Automated Complexity Analysis of Term Rewrite Systems

Martin Avanzini (martin.avanzini@inria.fr)

informatiques mathématiques

ISR 2019, July 5–6, 2019

Introduction

```
let (∘) f g = fun z → f (g z) ;;
let rec walk = function
  | [] → id
  | x::xs → walk xs ∘ (fun ys → x::ys) ;;
let rev l = walk l [] ;;
```

Question: what is the runtime of `rev`?

Introduction

```
let (∘) f g = fun z → f (g z) ;;
let rec walk = function
  | [] → id
  | x::xs → walk xs ∘ (fun ys → x::ys) ;;
let rev l = walk l [] ;;
```

Question: what is the runtime of `rev`? depends on cost model

Introduction

```
let (∘) f g = fun z → f (g z) ;;
let rec walk = function
  | [] → id
  | x::xs → walk xs ∘ (fun ys → x::ys) ;;
let rev l = walk l [] ;;
```

Question: what is the runtime of `rev`? depends on cost model

1. Ideally, Worst Case Execution Time (μs on machine *X*)
   - analysis depends on compiler, OS, processor (caches, pipelines, branch prediction,...), etc.

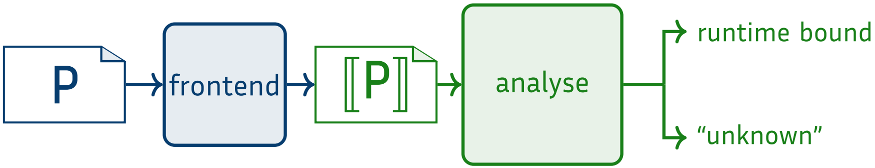
Introduction

```
let (∘) f g = fun z → f (g z) ;;
let rec walk = function
  | [] → id
  | x::xs → walk xs ∘ (fun ys → x::ys) ;;
let rev l = walk l [] ;;
```

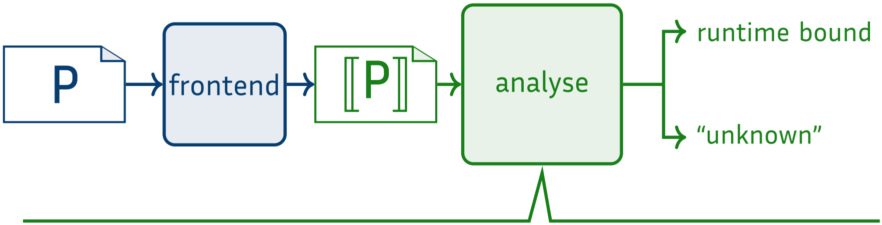
Question: what is the runtime of `rev`? depends on cost model

1. Ideally, Worst Case Execution Time (μs on machine *X*)
   - analysis depends on compiler, OS, processor (caches, pipelines, branch prediction,...), etc.
2. analysis of symbolic cost, e.g., #reduction steps
   - often informative enough while asymptotic precise
   - rewriting techniques can help inferring such bounds, automatically

Setup



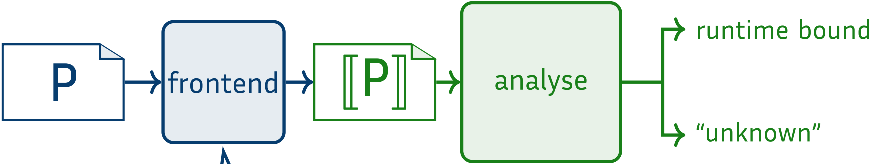
Setup



Fully Automated Rewriting Tools

- ⋆ AProVE `http://aprove.informatik.rwth-aachen.de`
- ⋆ CaT `http://cl-informatik.uibk.ac.at/software/cat`
- ⋆ Matchbox `http://dfa.imn.htwk-leipzig.de/matchbox`
- ⋆ TCT `http://cl-informatik.uibk.ac.at/software/tct`

Setup



- ★ Prolog
  - C. Otto et al. *"Automated Termination Analysis of Java Bytecode by Term Rewriting"*. In *Proc. of 21st RTA*, pp. 259–276, 2010.
- ★ Java / JBC
  - J. Giesl et al. *"Symbolic Evaluation Graphs and Term Rewriting - A General Methodology for Analyzing Logic Programs"*. In *Proc. of 22nd LOPSTR*, p. 1, 2012.
  - G. Moser and M. Schaper. *"From Jinja Bytecode to Term Rewriting: A Complexity Reflecting Transformation"*. *IC*, 2017.
- ★ OCaml
  - M. Avanzini, U. Dal Lago, and G. Moser. *"Analysing the Complexity of Functional Programs: Higher-Order Meets First-Order"*. In *Proc. of 20th ICFP*, pp. 152–164, 2015.

Today's Lecture

From Termination to Derivational Complexity Analysis

1. termination techniques and their induced complexity
2. inferring polynomial bounds

Rewriting as a Computational Model and Runtime Complexity

3. runtime complexity as a reasonable cost model
4. basic methods for polynomial runtime analysis

Tomorrow's Lecture

From Theory to Automation

5. towards a modular runtime complexity analysis
6. case study: TCT, its complexity framework

Applications to Program Analysis

8. case study: higher-order functional programs

Seminal Paper on Derivational Complexity

D. Hofbauer and C. Lautemann. *"Termination Proofs and the Length of Derivations"*. In *Proc. of 3rd RTA*, pp. 167–177, 1989.

Seminal Paper on Derivational Complexity

D. Hofbauer and C. Lautemann. *“Termination Proofs and the Length of Derivations”*. In *Proc. of 3rd RTA*, pp. 167–177, 1989.

Definition (induced derivational complexity)

Method X induces derivational complexity from class $\mathcal{C}$ if

$$\text{“}\mathcal{R}\text{ terminating by X”} \quad\Longrightarrow\quad \mathsf{dc}_{\mathcal{R}} \in \mathcal{C}\ .$$

Seminal Paper on Derivational Complexity

D. Hofbauer and C. Lautemann. *"Termination Proofs and the Length of Derivations"*. In *Proc. of 3rd RTA*, pp. 167–177, 1989.

Definition (induced derivational complexity)

Method X induces derivational complexity from class $\mathcal{C}$ if

$$\text{“}\mathcal{R}\text{ terminating by X”} \quad \Longrightarrow \quad \mathrm{dc}_{\mathcal{R}} \in \mathcal{C}\ .$$

Theorem (Hofbauer & Lautemann, RTA'89)

Polynomial Interpretations induced double-exponential derivational complexity.

Derivational Complexity (DC)

Definition (derivation height, derivational complexity)

consider ARS $\rightarrow \subseteq A \times A$ over objects A equipped with size : $A \rightarrow \mathbb{N}$

★ derivation height function wrt. $\rightarrow$ is

$$\mathsf{dh}_{\rightarrow} : A \rightarrow \mathbb{N} \cup \{\infty\}$$
$$\mathsf{dh}_{\rightarrow}(a) \triangleq \sup\{\ell \mid \exists(a_1, \ldots, a_\ell).\ a \rightarrow a_1 \rightarrow \ldots \rightarrow a_\ell\}$$

Derivational Complexity (DC)

Definition (derivation height, derivational complexity)

consider ARS $\rightarrow \subseteq A \times A$ over objects A equipped with $\mathsf{size}\colon A \rightarrow \mathbb{N}$

- ⋆ derivation height function wrt. $\rightarrow$ is

$$\mathsf{dh}_{\rightarrow}\colon A \rightarrow \mathbb{N} \cup \{\infty\}$$
$$\mathsf{dh}_{\rightarrow}(a) \triangleq \sup\{\ell \mid \exists(a_1, \ldots, a_\ell).\ a \rightarrow a_1 \rightarrow \ldots \rightarrow a_\ell\}$$

- ⋆ derivational complexity function wrt. $\rightarrow$ and start objects $S \subseteq A$ is

$$\mathsf{dc}_{\rightarrow,S}\colon \mathbb{N} \rightarrow \mathbb{N} \cup \{\infty\}$$
$$\mathsf{dc}_{\rightarrow,S}(n) \triangleq \sup\{\, \mathsf{dh}_{\rightarrow}(a) \mid a \in S, \mathsf{size}(a) \leq n \,\}\ .$$

Derivational Complexity (DC)

Definition (derivation height, derivational complexity)

consider ARS $\to \subseteq A \times A$ over objects A equipped with size: $A \to \mathbb{N}$

- derivation height function wrt. $\to$ is

$$\mathsf{dh}_{\to} \colon A \to \mathbb{N} \cup \{\infty\}$$
$$\mathsf{dh}_{\to}(a) \triangleq \sup\{\ell \mid \exists (a_1, \ldots, a_\ell).\ a \to a_1 \to \ldots \to a_\ell\}$$

- derivational complexity function wrt. $\to$ and start objects $S \subseteq A$ is

$$\mathsf{dc}_{\to,S} \colon \mathbb{N} \to \mathbb{N} \cup \{\infty\}$$
$$\mathsf{dc}_{\to,S}(n) \triangleq \sup\{\, \mathsf{dh}_{\to}(a) \mid a \in S, \mathsf{size}(a) \leq n \,\}\ .$$

- for TRS $\mathcal{R}$ over terms $\mathcal{T}$, derivational complexity is

$$\mathsf{dc}_{\mathcal{R}}(n) \triangleq \mathsf{dc}_{\to_{\mathcal{R}},\mathcal{T}}(n)\ .$$

Derivational Complexity (DC)

Example

$\to$	A	size	$\mathrm{dc}_{\to,A}$
$>_{\mathbb{N}}$	$\mathbb{N}$	id	

Derivational Complexity (DC)

Example

$\to$	A	size	$\mathrm{dc}_{\to,A}$
$>_{\mathbb{N}}$	$\mathbb{N}$	id	n
$>_{\mathbb{Z}}$	$\mathbb{Z}$	$\lvert\cdot\rvert$	

Derivational Complexity (DC)

Example

$\to$	A	size	$\mathrm{dc}_{\to,A}$
$>_{\mathbb{N}}$	$\mathbb{N}$	id	n
$>_{\mathbb{Z}}$	$\mathbb{Z}$	$\lvert \cdot \rvert$	∞
$>_{\mathbb{Q}}$	$\mathbb{Q}_{\geq 0}$	$\lceil \cdot \rceil$	

Derivational Complexity (DC)

Example

$\to$	A	size	$\mathrm{dc}_{\to,A}$
$>_{\mathbb{N}}$	$\mathbb{N}$	id	n
$>_{\mathbb{Z}}$	$\mathbb{Z}$	$\lvert\cdot\rvert$	∞
$>_{\mathbb{Q}}$	$\mathbb{Q}_{\geq 0}$	$\lceil\cdot\rceil$	∞
$>_{\mathbb{N}}^{\mathsf{prod}}$	$\mathbb{N}^k$	$\sum_{i=1}^{k} n_i$	

Derivational Complexity (DC)

Example

$\to$	A	size	$\mathrm{dc}_{\to,A}$
$>_{\mathbb{N}}$	$\mathbb{N}$	id	n
$>_{\mathbb{Z}}$	$\mathbb{Z}$	$\lvert\cdot\rvert$	∞
$>_{\mathbb{Q}}$	$\mathbb{Q}_{\geq 0}$	$\lceil\cdot\rceil$	∞
$>_{\mathbb{N}}^{\mathrm{prod}}$	$\mathbb{N}^k$	$\sum_{i=1}^{k} n_i$	n

$\mathcal{R}$	$\mathrm{dc}_{\mathcal{R}}(n)$
$\mathsf{a}(\mathsf{a}(x)) \to \mathsf{a}(\mathsf{b}(\mathsf{a}(x)))$	

Derivational Complexity (DC)

Example

$\rightarrow$	A	size	$\mathsf{dc}_{\rightarrow,A}$
$>_{\mathbb{N}}$	$\mathbb{N}$	id	n
$>_{\mathbb{Z}}$	$\mathbb{Z}$	$\lvert\cdot\rvert$	∞
$>_{\mathbb{Q}}$	$\mathbb{Q}_{\geq 0}$	$\lceil\cdot\rceil$	∞
$>_{\mathbb{N}}^{\mathsf{prod}}$	$\mathbb{N}^k$	$\sum_{i=1}^{k} n_i$	n

