Complexity Analysis of Unfolding Graph Rewriting

Polynomial Complexity

Naohi Eguchi

Institute of Computer Science
University of Innsbruck
Austria

June 18, 2014, Computational Logic Seminar

Introduction 1/2

- Discussing computation resources (on Turing machines), underlying constructors are (implicitly) limited to simple ones:
  $\mathtt{nat} \to \mathtt{nat}$,
  $\mathtt{word} \to \mathtt{word}$,
  $\mathtt{nat} \times \mathtt{list(nat)} \to \mathtt{list(nat)}$, etc.
- Namely $\mathtt{tree} \times \cdots \times \mathtt{tree} \to \mathtt{tree}$ is not allowed.
- On the side of (sub-)recursive function theory, (primitive) recursion is limited to:

$$\begin{aligned} f(s(x), \vec{y}) &= h(x, \vec{y}, f(x, \vec{y})) \\ f(a(x), \vec{y}) &= h_a(x, \vec{y}, f(x, \vec{y})) \\ f(cons(x, xs), \vec{y}) &= h(x, xs, \vec{y}, f(xs, \vec{y})) \end{aligned}$$

- But the general form of primitive recursion is not considered:
  $f(c(x_1, \ldots, x_k), \vec{y}) = h_c(x_1, \ldots, x_k, \vec{y}, f(x_1, \vec{y}), \ldots, f(x_k, \vec{y}))$

Introduction 2/2

Example

$$\mathcal{R}: \begin{array}{lcl} g(\epsilon, z) & \to & z \\ f(0, y) & \to & \epsilon \end{array} \qquad \begin{array}{rcl} g(c(x, y), z) & \to & c(g(x, z), g(y, z)) \\ f(s(x), y) & \to & g(y, f(x, y)) \end{array}$$

Let $c^0(\epsilon) = \epsilon$, $c^{n+1}(\epsilon) = c(c^n(\epsilon), c^n(\epsilon))$.

$$\begin{array}{lll} f(s^m(0), c^n(\epsilon)) & \to^*_{\mathcal{R}} g(c^n(\epsilon), \ldots, g(c^n(\epsilon), \epsilon)) & \text{in } m \text{ steps} \\ & \to^*_{\mathcal{R}} g(c^n(\epsilon), \ldots, c^n(\epsilon)) & \text{in } O(2^n) \text{ steps} \\ & \to^*_{\mathcal{R}} c^{mn}(\epsilon) & \text{in } (m-1)O(2^n) \text{ steps} \end{array}$$

- $f(s^m(0), c^n(\epsilon)) \to^k_{\mathcal{R}} c^{mn}(\epsilon)$, $k \in O(m2^n) = O(|s^m(0)||c^n(\epsilon)|)$.
- $|c^{mn}(\epsilon)| \in O(2^{mn}) = O(2^{|s^m(0)|} \cdot |c^n(\epsilon)|)$.

Namely, rewriting in $\mathcal{R}$ leads to a normal form of exponential size in a polynomial step (measured by the sizes of starting terms). This does not happen on Turing machines.

Overview

- Direct complexity analysis of the general primitive recursion?
- Representing the general primitive recursion with infinite graph rewrite rules (Dal Lago, Martini and Zorzi).
- Precedence termination (Middeldorp, Ohsaki and Zantema).
- This work: Precedence termination with argument separation.
  $\Rightarrow$ Polynomial runtime complexity analysis of infinite graph rewrite systems.

Direct complexity analysis? 1/2

Complexity of TRSs refers to runtime complexity:
$\mathrm{rc}_{\mathcal{R}}(n) = \max\{k \mid (\exists s,t) s \to_{\mathcal{R}}^{k} t, |s| \leq n, s\text{: argument-normalised}\}$

Example

$$\mathcal{R}: \begin{array}{lll} g(\epsilon, z) & >_{\mathsf{rpo}} & z \\ f(0, y) & >_{\mathsf{rpo}} & \epsilon \end{array} \quad \begin{array}{rll} g(c(x,y), z) & >_{\mathsf{rpo}} & c(g(x,z), g(y,z)) \\ f(s(x), y) & >_{\mathsf{rpo}} & g(y, f(x,y)) \end{array}$$

By Hofbauer's theorem, the runtime complexity of $\mathcal{R}$ is at most primitive recursive.
Polynomial runtime complexity? (or polytime computability?)

