mathcal{R}$ is polynomially bounded, then each function f computed by $\mathcal{R}$ is polytime computable.

$$\mathrm{rc}^{\mathrm{i}}_{\mathcal{R}}(n) \leqslant n^k \;\Rightarrow\; f \in \mathrm{FP} \qquad f \text{ computed by } \mathcal{R}$$

Proof Idea

1. *employ innermost graph rewriting for computation of results*

   adequacy theorem

Main Result Revisited

Theorem

If the (innermost) runtime-complexity of $\mathcal{R}$ is polynomially bounded, then each function f computed by $\mathcal{R}$ is polytime computable.

$$\mathsf{rc}^{\mathrm{i}}_{\mathcal{R}}(n) \leqslant n^k \;\Rightarrow\; f \in \mathsf{FP}$$

f computed by $\mathcal{R}$

Proof Idea

1. *employ innermost graph rewriting for computation of results*

   adequacy theorem

2. *graphs grow only polynomial in size* $S \xrightarrow{\geq \ell}_{\mathcal{G}} T \Rightarrow |T| \leqslant |S| + \ell\Delta$

Main Result Revisited

Theorem

If the (innermost) runtime-complexity of $\mathcal{R}$ is polynomially bounded, then each function f computed by $\mathcal{R}$ is polytime computable.

$$\mathsf{rc}^{\mathrm{i}}_{\mathcal{R}}(n) \leqslant n^k \Rightarrow f \in \mathsf{FP} \qquad f \text{ computed by } \mathcal{R}$$

Proof Idea

1. *employ innermost graph rewriting for computation of results*
   adequacy theorem
2. *graphs grow only polynomial in size* $\quad S \xrightarrow{\geq}{}^{\ell}_{\mathcal{G}} T \Rightarrow |T| \leqslant |S| + \ell\Delta$
3. *each step computable in polynomial time*

Main Result Revisited

Theorem

If the (innermost) runtime-complexity of $\mathcal{R}$ is polynomially bounded, then each function f computed by $\mathcal{R}$ is polytime computable.

$$\mathsf{rc}^{\mathrm{i}}_{\mathcal{R}}(n) \leqslant n^k \;\Rightarrow\; f \in \mathsf{FP} \qquad f \text{ computed by } \mathcal{R}$$

Proof Idea

1. *employ innermost graph rewriting for computation of results* — *adequacy theorem*
2. *graphs grow only polynomial in size* — $S \xRightarrow{\geqslant}{}^{\ell}_{\mathcal{G}} T \Rightarrow |T| \leqslant |S| + \ell\Delta$
3. *each step computable in polynomial time in size of starting term*

Main Result Revisited

Theorem

If the (innermost) runtime-complexity of $\mathcal{R}$ is polynomially bounded, then each function f computed by $\mathcal{R}$ is polytime computable.

$$\mathrm{rc}^{\mathrm{i}}_{\mathcal{R}}(n) \leqslant n^k \;\Rightarrow\; f \in \mathsf{FP} \qquad f \text{ computed by } \mathcal{R}$$

Proof Idea

1. *employ innermost graph rewriting for computation of results* — *adequacy theorem*
2. *graphs grow only polynomial in size* — $S \xrightarrow{\geq}{}^{\ell}_{\mathcal{G}} T \Rightarrow |T| \leqslant |S| + \ell\Delta$
3. *each step computable in polynomial time in size of starting term*
4. *overall polynomial number of steps required*

Conclusion

Conclusion

- notion of runtime-complexity is a reason cost model for rewriting
  1. cost of computation naturally expressed
  2. polynomially related to actual cost on Turing machines

Conclusion

- notion of runtime-complexity is a reason cost model for rewriting
  1. cost of computation naturally expressed
  2. polynomially related to actual cost on Turing machines
- runtime-complexity analysis gives rise automation

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

Conclusion

- notion of runtime-complexity is a reason cost model for rewriting
  1. cost of computation naturally expressed
  2. polynomially related to actual cost on Turing machines
- runtime-complexity analysis gives rise automation
  - CaT
    http://cl-informatik.uibk.ac.at/research/software/ttt2
  - TCT
    http://cl-informatik.uibk.ac.at/research/software/tct
  - Matchbox/Poly
    http://dfa.imn.htwk-leipzig.de/matchbox/poly

Conclusion

- notion of runtime-complexity is a reason cost model for rewriting
  1. cost of computation naturally expressed
  2. polynomially related to actual cost on Turing machines
- runtime-complexity analysis gives rise automation
  - CaT
    http://cl-informatik.uibk.ac.at/research/software/ttt2
  - TCT
    http://cl-informatik.uibk.ac.at/research/software/tct
  - Matchbox/Poly
    http://dfa.imn.htwk-leipzig.de/matchbox/poly

Future Work

- extension of results to full rewriting — just finished
  - classification of nondeterministic computation possible

Conclusion

- notion of runtime-complexity is a reason cost model for rewriting
  1. cost of computation naturally expressed
  2. polynomially related to actual cost on Turing machines
- runtime-complexity analysis gives rise automation
  - CaT
    `http://cl-informatik.uibk.ac.at/research/software/ttt2`
  - TCT
    `http://cl-informatik.uibk.ac.at/research/software/tct`
  - Matchbox/Poly
    `http://dfa.imn.htwk-leipzig.de/matchbox/poly`

Future Work

- extension of results to full rewriting — just finished
  - classification of nondeterministic computation possible
- complexity preserving translations of (pure) functional programs