$\mathcal{R}$	$\mathsf{dc}_{\mathcal{R}}(n)$
$\mathsf{a}(\mathsf{a}(x)) \rightarrow \mathsf{a}(\mathsf{b}(\mathsf{a}(x)))$ $\mathsf{a}(\mathsf{b}(x)) \rightarrow \mathsf{b}(\mathsf{a}(x))$	$\mathcal{O}(n)$

Derivational Complexity (DC)

Example

$\to$	A	size	$\mathrm{dc}_{\to,A}$
$>_{\mathbb{N}}$	$\mathbb{N}$	id	n
$>_{\mathbb{Z}}$	$\mathbb{Z}$	$\lvert\cdot\rvert$	∞
$>_{\mathbb{Q}}$	$\mathbb{Q}_{\geq 0}$	$\lceil\cdot\rceil$	∞
$>_{\mathbb{N}}^{\mathsf{prod}}$	$\mathbb{N}^k$	$\sum_{i=1}^{k} n_i$	n

$\mathcal{R}$	$\mathrm{dc}_{\mathcal{R}}(n)$
$\mathsf{a}(\mathsf{a}(x)) \to \mathsf{a}(\mathsf{b}(\mathsf{a}(x)))$	$\mathcal{O}(n)$
$\mathsf{a}(\mathsf{b}(x)) \to \mathsf{b}(\mathsf{a}(x))$	$\mathcal{O}(n^2)$

Reduction Orders

Definition (rewrite order, reduction order)

- ⋆ a rewrite order is a proper order $\succ$ on that is:
  1. closed under substitutions: $s \succ t \implies s\sigma \succ t\sigma$
  2. closed under contexts: $s \succ t \implies C[s] \succ C[t]$
- ⋆ a reduction order is a well-founded rewrite order

Reduction Orders

Definition (rewrite order, reduction order)

- ⋆ a rewrite order is a proper order $\succ$ on that is:
  1. closed under substitutions: $s \succ t \implies s\sigma \succ t\sigma$
  2. closed under contexts: $s \succ t \implies C[s] \succ C[t]$
- ⋆ a reduction order is a well-founded rewrite order

Example

Knuth-Bendix Order, Multiset Path Order, Lexicographic Path Orders, Recursive Path Order, Interpretation Method, ...

Reduction Orders

Definition (rewrite order, reduction order)

★ a rewrite order is a proper order $\succ$ on that is:

1. closed under substitutions: $s \succ t \implies s\sigma \succ t\sigma$
2. closed under contexts: $s \succ t \implies C[s] \succ C[t]$

★ a reduction order is a well-founded rewrite order

Example

Knuth-Bendix Order, Multiset Path Order, Lexicographic Path Orders, Recursive Path Order, Interpretation Method, ...

Lemma

If rewrite order $\succ$ is compatible with TRS $\mathcal{R}$, i.e. $\mathcal{R} \subseteq \succ$, then

$$s \to_{\mathcal{R}} t \implies s \succ t\,.$$

Question: why?

Reduction Orders (II)

Theorem (Termination Via Reduction Orders)

TRS $\mathcal{R}$ is terminating iff *there exists a compatible reduction order $\succ$.*

Proof of Soundness ($\Leftarrow$).

- if $\succ$ is a rewrite order compatible with $\mathcal{R}$, then each reduction

$$t \to_{\mathcal{R}} t_1 \to_{\mathcal{R}} t_2 \to_{\mathcal{R}} \cdots ,$$

translates to

$$t \succ t_1 \succ t_2 \succ \cdots .$$

- if $\succ$ is well-founded, this sequence must be finite □

Reduction Orders (II)

Theorem (Termination Via Reduction Orders)

TRS $\mathcal{R}$ is terminating iff *there exists a compatible reduction order $\succ$.*

Proof of Soundness ($\Leftarrow$).

- ⋆ if $\succ$ is a rewrite order compatible with $\mathcal{R}$, then each reduction

$$t \to_{\mathcal{R}} t_1 \to_{\mathcal{R}} t_2 \to_{\mathcal{R}} \cdots ,$$

translates to $$t \succ t_1 \succ t_2 \succ \cdots .$$

- ⋆ if $\succ$ is well-founded, this sequence must be finite □

Theorem

If $\mathcal{R}$ is compatible with reduction order $\succ$ then

$$\mathsf{dc}_{\mathcal{R}}(n) \leq \mathsf{dc}_{\to_{\mathcal{R}} \cap \succ, \mathcal{T}}(n) \leq \mathsf{dc}_{\succ, \mathcal{T}}(n) .$$

Induced DC

- interpretation method
  - polynomial and matrix interpretations
- multiset path orders
- dependency pair method

Interpretation Method

Definition (well-founded monotone algebra, $>_{\mathcal{A}}$)

- ⋆ well-founded monotone algebra (WMA) $(\mathcal{A}, >)$ with carrier A consists of
  - well-founded proper order $> \subseteq A \times A$, and
  - strictly monotone interpretations $\mathtt{f}_{\mathcal{A}} : A^k \to A$ for every k-ary $\mathtt{f}$

$$a_i > b \quad \Longrightarrow \quad \mathtt{f}_{\mathcal{A}}(a_1, \ldots, a_i, \ldots, a_k) > \mathtt{f}_{\mathcal{A}}(a_1, \ldots, b, \ldots, a_k)$$

Interpretation Method

Definition (well-founded monotone algebra, $>_{\mathcal{A}}$)

- ⋆ well-founded monotone algebra (WMA) $(\mathcal{A}, >)$ with carrier A consists of
  - – well-founded proper order $> \subseteq A \times A$, and
  - – strictly monotone interpretations $\mathtt{f}_{\mathcal{A}} : A^k \to A$ for every k-ary $\mathtt{f}$

$$a_i > b \quad \Longrightarrow \quad \mathtt{f}_{\mathcal{A}}(a_1, \ldots, a_i, \ldots, a_k) > \mathtt{f}_{\mathcal{A}}(a_1, \ldots, b, \ldots, a_k)$$

- ⋆ induced order $>_{\mathcal{A}}$ on terms is

$$s >_{\mathcal{A}} t \quad :\Longleftrightarrow \quad [\![s]\!]^{\alpha}_{\mathcal{A}} > [\![t]\!]^{\alpha}_{\mathcal{A}} \text{ for all assignments } \alpha$$

where $[\![s]\!]^{\alpha}_{\mathcal{A}}$ is interpretation of s wrt. algebra $\mathcal{A}$ and assignment α.

Interpretation Method

Definition (well-founded monotone algebra, $>_{\mathcal{A}}$)

- well-founded monotone algebra (WMA) $(\mathcal{A}, >)$ with carrier A consists of
  - well-founded proper order $> \subseteq A \times A$, and
  - strictly monotone interpretations $\mathtt{f}_{\mathcal{A}} : A^k \to A$ for every k-ary $\mathtt{f}$

$$a_i > b \quad \Longrightarrow \quad \mathtt{f}_{\mathcal{A}}(a_1, \ldots, a_i, \ldots, a_k) > \mathtt{f}_{\mathcal{A}}(a_1, \ldots, b, \ldots, a_k)$$

- induced order $>_{\mathcal{A}}$ on terms is

$$s >_{\mathcal{A}} t \quad :\Longleftrightarrow \quad [\![s]\!]^{\alpha}_{\mathcal{A}} > [\![t]\!]^{\alpha}_{\mathcal{A}} \text{ for all assignments } \alpha$$

where $[\![s]\!]^{\alpha}_{\mathcal{A}}$ is interpretation of s wrt. algebra $\mathcal{A}$ and assignment α.

Lemma

If $(\mathcal{A}, >)$ is a WMA then $>_{\mathcal{A}}$ is a reduction order.

Polynomial Interpretations

Definition

Polynomial interpretation (PI) is WMA $(\mathcal{A}, >_{\mathbb{N}})$ where all interpretations $\mathtt{f}_{\mathcal{A}}$ are strictly monotone polynomials.

Polynomial Interpretations

Definition

Polynomial interpretation (PI) is WMA $(\mathcal{A}, >_{\mathbb{N}})$ where all interpretations $\mathtt{f}_{\mathcal{A}}$ are strictly monotone polynomials.

Example (I)

- ⋆ Consider the append function:

$$[\,] +\!\!+ ys \to ys \qquad\qquad (x :: xs) +\!\!+ ys \to x :: (xs +\!\!+ ys)\ .$$

- ⋆ terminating with polynomial interpretation?

Polynomial Interpretations

Definition

Polynomial interpretation (PI) is WMA $(\mathcal{A}, >_{\mathbb{N}})$ where all interpretations $\mathtt{f}_{\mathcal{A}}$ are strictly monotone polynomials.

Example (I)

- Consider the append function:

$$[\,] +\!\!+\, ys \to ys \qquad (x :: xs) +\!\!+\, ys \to x :: (xs +\!\!+\, ys)\,.$$

- terminating with polynomial interpretation? Yes, e.g.

$$n +\!\!+_{\mathcal{A}} m \triangleq 2 \cdot n + m \qquad [\,]_{\mathcal{A}} \triangleq 1 \qquad n ::_{\mathcal{A}} m \triangleq n + m\,.$$

Polynomial Interpretations

Definition

Polynomial interpretation (PI) is WMA $(\mathcal{A}, >_{\mathbb{N}})$ where all interpretations $\mathtt{f}_{\mathcal{A}}$ are strictly monotone polynomials.

Example (II)

- ⋆ Consider Ackermann function:

$$\mathtt{ack}(0, y) \to \mathtt{s}(y) \qquad \mathtt{ack}(\mathtt{s}(x), \mathtt{s}(y)) \to \mathtt{ack}(x, \mathtt{ack}(\mathtt{s}(x), y))$$

$$\mathtt{ack}(\mathtt{s}(x), 0) \to \mathtt{ack}(x, \mathtt{s}(0))$$

- ⋆ terminating with polynomial interpretation?

Polynomial Interpretations

Definition

Polynomial interpretation (PI) is WMA $(\mathcal{A}, >_{\mathbb{N}})$ where all interpretations $\mathtt{f}_{\mathcal{A}}$ are strictly monotone polynomials.

Example (II)

- ⋆ Consider Ackermann function:

$$\mathtt{ack}(0, y) \to \mathtt{s}(y) \qquad \mathtt{ack}(\mathtt{s}(x), \mathtt{s}(y)) \to \mathtt{ack}(x, \mathtt{ack}(\mathtt{s}(x), y))$$
$$\mathtt{ack}(\mathtt{s}(x), 0) \to \mathtt{ack}(x, \mathtt{s}(0))$$

- ⋆ terminating with polynomial interpretation? No, because …

Polynomial Interpretations

Definition

Polynomial interpretation (PI) is WMA $(\mathcal{A}, >_{\mathbb{N}})$ where all interpretations $\mathtt{f}_{\mathcal{A}}$ are strictly monotone polynomials.

Example (II)

- Consider Ackermann function:

$$\mathtt{ack}(0, y) \to \mathtt{s}(y) \qquad \mathtt{ack}(\mathtt{s}(x), \mathtt{s}(y)) \to \mathtt{ack}(x, \mathtt{ack}(\mathtt{s}(x), y))$$

$$\mathtt{ack}(\mathtt{s}(x), 0) \to \mathtt{ack}(x, \mathtt{s}(0))$$

- terminating with polynomial interpretation? No, because …

Theorem (Hofbauer & Lautemann, RTA'89)

PIs induce double-exponential DC. *(Bound is tight.)*

Polynomial Interpretations

Definition

Polynomial interpretation (PI) is WMA $(\mathcal{A}, >_{\mathbb{N}})$ where all interpretations $\mathtt{f}_{\mathcal{A}}$ are strictly monotone polynomials.