- Polynomial path order (Avanzini-Moser):
  $g(c(x,y),z) \not>_{\mathsf{pop}^*} c(g(x,z), g(y,z))$
- Light multiset path order (Marion):
  $g(c(x,y),z) >_{\mathsf{lmpo}} c(g(x,z), g(y,z))$
  But LMPO induces polytime computability of compatible TRSs over simple constructors only.

Direct complexity analysis? 2/2

Example

$$\mathcal{R}: \begin{array}{lll} g(\epsilon, z) & \to & z \\ f(0, y) & \to & \epsilon \end{array} \quad \begin{array}{rll} g(c(x,y), z) & \to & c(g(x,z), g(y,z)) \\ f(s(x), y) & \to & g(y, f(x,y)) \end{array}$$

Polynomial interpretation possible?
Seems difficult.

- It must be $[c](x,y) = x + y + k$ for some constant $k \in \mathbb{N}$.

  Algorithms with Polynomial Interpretation Termination Proof
  Bonfante, Cichon, Marion and Touzet. 2001.
- Hence 3^{x+y+k} seems necessary for $[g](x,y)$ to have:

$$\begin{aligned} [g(c(x,y),z)] &= [g](x+y+k, z) \\ &> [g](x,z) + [g](y,z) + k = [c(g(x,z), g(y,z))] \end{aligned}$$

Unfolding graph rewriting 1/2

Representing the general primitive recursion with infinite graph rewrite rules.

General Ramified Recurrence is Sound for Polynomial Time
Dal Lago, Martini and Zorzi. Proc. DICE '10.

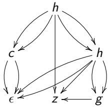
Idea of unfolding rewrite rules:

- Express equations
  $f(\epsilon, z) \to g(z)$, $f(c(x,y), z) \to h(x, y, z, f(x,z), f(y,z))$
  of general primitive recursion with infinite instances:
  $f(\epsilon, z) \to g(z)$,
  $f(c(\epsilon,\epsilon), z) \to h(\epsilon, \epsilon, z, g(z), g(z))$,
  $f(c(c(\epsilon,\epsilon), c(\epsilon,\epsilon)), z) \to$
  $h(c(\epsilon,\epsilon), c(\epsilon,\epsilon), z, h(\epsilon,\epsilon,z,g(z),g(z)), h(\epsilon,\epsilon,z,g(z),g(z)))$,...
- But then graph representation is more convenient.

Unfolding graph rewriting 2/2

$h(c(\epsilon,\epsilon), c(\epsilon,\epsilon), z, h(\epsilon,\epsilon,z,g(z),g(z)), h(\epsilon,\epsilon,z,g(z),g(z)))$
This term is expressed by the the following term graph.
Note:

- Definition of unfolding graph rewrite rules does not depend on the underlying TRS.
- They can be defined uniformly, independent of recursion terms.
- $\exists$ polytime algorithm s.t. $\lceil G \rceil \mapsto \lceil H \rceil$ if $G \xrightarrow{\mathrm{i}}_{\mathcal{G}} H$.



Theorem (Dal Lago-Martini-Zorzi '10)

$\forall f$: tiered recursive function $\exists \mathcal{G}$: infinite GRS defining f $\exists p$: poly. s.t. $G \xrightarrow{\mathrm{i}}{}^{k}_{\mathcal{G}} H \Longrightarrow \max\{k, |H|\} \leq p(|G|)$.

Precedence termination 1/2

Complexity Analysis of Unfolding Graph Rewriting: Primitive Recursive Complexity
Eguchi. Preprint.

- Primitive recursive (runtime) complexity analysis of infinite GRSs based on unfolding graph rewriting.
- Submitted to a Japanese workshop (PPL '14), but rejected due to many mistakes.
- A reviewer pointed out every unfolding graph rewrite rule is precedence terminating in the sense of:

  Transforming Termination by Self-Labeling
  Middeldorp, Ohsaki and Zantema. Proc. CADE '96.