Example (II)

- Consider Ackermann function:

$$\mathtt{ack}(0, y) \to \mathtt{s}(y) \qquad \mathtt{ack}(\mathtt{s}(x), \mathtt{s}(y)) \to \mathtt{ack}(x, \mathtt{ack}(\mathtt{s}(x), y))$$

$$\mathtt{ack}(\mathtt{s}(x), 0) \to \mathtt{ack}(x, \mathtt{s}(0))$$

- terminating with polynomial interpretation? No, because ...

Theorem (Hofbauer & Lautemann, RTA'89)

PIs induce double-exponential DC. *(Bound is tight.)*

Question: how to prove this statement?

Polynomial Interpretations (II)

Definition (Upper-Bound)

Function $u\colon \mathbb{N} \to \mathbb{N}$ is upper-bound for PI $(\mathcal{A}, >_{\mathbb{N}})$ over signature $\mathcal{F}$ if:

$$\forall \mathtt{f} \in \mathcal{F}.\, \forall a \in A.\, \mathtt{f}_{\mathcal{A}}(a, \ldots, a) \leq u(a) \,.$$

Polynomial Interpretations (II)

Definition (Upper-Bound)

Function $u\colon \mathbb{N} \to \mathbb{N}$ is upper-bound for PI $(\mathcal{A}, >_{\mathbb{N}})$ over signature $\mathcal{F}$ if:

$$\forall \mathtt{f} \in \mathcal{F}.\, \forall a \in A.\, \mathtt{f}_{\mathcal{A}}(a, \ldots, a) \leq u(a)\ .$$

Lemma

Define $\alpha_0(x) \triangleq 0$. Suppose TRS $\mathcal{R}$ compatible with $(\mathcal{A}, >_{\mathbb{N}})$. Then:

$$\forall t.\, \mathsf{dh}_{\mathcal{R}}(t) \leq [\![t]\!]_{\mathcal{A}}^{\alpha_0} \leq u^{\mathsf{size}(t)}(0),\ \textit{hence}\ \mathsf{dc}_{\mathcal{R}}(n) \leq u^{n}(0)\ .$$

Polynomial Interpretations (II)

Definition (Upper-Bound)

Function $u\colon \mathbb{N} \to \mathbb{N}$ is upper-bound for PI $(\mathcal{A}, >_{\mathbb{N}})$ over signature $\mathcal{F}$ if:

$$\forall \mathtt{f} \in \mathcal{F}.\, \forall a \in A.\ \mathtt{f}_{\mathcal{A}}(a, \ldots, a) \leq u(a)\ .$$

Lemma

Define $\alpha_0(x) \triangleq 0$. Suppose TRS $\mathcal{R}$ compatible with $(\mathcal{A}, >_{\mathbb{N}})$. Then:

$$\forall t.\ \mathsf{dh}_{\mathcal{R}}(t) \leq [\![t]\!]_{\mathcal{A}}^{\alpha_0} \leq u^{\mathsf{size}(t)}(0), \textit{ hence } \mathsf{dc}_{\mathcal{R}}(n) \leq u^n(0)\ .$$

shape	upper-bound	induced DC
additive	$u(a) = a + \mathbf{d}$	$\mathcal{O}(n)$
linear	$u(a) = \mathbf{c} \cdot a + \mathbf{d}$	$\mathcal{O}(2^n)$
polynomial	$u(a) = \mathbf{c} \cdot a^{\mathbf{k}} + \mathbf{d}$	$\mathcal{O}(2^{2^n})$

Table: induced derivational complexity by shape; bounds are tight.

Polynomial Interpretations (II)

Example

TRS $\mathcal{R}_{++}$ consisting of rules

$$[\,] +\!\!+ ys \to ys \qquad (x :: xs) +\!\!+ ys \to x :: (xs +\!\!+ ys) .$$

terminating with polynomial interpretation

$$n +\!\!+_{\mathcal{A}} m \triangleq 2 \cdot n + m \qquad [\,]_{\mathcal{A}} \triangleq 1 \qquad n ::_{\mathcal{A}} m \triangleq n + m .$$

linear shape ⇒ classified exponential DC

shape	upper-bound	induced DC
additive	$u(a) = a + \mathbf{d}$	$\mathcal{O}(n)$
linear	$u(a) = \mathbf{c} \cdot a + \mathbf{d}$	$\mathcal{O}(2^n)$
polynomial	$u(a) = \mathbf{c} \cdot a^{\mathbf{k}} + \mathbf{d}$	$\mathcal{O}(2^{2^n})$

Table: induced derivational complexity by shape; bounds are tight.

Matrix Interpretations

Definition

Matrix interpretation (MI) of degree d is WMA $(\mathcal{A}, \gg)$ over $\mathbb{N}^d$ where

- ⋆ all interpretations $\mathtt{f}_\mathcal{A}$ are of the form

$$\mathtt{f}_\mathcal{A}(\vec{x_1}, \ldots, \vec{x_k}) = M_1 \cdot \vec{x_1} + \cdots + M_k \cdot \vec{x_k} + V$$

  where $V \in \mathbb{N}^d$ and $M_1, \ldots, M_k \in \mathbb{N}^{d \times d}$ with $(M_i)_{1,1} \geqslant 1$

- ⋆ $\vec{x} \gg \vec{y} :\Longleftrightarrow \; x_1 > y_1 \wedge \vec{x} \geqslant \vec{y}$

D. Hofbauer and J. Waldmann. *"Termination of String Rewriting with Matrix Interpretations"*. In *Proc. of 17th RTA*, pp. 328–342, 2006.

Matrix Interpretations

Definition

Matrix interpretation (MI) of degree d is WMA $(\mathcal{A}, \gg)$ over $\mathbb{N}^d$ where

- ⋆ all interpretations $\mathtt{f}_{\mathcal{A}}$ are of the form

$$\mathtt{f}_{\mathcal{A}}(\vec{x_1}, \ldots, \vec{x_k}) = M_1 \cdot \vec{x_1} + \cdots + M_k \cdot \vec{x_k} + V$$

  where $V \in \mathbb{N}^d$ and $M_1, \ldots, M_k \in \mathbb{N}^{d \times d}$ with $(M_i)_{1,1} \geqslant 1$
- ⋆ $\vec{x} \gg \vec{y} :\Longleftrightarrow \; x_1 > y_1 \wedge \vec{x} \geqslant \vec{y}$

Example

One-ruled TRS $\mathcal{R}_{\mathtt{aa}}$

$$\mathtt{a}(\mathtt{a}(x)) \to \mathtt{a}(\mathtt{b}(\mathtt{a}(x)))$$

compatible with matrix interpretation

$$\mathtt{a}_{\mathcal{A}}(\vec{n}) \triangleq \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \cdot \vec{n} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad \mathtt{b}_{\mathcal{A}}(\vec{n}) \triangleq \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \cdot \vec{n} \,.$$

Matrix Interpretations (II)

Theorem (Hofbauer & Waldmann, RTA'06)

MIs induce exponential DC.

D. Hofbauer and J. Waldmann. *"Termination of String Rewriting with Matrix Interpretations"*. In *Proc. of 17th RTA*, pp. 328–342, 2006.

Matrix Interpretations (II)

Theorem (Hofbauer & Waldmann, RTA'06)

MIs induce exponential DC.

Definition (Upper-triangular interpretation)

Matrix M is upper-triangular if

$$\forall i.\, M_{i,i} \leq 1 \qquad \text{and} \qquad \forall i > j. M_{i,j} = 0\,.$$

Theorem (Middeldorp et al. CAI'11)

MIs induce DC $\mathcal{O}(n^d)$ if all coefficients are upper-triangular with diagonal sum at most d.

A. Middeldorp et al. *"Joint Spectral Radius Theory for Automated Complexity Analysis of Rewrite Systems"*. In *Proc. of 4th CAI*, pp. 1–20, 2011.

D. Hofbauer and J. Waldmann. *"Termination of String Rewriting with Matrix Interpretations"*. In *Proc. of 17th RTA*, pp. 328–342, 2006.

Matrix Interpretations

Example

One-ruled TRS $\mathcal{R}_{\mathtt{aa}}$

$$\mathtt{a}(\mathtt{a}(x)) \to \mathtt{a}(\mathtt{b}(\mathtt{a}(x)))$$

compatible with matrix interpretation

$$\mathtt{a}_{\mathcal{A}}(\vec{n}) \triangleq \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \cdot \vec{n} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad \mathtt{b}_{\mathcal{A}}(\vec{n}) \triangleq \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \cdot \vec{n} \, .$$

Question: induced derivational complexity?

Matrix Interpretations

Example

One-ruled TRS $\mathcal{R}_{\mathtt{aa}}$

$$\mathtt{a}(\mathtt{a}(x)) \to \mathtt{a}(\mathtt{b}(\mathtt{a}(x)))$$

compatible with matrix interpretation

$$\mathtt{a}_{\mathcal{A}}(\vec{n}) \triangleq \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \cdot \vec{n} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad \mathtt{b}_{\mathcal{A}}(\vec{n}) \triangleq \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \cdot \vec{n} \, .$$

Question: induced derivational complexity? linear

Matrix Interpretations

Example

TRS $\mathcal{R}_{++}$ consisting of rules

$$[\,] ++ ys \to ys \qquad (x :: xs) ++ ys \to x :: (xs ++ ys)\;.$$

terminating with polynomial interpretation

$$[\,]_{\mathcal{A}} \triangleq \begin{bmatrix} 7 \\ 1 \end{bmatrix} \qquad \vec{x} ::_{\mathcal{A}} \vec{xs} \triangleq \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \cdot \vec{x} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \vec{xs} + \begin{bmatrix} 10 \\ 1 \end{bmatrix}$$

$$\vec{xs} ++_{\mathcal{A}} \vec{ys} \triangleq \begin{bmatrix} 1 & 9 \\ 0 & 1 \end{bmatrix} \cdot \vec{xs} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \vec{ys}\;.$$

- ⋆ induced derivational complexity? Quadratic
- ⋆ Question: bound asymptotic tight?

Matrix Interpretations

Example

TRS $\mathcal{R}_{++}$ consisting of rules

$$[\,] \mathbin{++} ys \to ys \qquad (x :: xs) \mathbin{++} ys \to x :: (xs \mathbin{++} ys)\ .$$

terminating with polynomial interpretation

$$[\,]_{\mathcal{A}} \triangleq \begin{bmatrix} 7 \\ 1 \end{bmatrix} \qquad \vec{x} ::_{\mathcal{A}} \vec{xs} \triangleq \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \cdot \vec{x} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \vec{xs} + \begin{bmatrix} 10 \\ 1 \end{bmatrix}$$

$$\vec{xs} \mathbin{++}_{\mathcal{A}} \vec{ys} \triangleq \begin{bmatrix} 1 & 9 \\ 0 & 1 \end{bmatrix} \cdot \vec{xs} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \vec{ys}\ .$$

- ⋆ induced derivational complexity? Quadratic
- ⋆ Question: bound asymptotic tight? Yes: $[e_1, \dots, e_n] \underbrace{\mathbin{++} \cdots \mathbin{++}}_{m \text{ times}} [\,]$

The Multiset Path Ordering (MPO)

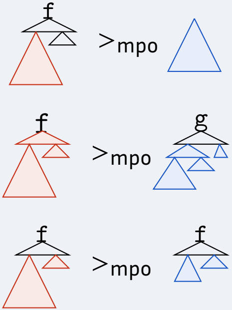
Definition (Multiset Path Order)

- ⋆ given precedence $>$ (proper, total order on function symbols)
- ⋆ induced multiset path order $>_{\mathsf{mpo}}$ is least order on terms s.t.

$$\frac{\exists i. s_i \geqslant_{\mathsf{mpo}} t}{\mathtt{f}(s_1, \ldots, s_i, \ldots, s_k) >_{\mathsf{mpo}} t}$$

$$\frac{\mathtt{f} > \mathtt{g} \quad \forall j. \mathtt{f}(s_1, \ldots, s_k) >_{\mathsf{mpo}} t_j}{\mathtt{f}(s_1, \ldots, s_k) >_{\mathsf{mpo}} \mathtt{g}(t_1, \ldots, t_k)}$$

$$\frac{\{s_1, \ldots, s_k\} >_{\mathsf{mpo}}^{\mathsf{mul}} \{t_1, \ldots, t_k\}}{\mathtt{f}(s_1, \ldots, s_k) >_{\mathsf{mpo}} \mathtt{f}(t_1, \ldots, t_k)}$$

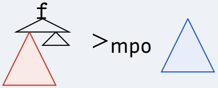


The Multiset Path Ordering (MPO)

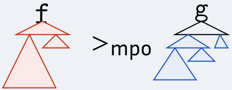
Definition (Multiset Path Order)

- ⋆ given precedence $>$ (proper, total order on function symbols)
- ⋆ induced multiset path order $>_{\mathsf{mpo}}$ is least order on terms s.t.

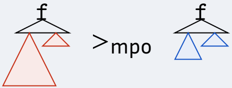
$$\frac{\exists i.\, s_i \geqslant_{\mathsf{mpo}} t}{\mathtt{f}(s_1, \ldots, s_i, \ldots, s_k) >_{\mathsf{mpo}} t}$$



$$\frac{\mathtt{f} > \mathtt{g} \quad \forall j.\, \mathtt{f}(s_1, \ldots, s_k) >_{\mathsf{mpo}} t_j}{\mathtt{f}(s_1, \ldots, s_k) >_{\mathsf{mpo}} \mathtt{g}(t_1, \ldots, t_k)}$$



$$\frac{\{s_1, \ldots, s_k\} >^{\mathsf{mul}}_{\mathsf{mpo}} \{t_1, \ldots, t_k\}}{\mathtt{f}(s_1, \ldots, s_k) >_{\mathsf{mpo}} \mathtt{f}(t_1, \ldots, t_k)}$$



Theorem

$>_{\mathsf{mpo}}$ *is a reduction order.*

MPO Characterizes Primitive Recursive Functions

Definition (Primitive Recursive Functions)

Class of primitive recursive functions (PR) is least set of functions over $\mathbb{N}$ s.t.

1. containing initial functions

$$\mathsf{zero}() \triangleq 0 \quad \mathsf{succ}(x) \triangleq x + 1 \quad \pi_{i,k}(x_1, \ldots, x_k) \triangleq x_i \quad (\forall 0 < i \leq k \in \mathbb{N}) \, ,$$

2. closed under composition

$$h, g_1, \ldots, g_k \in \mathsf{PR} \implies f(\vec{x}) \triangleq h(g_1(\vec{x}), \ldots, g_k(\vec{x})) \in \mathsf{PR} \, ,$$

3. closed under primitive recursion

$$g, h \in \mathsf{PR} \implies \begin{pmatrix} f(0, \vec{x}) \triangleq g(\vec{x}) \\ f(z+1, \vec{x}) \triangleq h(\vec{x}, f(z, \vec{x})) \end{pmatrix} \in \mathsf{PR} \, .$$

MPO Characterizes Primitive Recursive Functions

Definition (Rewriting Characterization of PR)

signature $\mathcal{F}_{\mathsf{PR}}$ and (infinite) rewrite system $\mathcal{R}_{\mathsf{PR}}$ inductively defined by:

1. constant $0 \in \mathcal{F}_{\mathsf{PR}}$, unary symbol $\mathtt{s} \in \mathcal{F}_{\mathsf{PR}}$ and

$$\mathtt{proj}_{i,k} \in \mathcal{F}_{\mathsf{PR}} \qquad \mathtt{proj}_{i,k}(x_1, \ldots, x_k) \to x_i \in \mathcal{R}_{\mathsf{PR}} \quad (\forall 0 < i \leq k \in \mathbb{N}) \,,$$

2. if $h, \vec{g} \in \mathcal{F}_{\mathsf{PR}}$ then

$$\mathsf{comp}[\vec{g}, h] \in \mathcal{F}_{\mathsf{PR}} \qquad \mathsf{comp}[\vec{g}, h](\vec{x}) \to h(g_1(\vec{x}), \ldots, g_k(\vec{x})) \in \mathcal{R}_{\mathsf{PR}} \,,$$

3. if $g, h \in \mathcal{F}_{\mathsf{PR}}$ then

$$\mathsf{rec}[g, h] \in \mathcal{F}_{\mathsf{PR}} \quad \left(\begin{array}{c} \mathsf{rec}[g, h](0, \vec{x}) \to g(\vec{x}) \\ \mathsf{rec}[g, h](z + 1, \vec{x}) \to h(\vec{x}, \mathsf{rec}[g, h](z, \vec{x})) \end{array} \right) \in \mathcal{R}_{\mathsf{PR}} \,.$$

E. A. Cichon and A. Weiermann. *"Term Rewriting Theory for the Primitive Recursive Functions"*. *APAL, Vol. 83*, pp. 199–223, 1997.

MPO Characterizes Primitive Recursive Functions

Theorem (PR $\Rightarrow$ MPO compatible)

Every $f \in PR$ is computed by some TRS compatible with MPO.

Proof Outline.

1. Every $f \in \mathrm{PR}$ is "computed" by finite $\mathcal{R}_f \subsetneq \mathcal{R}_{\mathrm{PR}}$.
2. $\mathcal{R}_f \subseteq \succ_{\mathrm{mpo}}$ where $\succ$ defined s.t.

$$\mathrm{comp}[\ldots, h, \ldots] \succ h\ , \qquad \mathrm{rec}[g, h] \succ g, h\ .$$

□

MPO Characterizes Primitive Recursive Functions

Theorem (PR $\Rightarrow$ MPO compatible)

Every $f \in PR$ is computed by some TRS compatible with MPO.

Proof Outline.

1. Every $f \in \mathrm{PR}$ is "computed" by finite $\mathcal{R}_f \subsetneq \mathcal{R}_{\mathrm{PR}}$.
2. $\mathcal{R}_f \subseteq >_{\mathrm{mpo}}$ where $>$ defined s.t.

$$\mathrm{comp}[\ldots, h, \ldots] > h \,, \qquad \mathrm{rec}[g, h] > g, h \,. \qquad \square$$

Theorem (Hofbauer, TCS'92)

MPO induces primitive recursive DC.

D. Hofbauer. "*Termination Proofs by Multiset Path Orderings Imply Primitive Recursive Derivation Lengths*". *TCS, Vol. 105*, pp. 129–140, 1992.

MPO Characterizes Primitive Recursive Functions

Theorem (PR ⇒ MPO compatible)

Every $f \in PR$ is computed by some TRS compatible with MPO.

Proof Outline.

1. Every $f \in \mathrm{PR}$ is "computed" by finite $\mathcal{R}_f \subsetneq \mathcal{R}_{\mathrm{PR}}$.
2. $\mathcal{R}_f \subseteq >_{\mathrm{mpo}}$ where $\succ$ defined s.t.

$$\mathsf{comp}[\ldots, h, \ldots] \succ h \,, \qquad \mathsf{rec}[g, h] \succ g, h \,. \qquad \square$$

Theorem (Hofbauer, TCS'92)

MPO induces primitive recursive DC.

Corollary (MPO compatible ⇒ PR)

If $\mathcal{R}$ "computes a function" $f \colon \mathbb{N}^k \to \mathbb{N}$ and $\mathcal{R}$ is compatible with MPO then $f \in PR$.

Dependency Pairs

Definition (Dependency Pair)

If $\mathtt{f}(l_1, \ldots, l_m) \to C[\mathrm{g}(t_1, \ldots, t_n)] \in \mathcal{R}$ with $\mathtt{g}$ defined by rule, then

$$\mathtt{f}^{\#}(l_1, \ldots, l_m) \to \mathtt{g}^{\#}(t_1, \ldots, t_n)$$

is a dependency pair (DP) of $\mathcal{R}$; $\mathsf{DP}(\mathcal{R})$ collects all DPs of $\mathcal{R}$.

T. Arts and J. Giesl. *"Proving Innermost Normalisation Automatically"*. In *Proc. of 8th RTA*, pp. 157–171, 1997.

Dependency Pairs

Definition (Dependency Pair)

If $\mathtt{f}(l_1, \ldots, l_m) \to C[\mathtt{g}(t_1, \ldots, t_n)] \in \mathcal{R}$ with $\mathtt{g}$ defined by rule, then

$$\mathtt{f}^{\#}(l_1, \ldots, l_m) \to \mathtt{g}^{\#}(t_1, \ldots, t_n)$$

is a dependency pair (DP) of $\mathcal{R}$; $\mathtt{DP}(\mathcal{R})$ collects all DPs of $\mathcal{R}$.

Example

$\mathcal{R}_{\mathsf{rev}}$	$\mathtt{DP}(\mathcal{R}_{\mathsf{rev}})$
$[\,] \mathbin{+\!\!+} ys \to ys$	
$(x :: xs) \mathbin{+\!\!+} ys \to x :: (xs \mathbin{+\!\!+} ys)$	$(x :: xs) \mathbin{+\!\!+}^{\#} ys \to xs \mathbin{+\!\!+}^{\#} ys$
$\mathtt{rev}([\,]) \to [\,]$	
$\mathtt{rev}(x :: xs) \to \mathtt{rev}(xs) \mathbin{+\!\!+} [x]$	$\mathtt{rev}^{\#}(x :: xs) \to \mathtt{rev}^{\#}(xs)$
	$\mathtt{rev}^{\#}(x :: xs) \to \mathtt{rev}(xs) \mathbin{+\!\!+}^{\#} [x]$

Dependency Pairs (II)

Theorem

TRS $\mathcal{R}$ is terminating iff there is no infinite and minimal chain

$$\mathtt{f}^{\#}(s_1, \ldots, s_m) \rightarrow_{\mathtt{DP}(\mathcal{R})} \mathtt{g}^{\#}(t_1, \ldots, t_n) \rightarrow^*_{\mathcal{R}} \mathtt{g}^{\#}(u_1, \ldots, u_n) \rightarrow_{\mathtt{DP}(\mathcal{R})} \ldots$$

T. Arts and J. Giesl. *"Proving Innermost Normalisation Automatically"*. In *Proc. of 8th RTA*, pp. 157–171, 1997.

Dependency Pairs (II)

Theorem

TRS $\mathcal{R}$ is terminating iff there is no infinite and minimal chain

$$\mathtt{f}^{\#}(s_1,\ldots,s_m) \rightarrow_{\mathtt{DP}(\mathcal{R})} \mathtt{g}^{\#}(t_1,\ldots,t_n) \rightarrow^{*}_{\mathcal{R}} \mathtt{g}^{\#}(u_1,\ldots,u_n) \rightarrow_{\mathtt{DP}(\mathcal{R})} \ldots$$

Proof techniques: reduction pairs, usable rules, subterm criterion, rule removal, narrowing, dependency graph cycle analysis, ...

R. Thiemann. *"The DP Framework for Proving Termination of Term Rewriting"*. "The DP Framework for Proving Termination of Term Rewriting", 2007.

Dependency Pairs (II)

Theorem

TRS $\mathcal{R}$ is terminating iff there is no infinite and minimal chain

$$f^{\#}(s_1, \ldots, s_m) \rightarrow_{\mathsf{DP}(\mathcal{R})} g^{\#}(t_1, \ldots, t_n) \rightarrow^{*}_{\mathcal{R}} g^{\#}(u_1, \ldots, u_n) \rightarrow_{\mathsf{DP}(\mathcal{R})} \cdots$$

Proof techniques: reduction pairs, usable rules, subterm criterion, rule removal, narrowing, dependency graph cycle analysis, ...

Theorem (Moser & Schnabl, RTA'09)

- ⋆ *DC of $\mathcal{R}$ can be double-exponential in length of $\rightarrow_{\mathsf{DP}(\mathcal{R})} \cdot \rightarrow^{*}_{\mathcal{R}}$ chains*
- ⋆ *non-primitive recursive overhead in dependency pair framework (subterm criterion + rule removal).*

G. Moser and A. Schnabl. *"The Derivational Complexity Induced by the Dependency Pair Method"*. In *Proc. of 20th RTA*, pp. 276–290, 2009.

Summary

* direct methods
  - Knuth-Bendix order 1969
  - **polynomial interpretations** 1975
  - lexicographic path order 1980
  - **multiset path order** 1982
  - context dependent interpretations 2001
  - match bounds 2003
  - **matrix interpretations** 2006
  - ...
* transformation methods
  - semantic labeling 1995
  - **dependency pairs** 1997
  - ...

Summary

- direct methods
  - Knuth-Bendix order — 2-rec, 2000 / 1969
  - **polynomial interpretations** — double-exp, 1989 / 1975
    - **additive** — **linear**, 2011
  - lexicographic path order — multi-rec, 1995 / 1980
  - **multiset path order** — prim-rec, 1990 / 1982
  - context dependent interpretations — double-exp, 2001 / 2001
  - match bounds — **linear**, 2003 / 2003
  - **matrix interpretations** — double-exp, 2006 / 2006
    - **triangular** — **polynomial**, 2011
  - ...
- transformation methods
  - semantic labeling — arbitrary overhead, 2008 / 1995
  - **dependency pairs** — 2-exp overhead, 2011 / 1997
  - ...

Runtime Complexity Analysis

- rewriting as a model of computation
- invariance theorem
- methods for assessing polynomial runtime

Derivational Complexity (II)

- consider TRS $\mathcal{R}_{\mathsf{dbl}}$ consisting of two rules:

$$\mathtt{dbl}(\mathtt{0}) \to \mathtt{0} \qquad \mathtt{dbl}(\mathtt{s}(x)) \to \mathtt{s}(\mathtt{s}(\mathtt{dbl}(x)))$$

- $\mathcal{R}_{\mathsf{dbl}}$ doubles natural numbers n in unary notation $\underline{n} = \underbrace{\mathtt{s}(\ldots\mathtt{s}}_{n \text{ times}}(\mathtt{0})\ldots)$

Derivational Complexity (II)

- consider TRS $\mathcal{R}_{\mathsf{dbl}}$ consisting of two rules:

$$\mathtt{dbl}(\mathtt{0}) \to \mathtt{0} \qquad \mathtt{dbl}(\mathtt{s}(x)) \to \mathtt{s}(\mathtt{s}(\mathtt{dbl}(x)))$$

- $\mathcal{R}_{\mathsf{dbl}}$ doubles natural numbers n in unary notation $\underline{n} = \underbrace{\mathtt{s}(\ldots\mathtt{s}}_{n \text{ times}}(\mathtt{0})\ldots)$
- complexity of function `dbl` is linear
- derivational complexity of $\mathcal{R}_{\mathsf{dbl}}$ is exponential

$$\mathsf{dh}_{\to_{\mathcal{R}_{\mathsf{dbl}}}}(\mathtt{dbl}(\underline{n})) = n + 1$$

$$\mathsf{dh}_{\to_{\mathcal{R}_{\mathsf{dbl}}}}(\mathtt{dbl}(\mathtt{dbl}(\underline{n}))) = (2 \cdot n + 1) + (n + 1)$$

$$\mathsf{dh}_{\to_{\mathcal{R}_{\mathsf{dbl}}}}(\mathtt{dbl}(\mathtt{dbl}(\mathtt{dbl}(\underline{n})))) = (4 \cdot n + 1) + (2 \cdot n + 1) + (n + 1)$$

$$\vdots$$

$$\mathsf{dh}_{\to_{\mathcal{R}_{\mathsf{dbl}}}}(\mathtt{dbl}^k(\underline{n})) = \sum_{i=0}^{k-1}(2^k \cdot n + 1)$$

Runtime Complexity of TRS

Definition (runtime complexity function)

Runtime complexity $\mathsf{rc}_{\mathcal{R}} : \mathbb{N} \to \mathbb{N} \cup \{\infty\}$ of TRS $\mathcal{R}$ is

$$\mathsf{rc}_{\mathcal{R}}(n) \triangleq \mathsf{dc}_{\to_{\mathcal{R}},\mathcal{B}}(n) \quad \text{with} \quad \underbrace{\mathcal{B} \triangleq \{\mathtt{f}(v_1,\dots,v_k) \mid \mathtt{f} \in \mathcal{D}, v_i \in \mathcal{V}al\}}_{\text{basic terms}},$$

where

- ⋆ signature partitioned into defined symbols $\mathcal{D}$ and constructors $\mathcal{C}$
  - – usually, $\mathcal{D}$ given implicitly by roots of left-hand sides
- ⋆ values $\mathcal{V}al$ are terms build from constructors $\mathcal{C}$

Runtime Complexity of TRS

Definition (runtime complexity function)

Runtime complexity $\mathsf{rc}_{\mathcal{R}} : \mathbb{N} \to \mathbb{N} \cup \{\infty\}$ of TRS $\mathcal{R}$ is

$$\mathsf{rc}_{\mathcal{R}}(n) \triangleq \mathsf{dc}_{\to_{\mathcal{R}},\mathcal{B}}(n) \quad \text{with} \quad \underbrace{\mathcal{B} \triangleq \{\mathtt{f}(v_1, \ldots, v_k) \mid \mathtt{f} \in \mathcal{D}, v_i \in \mathcal{V}al\}}_{\text{basic terms}} ,$$

where

- ⋆ signature partitioned into defined symbols $\mathcal{D}$ and constructors $\mathcal{C}$
  - usually, $\mathcal{D}$ given implicitly by roots of left-hand sides
- ⋆ values $\mathcal{V}al$ are terms build from constructors $\mathcal{C}$

Example

Runtime of $\mathcal{R}_{\mathsf{dbl}}$ is linear.

Rewriting as a Model of Computation

Definition (computation)

TRS $\mathcal{R}$ computes relation $R_{\mathtt{f}} \subseteq \mathcal{V}al^k \times \mathcal{V}al$ for each $\mathtt{f} \in \mathcal{D}$ s.t.

$$(v_1, \ldots, v_k) \; R_{\mathtt{f}} \; w \quad \Longleftrightarrow \quad \mathtt{f}(v_1, \ldots, v_k) \rightarrow^! w \in \mathcal{V}al \,.$$

Rewriting as a Model of Computation

Definition (computation)

TRS $\mathcal{R}$ computes relation $R_{\mathtt{f}} \subseteq \mathcal{V}al^k \times \mathcal{V}al$ for each $\mathtt{f} \in \mathcal{D}$ s.t.

$$(v_1, \ldots, v_k) \; R_{\mathtt{f}} \; w \iff \mathtt{f}(v_1, \ldots, v_k) \rightarrow^! w \in \mathcal{V}al \, .$$

Note: if $\mathcal{R}$ is confluent, $R_{\mathtt{f}}$ is a k-ary function

Rewriting as a Model of Computation

Definition (computation)

TRS $\mathcal{R}$ computes relation $R_{\mathtt{f}} \subseteq \mathcal{V}al^k \times \mathcal{V}al$ for each $\mathtt{f} \in \mathcal{D}$ s.t.

$$(v_1, \ldots, v_k)\ R_{\mathtt{f}}\ w \quad \Longleftrightarrow \quad \mathtt{f}(v_1, \ldots, v_k) \to^! w \in \mathcal{V}al\,.$$

Note: if $\mathcal{R}$ is confluent, $R_{\mathtt{f}}$ is a k-ary function

Question: is runtime complexity a reasonable cost model?

Rewriting as a Model of Computation

Definition (computation)

TRS $\mathcal{R}$ computes relation $R_{\mathtt{f}} \subseteq \mathcal{V}al^k \times \mathcal{V}al$ for each $\mathtt{f} \in \mathcal{D}$ s.t.

$$(v_1, \ldots, v_k)\ R_{\mathtt{f}}\ w \iff \mathtt{f}(v_1, \ldots, v_k) \to^! w \in \mathcal{V}al\,.$$

Note: if $\mathcal{R}$ is confluent, $R_{\mathtt{f}}$ is a k-ary function

Question: is runtime complexity a reasonable cost model?

1. counting #reduction steps is natural ✓
2. related to the cost of an "implementation" ?

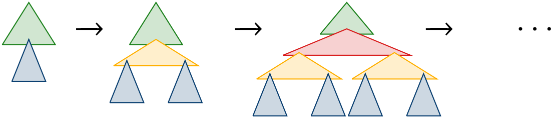
Invariance Thesis

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

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

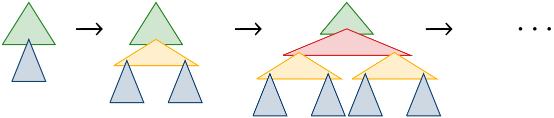
Invariance Thesis

- ⋆ invariance long lasting open question for rewriting based calculi
  - a single rewrite step may copy arbitrarily large terms
  - terms may grow exponential in the length of derivations

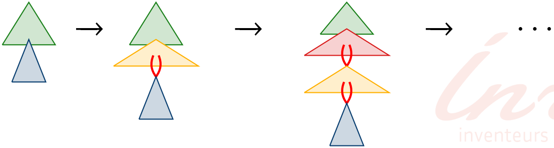


Invariance Thesis

- ⋆ invariance long lasting open question for rewriting based calculi
  - a single rewrite step may copy arbitrarily large terms
  - terms may grow exponential in the length of derivations

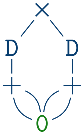
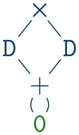


- ⋆ implementation via graph rewriting avoids space explosion
  - copying replaced by sharing
  - size-growth constant in length of derivation



Graph Rewriting in a Nutshell

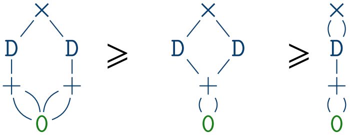
1. terms represented as graphs

graphs  $\geqslant$  $\geqslant$  represent $\mathtt{D}(x + x) \times \mathtt{D}(x + x)$

Graph Rewriting in a Nutshell

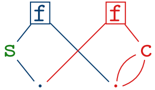
1. terms represented as graphs

graphs



represent $\mathtt{D}(x + x) \times \mathtt{D}(x + x)$

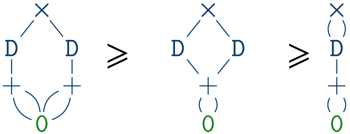
2. rules are graph with two designated roots for LHS $\boxed{\mathtt{f}}$ and RHS $\boxed{\mathtt{g}}$
   - unlabelled leafs act as variables



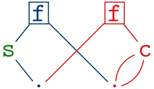
represents $\mathtt{f}(s(x_1), x_2) \to \mathtt{f}(x_1, c(x_2, x_2))$

Graph Rewriting in a Nutshell

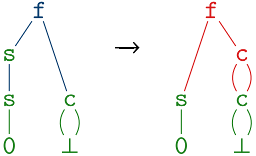
1. terms represented as graphs

   graphs  represent $\texttt{D}(x + x) \times \texttt{D}(x + x)$

2. rules are graph with two designated roots for LHS $\boxed{\texttt{f}}$ and RHS $\boxed{\texttt{g}}$
   - unlabelled leafs act as variables

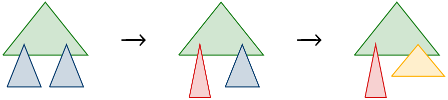
    represents $\texttt{f}(s(x_1), x_2) \to \texttt{f}(x_1, c(x_2, x_2))$

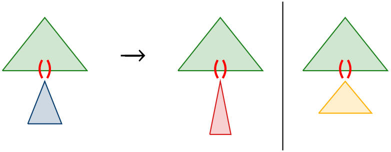
3. rule application replaces homomorphic copy of LHS with RHS



Discrepancies to Term Rewriting

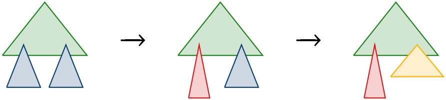
1. shared redexes cause parallel rewrites

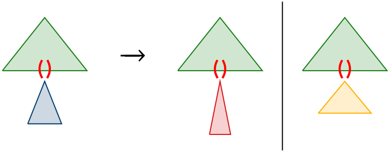
on terms,  ,

but  on graphs

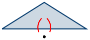
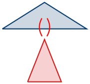
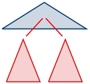
Discrepancies to Term Rewriting

1. shared redexes cause parallel rewrites

   on terms,  ,

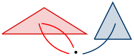
   but  on graphs

2. graph matching based on pointer equality

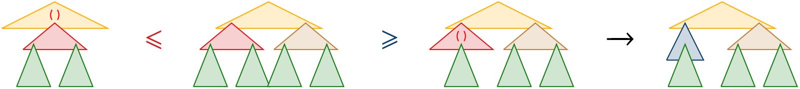
   LHS  matches  but matches not 

Implementing Term via Graph Rewriting

Folklore: term rewriting can be implemented via graph rewriting

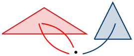
1. translate each rewrite rule $l \rightarrow r$ to graph rule 

2. unfold & fold graph before rule application

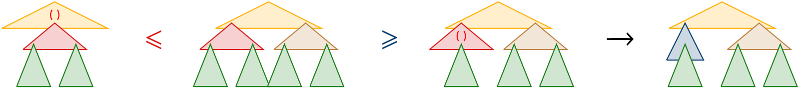


Implementing Term via Graph Rewriting

Folklore: term rewriting can be implemented via graph rewriting

1. translate each rewrite rule $l \to r$ to graph rule 

2. unfold & fold graph before rule application

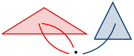


★ unfolding must be handled with care to avoid space-explosion

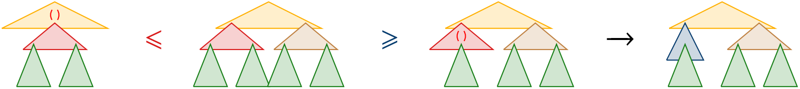
M. Avanzini and G. Moser. *"Closing the Gap Between Runtime Complexity and Polytime Computability"*. In *Proc. of 21st RTA*, pp. 33–48, 2010.

Implementing Term via Graph Rewriting

Folklore: term rewriting can be implemented via graph rewriting

1. translate each rewrite rule $l \to r$ to graph rule 

2. unfold & fold graph before rule application



★ unfolding must be handled with care to avoid space-explosion

★ observation gives rise to reduction relation $\lhd\!\!\blacktriangleright\!\!\!\to$ on graphs

   – restricted unfolding $\lhd$ copies *only* shared nodes along path to redex
   – restricted folding $\blacktriangleright$ introduces maximal sharing strictly below redex

M. Avanzini and G. Moser. *"Closing the Gap Between Runtime Complexity and Polytime Computability"*. In *Proc. of 21st RTA*, pp. 33–48, 2010.

Space Efficient Implementation of Term Rewriting

Theorem (Adequacy Theorem)

$$S \mathrel{\triangleleft\!\!\blacktriangleright\!\!\!\longrightarrow} T \quad \Longleftrightarrow \quad \mathrm{term}(S) \to \mathrm{term}(T)$$

Lemma (Time Lemma)

$$S \mathrel{\triangleleft\!\!\blacktriangleright\!\!\!\longrightarrow} T \quad \Longrightarrow \quad \textit{T computable from S in almost cubic time on TM}$$

Lemma (Space Lemma)

$$S \mathrel{\triangleleft\!\!\blacktriangleright\!\!\!\longrightarrow} T \quad \Longrightarrow \quad \mathrm{size}(T) \in \mathcal{O}(\ell \cdot \mathrm{size}(S) + \ell^2)$$

Invariance Theorem

Theorem (Invariance Theorem)

Let $\mathcal{R}$ be a confluent rewrite system with runtime $g(n)$.

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

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

Corollary (Polytime Invariance)

Let $\mathcal{R}$ be a confluent rewrite system with polynomially bounded runtime.

Then the functions computed by $\mathcal{R}$ are in FPTime.

Invariance Theorem

Theorem (*Non-deterministic* Invariance Theorem)

Let $\mathcal{R}$ be a rewrite system with runtime $g(n)$.

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

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

Corollary (*Non-deterministic* Polytime Invariance)

Let $\mathcal{R}$ be a rewrite system with polynomially bounded runtime.

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

Methods That Classify Polynomial RC

- polynomial & matrix interpretations, revisited
- usable argument positions
- polynomial path orders

Interpretations, Revisited

Central Observation:

- $\mathcal{R} \subseteq >_{\mathcal{A}} \implies \mathsf{dh}_{\rightarrow_{\mathcal{R}}}(\mathtt{f}(v_1, \ldots, v_k)) \leq \mathtt{f}_{\mathcal{A}}([\![v_1]\!]^{\alpha_0}_{\mathcal{A}}, \ldots, [\![v_k]\!]^{\alpha_0}_{\mathcal{A}})$
- for basic start terms, sufficient to control interpretations of constructors

Interpretations, Revisited

Central Observation:

- ★ $\mathcal{R} \subseteq >_{\mathcal{A}} \implies \mathsf{dh}_{\to_{\mathcal{R}}}(\mathtt{f}(v_1, \ldots, v_k)) \leq \mathtt{f}_{\mathcal{A}}([\![v_1]\!]^{\alpha_0}_{\mathcal{A}}, \ldots, [\![v_k]\!]^{\alpha_0}_{\mathcal{A}})$
- ★ for basic start terms, sufficient to control interpretations of constructors

Theorem

interpretation of constructors	*induced RC*	*characterisation*
additive	$\mathcal{O}(n^d)$ $^{(\dagger)}$	PTime
linear	$\mathcal{O}(2^n)$	ETime
polynomial	$\mathcal{O}(2^{2^n})$	E_2Time

(†) d is maximum degree of interpretations $\mathtt{f}_{\mathcal{A}}$ *for* $\mathtt{f} \in \mathcal{D}$*.*

G. Bonfante et al. *"Algorithms with Polynomial Interpretation Termination Proof"*. *JFP, Vol. 11*, pp. 33–53, 2001.

Interpretations, Revisited

Central Observation:

- ⋆ $\mathcal{R} \subseteq >_{\mathcal{A}} \implies \mathsf{dh}_{\to_{\mathcal{R}}}(\mathtt{f}(v_1, \ldots, v_k)) \leq \mathtt{f}_{\mathcal{A}}([\![v_1]\!]^{\alpha_0}_{\mathcal{A}}, \ldots, [\![v_k]\!]^{\alpha_0}_{\mathcal{A}})$
- ⋆ for basic start terms, sufficient to control interpretations of constructors

Theorem

interpretation of constructors	*induced RC*	*characterisation*
additive	$\mathcal{O}(n^d)$ (†)	PTime
linear	$\mathcal{O}(2^n)$	ETime
polynomial	$\mathcal{O}(2^{2^n})$	E_2Time

(†) d is maximum degree of interpretations $\mathtt{f}_{\mathcal{A}}$ for $\mathtt{f} \in \mathcal{D}$.

- ⋆ similar for MIs, induced RC controlled by restricting interpretation of constructors

Interpretations, Revisited

Example

TRS $\mathcal{R}_{+\!\!+}$ consisting of rules

$$[\,] +\!\!+ ys \to ys \qquad (x :: xs) +\!\!+ ys \to x :: (xs +\!\!+ ys)\,.$$

terminating with polynomial interpretation

$$n +\!\!+_{\mathcal{A}} m \triangleq 2 \cdot n + m \qquad [\,]_{\mathcal{A}} \triangleq 1 \qquad n ::_{\mathcal{A}} m \triangleq n + m\,.$$

★ linear shape ⇒ classified linear RC

Usable Argument Positions

Example

TRS $\mathcal{R}_{\div}$ consists of rules

$$x - 0 \to 0 \qquad 0 \div \mathtt{s}(y) \to 0$$
$$\mathtt{s}(x) - \mathtt{s}(y) \to x - y \qquad \mathtt{s}(x) \div \mathtt{s}(y) \to \mathtt{s}((x - y) \div \mathtt{s}(y))$$

Usable Argument Positions

Example

TRS $\mathcal{R}_{\div}$ consists of rules

$$x - 0 \to 0 \qquad 0 \div \mathtt{s}(y) \to 0$$

$$\mathtt{s}(x) - \mathtt{s}(y) \to x - y \qquad \mathtt{s}(x) \div \mathtt{s}(y) \to \mathtt{s}((x - y) \div \mathtt{s}(y))$$

★ Question: orientable by PI?

Usable Argument Positions

Example

TRS $\mathcal{R}_{\div}$ consists of rules

$$x - 0 \to 0 \qquad 0 \div \mathsf{s}(y) \to 0$$

$$\mathsf{s}(x) - \mathsf{s}(y) \to x - y \qquad \mathsf{s}(x) \div \mathsf{s}(y) \to \mathsf{s}((x - y) \div \mathsf{s}(y))$$

★ Question: orientable by PI? No, due to last rule

Usable Argument Positions

Example

TRS $\mathcal{R}_{\div}$ consists of rules

$$x - 0 \to 0 \qquad 0 \div \mathtt{s}(y) \to 0$$
$$\mathtt{s}(x) - \mathtt{s}(y) \to x - y \qquad \mathtt{s}(x) \div \mathtt{s}(y) \to \mathtt{s}((x - y) \div \mathtt{s}(y))$$

- ⋆ Question: orientable by PI? No, due to last rule
- ⋆ monotonicity required for closure under contexts:

$$s \to_{\mathcal{R}} t \wedge [\![s]\!]_{\mathcal{A}} > [\![t]\!]_{\mathcal{A}} \implies [\![\mathtt{f}(\ldots, s, \ldots)]\!]_{\mathcal{A}} > [\![\mathtt{f}(\ldots, t, \ldots)]\!]_{\mathcal{A}} \,.$$

Usable Argument Positions

Example

TRS $\mathcal{R}_{\div}$ consists of rules

$$x - 0 \rightarrow 0 \qquad 0 \div \mathsf{s}(y) \rightarrow 0$$
$$\mathsf{s}(x) - \mathsf{s}(y) \rightarrow x - y \qquad \mathsf{s}(x) \div \mathsf{s}(y) \rightarrow \mathsf{s}((x - y) \div \mathsf{s}(y))$$

- ⋆ Question: orientable by PI? No, due to last rule
- ⋆ monotonicity required for closure under contexts:

$$s \rightarrow_{\mathcal{R}} t \wedge [\![s]\!]_{\mathcal{A}} > [\![t]\!]_{\mathcal{A}} \implies [\![\mathtt{f}(\dots, s, \dots)]\!]_{\mathcal{A}} > [\![\mathtt{f}(\dots, t, \dots)]\!]_{\mathcal{A}} .$$

- ⋆ second argument of $-$ never reducible in reduction from basic term
  $\Rightarrow$ $[\![-]\!]_{\mathcal{A}}$ required monotonic only in first argument

Usable Argument Positions

Example

TRS $\mathcal{R}_{\div}$ consists of rules

$$x - 0 \to 0 \qquad 0 \div s(y) \to 0$$
$$s(x) - s(y) \to x - y \qquad s(x) \div s(y) \to s((x - y) \div s(y))$$

- ⋆ Question: orientable by PI? No, due to last rule
- ⋆ monotonicity required for closure under contexts:

$$s \to_{\mathcal{R}} t \wedge [\![s]\!]_{\mathcal{A}} > [\![t]\!]_{\mathcal{A}} \implies [\![f(\dots, s, \dots)]\!]_{\mathcal{A}} > [\![f(\dots, t, \dots)]\!]_{\mathcal{A}} .$$

- ⋆ second argument of $-$ never reducible in reduction from basic term
  $\Rightarrow$ $[\![-]\!]_{\mathcal{A}}$ required monotonic only in first argument
- ⋆ intuition formalised in notion of usable replacement map

Usable Arguments

Definition (Usable Replacement Map)

consider mapping μ s.t. $\mu(\mathtt{f}) \subseteq \{1, \ldots, k\}$ for every k-ary $\mathtt{f} \in \mathcal{F}$

N. Hirokawa and G. Moser. *"Automated Complexity Analysis Based on Context-Sensitive Rewriting"*. In *Proc. of 25th RTA and 12th TLCA*, pp. 257–271, 2014.

Usable Arguments

Definition (Usable Replacement Map)

consider mapping μ s.t. $\mu(\mathtt{f}) \subseteq \{1, \ldots, k\}$ for every k-ary $\mathtt{f} \in \mathcal{F}$

★ μ-positions $\mathcal{P}\mathsf{os}_\mu(t) \subseteq \mathcal{P}\mathsf{os}(t)$ in term t are

$$\mathcal{P}\mathsf{os}_\mu(x) \triangleq \{\epsilon\}$$

$$\mathcal{P}\mathsf{os}_\mu(\mathtt{f}(t_1, \ldots, t_k)) \triangleq \{\epsilon\} \cup \{i \cdot p \mid i \in \mu(\mathtt{f}),\ p \in \mathcal{P}\mathsf{os}_\mu(t_i)\}\ .$$

N. Hirokawa and G. Moser. *"Automated Complexity Analysis Based on Context-Sensitive Rewriting"*. In *Proc. of 25th RTA and 12th TLCA*, pp. 257–271, 2014.

Usable Arguments

Definition (Usable Replacement Map)

consider mapping μ s.t. $\mu(\mathtt{f}) \subseteq \{1, \dots, k\}$ for every k-ary $\mathtt{f} \in \mathcal{F}$

- ⋆ μ-positions $\mathcal{P}os_{\mu}(t) \subseteq \mathcal{P}os(t)$ in term t are

$$\mathcal{P}os_{\mu}(x) \triangleq \{\epsilon\}$$

$$\mathcal{P}os_{\mu}(\mathtt{f}(t_1, \dots, t_k)) \triangleq \{\epsilon\} \cup \{i \cdot p \mid i \in \mu(\mathtt{f}),\, p \in \mathcal{P}os_{\mu}(t_i)\} \;.$$

- ⋆ $\mathcal{T}_{\mu}(\to)$ is set of terms where only subterms at μ-positions are reducible wrt. $\to$

$$t \in \mathcal{T}_{\mu}(\to) :\Longleftrightarrow \forall p \in \mathcal{P}os(t) \setminus \mathcal{P}os_{\mu}(t).\; t|_p \in \mathsf{NF}(\to) \;.$$

N. Hirokawa and G. Moser. *"Automated Complexity Analysis Based on Context-Sensitive Rewriting"*. In *Proc. of 25th RTA and 12th TLCA*, pp. 257–271, 2014.

Usable Arguments

Definition (Usable Replacement Map)

consider mapping μ s.t. $\mu(\mathtt{f}) \subseteq \{1, \dots, k\}$ for every k-ary $\mathtt{f} \in \mathcal{F}$

- ⋆ μ-positions $\mathcal{P}\mathsf{os}_{\mu}(t) \subseteq \mathcal{P}\mathsf{os}(t)$ in term t are

$$\mathcal{P}\mathsf{os}_{\mu}(x) \triangleq \{\epsilon\}$$

$$\mathcal{P}\mathsf{os}_{\mu}(\mathtt{f}(t_1, \dots, t_k)) \triangleq \{\epsilon\} \cup \{i \cdot p \mid i \in \mu(\mathtt{f}),\ p \in \mathcal{P}\mathsf{os}_{\mu}(t_i)\} .$$

- ⋆ $\mathcal{T}_{\mu}(\rightarrow)$ is set of terms where only subterms at μ-positions are reducible wrt. $\rightarrow$

$$t \in \mathcal{T}_{\mu}(\rightarrow) :\Longleftrightarrow \forall p \in \mathcal{P}\mathsf{os}(t) \setminus \mathcal{P}\mathsf{os}_{\mu}(t).\ t|_p \in \mathsf{NF}(\rightarrow) .$$

- ⋆ μ is usable replacement map (URM) for TRS $\mathcal{R}$ on set of terms T

$$\rightarrow^*_{\mathcal{R}}(T) \subseteq \mathcal{T}_{\mu}(\rightarrow_{\mathcal{R}}) .$$

N. Hirokawa and G. Moser. *"Automated Complexity Analysis Based on Context-Sensitive Rewriting"*. In *Proc. of 25th RTA and 12th TLCA*, pp. 257–271, 2014.

Usable Arguments (II)

Definition (well-founded μ-monotone algebra)

well-founded μ-monotone algebra (WμMA) $(\mathcal{A}, >)$ with carrier A consists of

- ★ well-founded proper order $> \subseteq A \times A$, and
- ★ strictly μ-monotone interpretations $\mathtt{f}_{\mathcal{A}} : A^k \to A$ for every k-ary $\mathtt{f}$

$$a_i > b \wedge i \in \mu(\mathtt{f}) \implies \mathtt{f}_{\mathcal{A}}(a_1, \ldots, a_i, \ldots, a_k) > \mathtt{f}_{\mathcal{A}}(a_1, \ldots, b, \ldots, a_k)$$

Theorem

Let μ be a URM for $\mathcal{R}$ on basic terms $\mathcal{B}$. If WμMA $(\mathcal{A}, >)$ orients $\mathcal{R}$ then

$$\mathrm{rc}_{\mathcal{R}}(n) \leq \mathrm{dc}_{>_{\mathcal{A}}, \mathcal{B}}(n) .$$

Usable Arguments (III)

Example

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

$$x - 0 \to 0 \qquad 0 \div s(y) \to 0$$

$$s(x) - s(y) \to x - y \qquad s(x) \div s(y) \to s((x - y) \div s(y))$$

Usable Arguments (III)

Example

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

$$x - 0 \to 0 \qquad 0 \div s(y) \to 0$$

$$s(x) - s(y) \to x - y \qquad s(x) \div s(y) \to s((x - y) \div s(y))$$

★ Question: which maps constitute a URM for $\mathcal{R}_{\div}$?

symbol	μ_1	μ_2	μ_3	μ_4
s	$\varnothing$	$\varnothing$	$\{1\}$	$\{1\}$
$-$	$\varnothing$	$\varnothing$	$\varnothing$	$\{1, 2\}$
$\div$	$\varnothing$	$\{1\}$	$\{1\}$	$\{1, 2\}$

Usable Arguments (III)

Example

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

$$x - 0 \to 0 \qquad 0 \div s(y) \to 0$$

$$s(x) - s(y) \to x - y \qquad s(x) \div s(y) \to s((x - y) \div s(y))$$

- Question: which maps constitute a URM for $\mathcal{R}_{\div}$?

symbol	μ_1	μ_2	μ_3	μ_4
s	$\varnothing$	$\varnothing$	$\{1\}$	$\{1\}$
$-$	$\varnothing$	$\varnothing$	$\varnothing$	$\{1, 2\}$
$\div$	$\varnothing$	$\{1\}$	$\{1\}$	$\{1, 2\}$

- oriented by μ_3-monotone polynomial interpretation

$$0_{\mathcal{A}} \triangleq 1 \qquad s_{\mathcal{A}}(x) \triangleq x + 2 \qquad x -_{\mathcal{A}} y \triangleq x + 1 \qquad x \div_{\mathcal{A}} y \triangleq 3 \cdot x$$

- induced runtime complexity is linear

Recursive Path Orders and Polynomial RC

Motivation:

- ⋆ recursive path orders (e.g., MPO, LPO, KBO) fast to synthesise
- ⋆ can these orders be tamed to induce polynomial RC?

Yes!

- ⋆ polynomial path orders embody predicative recursion on MPO
- ⋆ induce (innermost) runtime complexity is polynomial

Predicative Recursion on Notation

Definition (predicative recursive functions)

BC is least set of functions over binary words s.t.

1. containing certain initial functions
2. closed under predicative composition

$$h, g_1, \ldots, g_{k+l} \in \mathsf{BC}$$
$$\implies f(\vec{x}; \vec{y}) \triangleq h(g_1(\vec{x};\,), \ldots, g_k(\vec{x};\,); g_{k+1}(\vec{x}; \vec{y}), \ldots, g_{k+l}(\vec{x}; \vec{y})) \in \mathsf{BC}\,,$$

3. closed under predicative recursion on notation

$$g, h_0, h_1 \in \mathsf{BC} \implies \left(\begin{array}{c} f(\epsilon, \vec{x}; \vec{y}) \triangleq g(\vec{x}; \vec{y}) \\ f(i \cdot z, \vec{x}; \vec{y}) \triangleq h_i(\vec{x}; \vec{y}, f(z, \vec{x}; \vec{y})) \end{array} \right) \in \mathsf{BC}\,.$$

S. Bellantoni and S. Cook. *"A new Recursion-Theoretic Characterization of the Polytime Functions"*. *CC, Vol. 2*, pp. 97–110, 1992.

Predicative Recursion on Notation

Definition (predicative recursive functions)

BC is least set of functions over binary words s.t.

1. containing certain initial functions
2. closed under predicative composition

$$h, g_1, \dots, g_{k+l} \in \mathrm{BC}$$
$$\implies f(\vec{x}; \vec{y}) \triangleq h(g_1(\vec{x};), \dots, g_k(\vec{x};); g_{k+1}(\vec{x}; \vec{y}), \dots, g_{k+l}(\vec{x}; \vec{y})) \in \mathrm{BC}\,,$$

3. closed under predicative recursion on notation

$$g, h_0, h_1 \in \mathrm{BC} \implies \left(\begin{array}{c} f(\epsilon, \vec{x}; \vec{y}) \triangleq g(\vec{x}; \vec{y}) \\ f(i \cdot z, \vec{x}; \vec{y}) \triangleq h_i(\vec{x}; \vec{y}, f(z, \vec{x}; \vec{y})) \end{array} \right) \in \mathrm{BC}\,.$$

Theorem

$$\mathrm{BC} = \mathrm{FPTime}\,.$$

Polynomial Path Orders (POP*)

Ingredients:

1. precedence $>$ on signature
2. for each symbol $\mathtt{f}$, separation of argument positions

$$\mathsf{normal}(\mathtt{f}) \uplus \mathsf{safe}(\mathtt{f}) = \{1, \ldots, \mathsf{ar}(\mathtt{f})\} \, .$$

M. Avanzini and G. Moser. *"Polynomial Path Orders"*. *LMCS, Vol. 9*, 2013.

Polynomial Path Orders (POP*)

Ingredients:

1. precedence $>$ on signature
2. for each symbol $\mathtt{f}$, separation of argument positions

$$\mathsf{normal}(\mathtt{f}) \uplus \mathsf{safe}(\mathtt{f}) = \{1, \ldots, \mathsf{ar}(\mathtt{f})\} \; .$$

Definition (auxiliary order $>_{\mathsf{pop}}$)

auxiliary order $>_{\mathsf{pop}}$ is least order on terms s.t.

$$\frac{\exists i.\, s_i \geqslant_{\mathsf{pop}} t \quad \mathtt{f} \in \mathcal{D} \implies i \in \mathsf{normal}(\mathtt{f})}{\mathtt{f}(s_1, \ldots, s_k) >_{\mathsf{pop}} t} \qquad \frac{\mathtt{f} > \mathtt{g} \quad \forall i.\, \mathtt{f}(\vec{x}) >_{\mathsf{pop}} t_i}{\mathtt{f}(\vec{s}) >_{\mathsf{pop}} \mathtt{g}(t_1, \ldots, t_k)}$$

M. Avanzini and G. Moser. *"Polynomial Path Orders"*. *LMCS, Vol. 9*, 2013.

Polynomial Path Orders (POP*)

Ingredients:

1. precedence $>$ on signature
2. for each symbol $\mathtt{f}$, separation of argument positions

$$\mathsf{normal}(\mathtt{f}) \uplus \mathsf{safe}(\mathtt{f}) = \{1, \ldots, \mathsf{ar}(\mathtt{f})\}\ .$$

Definition (auxiliary order $>_{\mathsf{pop}}$)

auxiliary order $>_{\mathsf{pop}}$ is least order on terms s.t.

$$\frac{\exists i.\ s_i \geqslant_{\mathsf{pop}} t \quad \mathtt{f} \in \mathcal{D} \implies i \in \mathsf{normal}(\mathtt{f})}{\mathtt{f}(s_1, \ldots, s_k) >_{\mathsf{pop}} t} \qquad \frac{\mathtt{f} > \mathtt{g} \quad \forall i.\ \mathtt{f}(\vec{x}) >_{\mathsf{pop}} t_i}{\mathtt{f}(\vec{s}) >_{\mathsf{pop}} \mathtt{g}(t_1, \ldots, t_k)}$$

Example

If $\mathtt{f} > \mathtt{g}$ then $\mathtt{f}(\mathtt{s}(;x);y) >_{\mathsf{pop}} \mathtt{g}(x;)$ but $\mathtt{f}(\mathtt{s}(;x);y) \not>_{\mathsf{pop}} \mathtt{g}(x;y)$

M. Avanzini and G. Moser. *"Polynomial Path Orders"*. *LMCS, Vol. 9*, 2013.

Polynomial Path Orders (POP*)

Ingredients:

1. precedence $>$ on signature
2. for each symbol $\mathtt{f}$, separation of argument positions

$$\mathsf{normal}(\mathtt{f}) \uplus \mathsf{safe}(\mathtt{f}) = \{1, \ldots, \mathsf{ar}(\mathtt{f})\} .$$

Definition (polynomial path order $>_{\mathsf{pop*}}$)

polynomial path order $>_{\mathsf{pop*}}$ is least order on terms s.t.

$$\frac{\exists i.\ s_i \geqslant_{\mathsf{pop*}} t}{\mathtt{f}(s_1, \ldots, s_k) >_{\mathsf{pop*}} t}$$

$$\frac{\begin{array}{c}\mathtt{f} \text{ occurs at most once in } \mathtt{g}(t_1, \ldots, t_k) \\ \mathtt{f} > \mathtt{g} \quad \forall i \in \mathsf{normal}(\mathtt{g}).\ \mathtt{f}(\vec{x}) >_{\mathsf{pop}} t_i \quad \forall i \in \mathsf{safe}(\mathtt{g}).\ \mathtt{f}(\vec{x}) >_{\mathsf{pop*}} t_i\end{array}}{\mathtt{f}(\vec{s}) >_{\mathsf{pop*}} \mathtt{g}(t_1, \ldots, t_k)}$$

$$\frac{\{s_1, \ldots, s_k\} >^{\mathsf{mul}}_{\mathsf{pop*}} \{t_1, \ldots, t_k\} \quad \exists i, j \in \mathsf{normal}(\mathtt{f}).\ s_i >_{\mathsf{pop*}} t_j}{\mathtt{f}(s_1, \ldots, s_k) >_{\mathsf{pop*}} \mathtt{f}(t_1, \ldots, t_k)}$$

Induced Runtime of POP*

Definition

Constructor TRS $\mathcal{R}$ is predicative recursive if compatible with $>_{\mathsf{pop*}}$.

Induced Runtime of POP*

Definition

Constructor TRS $\mathcal{R}$ is predicative recursive if compatible with $>_{\mathsf{pop}*}$.

Example

TRS

$$\mathtt{bt}(\mathtt{0};) \to \mathtt{L} \qquad \mathtt{bt}(\mathtt{s}(;n);) \to \mathtt{dup}(;\mathtt{bt}(n;)) \qquad \mathtt{dup}(;t) \to \mathtt{N}(;t,t) ,$$

is predicative recursive but has exponential runtime.

Induced Runtime of POP*

Definition

Constructor TRS $\mathcal{R}$ is predicative recursive if compatible with $>_{\mathsf{pop*}}$.

Example

TRS

$$\mathtt{bt}(0;) \to \mathtt{L} \qquad \mathtt{bt}(\mathtt{s}(;n);) \to \mathtt{dup}(;\mathtt{bt}(n;)) \qquad \mathtt{dup}(;t) \to \mathtt{N}(;t,t)\ ,$$

is predicative recursive but has exponential runtime.

Definition (Innermost Runtime Complexity (iRC))

$$\mathsf{rci}_{\mathcal{R}}(n) \triangleq \mathsf{dc}_{\xrightarrow{\mathsf{i}}_{\mathcal{R}},\mathcal{B}}(n)\ .$$

Theorem (A. & Moser, TCS'13)

If $\mathcal{R}$ *predicative recursive*, $\mathsf{rci}_{\mathcal{R}}(n) \leq p(n)$ *for some polynomial* p.

Further Notes on Recursive Path Orders

- ⋆ class of predicative recursive, confluent TRSs characterise FPTime

Further Notes on Recursive Path Orders

- ⋆ class of predicative recursive, confluent TRSs characterise FPTime
- ⋆ predicative recursive TRSs with single defined function can reach arbitrary iRC due to multiset status
- ⋆ restriction sPOP* (product status, weakened composition) of POP* induces bounds $\mathcal{O}(n^{\text{“recursion depth”}})$

M. Avanzini, N. Eguchi, and G. Moser. *“A new Order-theoretic Characterisation of the Polytime Computable Functions”*. *TCS, Vol. 585*, pp. 3–24, 2015.

Further Notes on Recursive Path Orders

- ★ class of predicative recursive, confluent TRSs characterise FPTime
- ★ predicative recursive TRSs with single defined function can reach arbitrary iRC due to multiset status
- ★ restriction sPOP* (product status, weakened composition) of POP* induces bounds $\mathcal{O}(n^{\text{“recursion depth”}})$
- ★ allowing multiple recursive calls retains FPTime characterisation via memoization

M. Avanzini, N. Eguchi, and G. Moser. *“A new Order-theoretic Characterisation of the Polytime Computable Functions”*. *TCS, Vol. 585*, pp. 3–24, 2015.

J.-Y. Marion. *“Analysing the Implicit Complexity of Programs”*. *IC, Vol. 183*, pp. 2–18, 2003.

Further Notes on Recursive Path Orders

- ⋆ class of predicative recursive, confluent TRSs characterise FPTime
- ⋆ predicative recursive TRSs with single defined function can reach arbitrary iRC due to multiset status
- ⋆ restriction sPOP* (product status, weakened composition) of POP* induces bounds $\mathcal{O}(n^{\text{"recursion depth"}})$
- ⋆ allowing multiple recursive calls retains FPTime characterisation via memoization
- ⋆ extending sPOP* with lexicographic status yields characterisation of exponential time functions

M. Avanzini, N. Eguchi, and G. Moser. *"A new Order-theoretic Characterisation of the Polytime Computable Functions"*. *TCS, Vol. 585*, pp. 3–24, 2015.

J.-Y. Marion. *"Analysing the Implicit Complexity of Programs"*. *IC, Vol. 183*, pp. 2–18, 2003.

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

Experimental Evaluation

```
$ cat lcs.raml
  firstline : L(int) -> L(int)
  firstline(l) = match l with
                  | nil -> nil
                  | (x::xs) -> +0::firstline xs;

  newline : (int,L(int),L(int)) -> L(int)
  newline (y,lastline,l) =
     match l with
       | nil     -> nil
       | (x::xs) -> match lastline with
                      | nil -> nil
                      | (belowVal::lastline') ->
                           let nl = newline(y,lastline',xs) in
                           let rightVal = right nl in
                           let diagVal =  right lastline' in
                           let elem = if x == y then diagVal+1 else max(belowVal,rightVal)
                           in elem::nl;
  right : L(int) -> int
  right l = match l with | nil -> +0 | (x::xs) -> x;

  lcstable : (L(int),L(int)) -> L(L(int))
  lcstable (l1,l2) = match l1 with
                       | nil -> [firstline l2]
                       | (x::xs) -> let m = lcstable (xs,l2) in
                                    match m with
                                      | nil -> nil
                                      | (l::ls) -> (newline (x,l,l2))::l::ls;

  lcs : (L(int),L(int)) -> int
  lcs(l1,l2) = let m = lcstable(l1,l2) in
               match m with | nil -> +0 | (l1::_) -> (match l1 with | nil -> +0 | (len::_) -> len);
```

Experimental Evaluation

```
$ raml2trs lcs.raml
  (STARTTERM CONSTRUCTOR-BASED)
  (STRATEGY INNERMOST)
  (VAR
    @_ @a @b @belowVal @diagVal @elem @l @l1 @l2 @lastline @lastline2 @len @ls @m @nl @rightVal
    @x @x_1 @x_2 @xs @y @y_1 @y_2)
  (RULES
    firstline(@l) -> firstline#1(@l)
    firstline#1(::(@x,@xs)) -> ::(#abs(#0()),firstline(@xs))
    firstline#1(nil) -> nil
    newline(@y,@lastline,@l) -> newline#1(@l,@lastline,@y)
    newline#1(::(@x,@xs),@lastline,@y) -> newline#2(@lastline,@x,@xs,@y)
    newline#1(nil,@lastline,@y) -> nil
    newline#2(::(@belowVal,@lastline2),@x,@xs,@y) ->
      newline#3(newline(@y,@lastline2,@xs),@belowVal,@lastline2,@x,@y)
    newline#2(nil,@x,@xs,@y) -> nil
    newline#3(@nl,@belowVal,@lastline2,@x,@y) ->
      newline#4(right(@nl),@belowVal,@lastline2,@nl,@x,@y)
    newline#4(@rightVal,@belowVal,@lastline2,@nl,@x,@y) ->
      newline#5(right(@lastline2),@belowVal,@nl,@rightVal,@x,@y)
    newline#5(@diagVal,@belowVal,@nl,@rightVal,@x,@y) ->
      newline#6(newline#7(#equal(@x,@y),@belowVal,@diagVal,@rightVal),@nl)
    newline#6(@elem,@nl) -> ::(@elem,@nl)
    newline#7(#false(),@belowVal,@diagVal,@rightVal) -> max(@belowVal,@rightVal)
    newline#7(#true(),@belowVal,@diagVal,@rightVal) -> +(@diagVal,#pos(#s(#0())))
    right(@l) -> right#1(@l)
    right#1(::(@x,@xs)) -> @x
    right#1(nil) -> #abs(#0())
    lcs(@l1,@l2) -> lcs#1(lcstable(@l1,@l2))
    lcs#1(@m) -> lcs#2(@m)

    [...]
```

Experimental Evaluation

Input	#rules	orders	TcT
appendAll	12	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
bfs	57	?	$\mathcal{O}(n)$
bft mmult	59	?	$\mathcal{O}(n^3)$
bitonic	78	?	$\mathcal{O}(n^4)$
bitvectors	148	?	$\mathcal{O}(n^2)$
clevermmult	39	?	$\mathcal{O}(n^2)$
duplicates	37	?	$\mathcal{O}(n^2)$
dyade	31	?	$\mathcal{O}(n^2)$
eratosthenes	74	?	$\mathcal{O}(n^2)$
flatten	31	?	$\mathcal{O}(n^2)$
insertionsort	36	?	$\mathcal{O}(n^2)$
listsort	56	?	$\mathcal{O}(n^2)$
lcs	87	?	$\mathcal{O}(n^2)$
matrix	74	?	$\mathcal{O}(n^3)$
mergesort	35	?	$\mathcal{O}(n^3)$
minsort	26	?	$\mathcal{O}(n^2)$
queue	35	?	$\mathcal{O}(n^5)$
quicksort	46	?	$\mathcal{O}(n^2)$
rationalPotential	14	$\mathcal{O}(n)$	$\mathcal{O}(n)$
splitandsort	70	?	$\mathcal{O}(n^3)$
subtrees	8	?	$\mathcal{O}(n^2)$
tuples	33	?	?

Figure: Analysis of translated resource aware ML programs.

Summary

- ⋆ RC is a reasonable cost model for rewriting
- ⋆ termination methods can be suited so as to induce polynomial RC
  - – amounts to "whole program analysis"
  - ⇒ intensionally weak

Summary

- ⋆ RC is a reasonable cost model for rewriting
- ⋆ termination methods can be suited so as to induce polynomial RC
  - – amounts to "whole program analysis"
  - ⇒ intensionally weak

Next Lecture: strengthen the analysis through modularity

1. combination of different techniques
2. analyse program parts (almost) independently