Precedence termination 2/2

Precedence: well-founded binary relation over function symbols.

Definition (Middeldorp-Ohsaki-Zantema)

Let $>$: precedence.
A rewrite rule $f(\vec{t}) \to r$ is precedence terminating if
$f > g$ for any $g \in \{h : \text{function symbol} \mid h \text{ appears in } r\}$.

$f(c(\epsilon, \epsilon), z) \to h(\epsilon, \epsilon, z, g(z), g(z))$: precedence terminating if
$f > h$, $f > g$ and $f > \epsilon$.

- For finite TRSs, precedence termination only induces exponential runtime complexity.
- Precedence terminating infinite TRSs cover (more than) all the primitive recursive functions.
- Question. $\mathcal{R}$: prec. termination + ?? $\Rightarrow$ $\mathrm{rc}_{\mathcal{R}}$: polynomial.

Precedence termination with argument separation 1/4

Separation of argument positions of functions. (Safe recursion)

A New Recursion-theoretic Characterization of the Polytime Functions
Bellantoni and Cook. 1992.

Example

$$\mathcal{R}: \begin{array}{lclclcl} g(\epsilon; z) & \to & z & \quad & g(c(x,y); z) & \to & c(; g(x; z), g(y; z)) \\ f(0, y;) & \to & \epsilon & & f(s(x), y;) & \to & g(y; f(x, y;)) \end{array}$$

$f(x_1, \dots, x_k; x_{k+1}, \dots, x_{k+l})$: called normal arguments of f.
Observation. Starting with an argument-normalised term:

- Terms in normal argument positions are always normalised.
- Rewriting occurs only in non-normal positions.
- Note: the argument separation is not always possible.

Precedence termination with argument separation 2/4

Definition

Let $>$: precedence. A rewrite rule $f(\vec{s}; \vec{t}) \to r$ is precedence terminating with argument separation if:

1. $f(\vec{s}; \vec{t}) \to r$ is precedence terminating.
2. $\forall g(\vec{u}; \vec{v})$: subterm of r appearing in a non-normal position, $\vec{u}$ are sub-terms of $\vec{s}$.

The definition can be modified for graph rewrite rules.

Theorem

Suppose $\forall L \to R \in \mathcal{G}$ GRS prec. terminating with argument sep.:

1. *Variable nodes are maximally shared in R.*
2. *$|R| \leq |L| + m$. (m: size of subgraphs of L connected to normal positions of root_L)*

Then $\exists p$: poly. s.t. $G \to^k_{\mathcal{G}} H \Rightarrow \max\{k, |H|\} \leq p(|G|)$.

Precedence termination with argument separation 3/4

Note:

- In Theorem: $\mathcal{G}$ is restricted to a constructor GRS and G to an argument-normalised term graph.
- Every precedence terminating TRS is precedence terminating with the trivial argument separation $f(;x_1,\ldots,x_k)$.
- Hence the assumption 2 on size is essential.
- Weaker assumption $|R| \leq 2|L|$ only implies ptimitive recursive runtime complexity.

Every tiered recursive function can be expressed by a constructor GRS precedence terminating with argument separation that fulfills the assumptions 1 and 2 in Theorem. Hence:

- Fact by Dal Lago et al. can be reproved by the new method.
- Unlike the fact, innermost rewriting is not necessary as long as rewriting starts with an argument-normalised term graph.

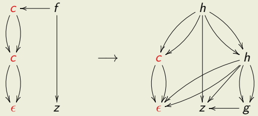
Precedence termination with argument separation 4/4

Example

$f(x;y)$

$h(x,y;z,u,v)$

$g(;x)$



- $L \to R$: precedence terminating with argument separation if $f > h$, $f > g$, $f > c$ and $f > \epsilon$.
- $|L| = 5$, $|R| = 6$, $m = 3$. Hence $|R| \leq 8 = |L| + m$.

$\forall L \to R \in \mathcal{G}$, $|R| \leq |L| + m$ (m: size of subgraphs of L connected to normal positions of root_L)

Conclusion

- Direct polynomial runtime complexity analysis of the general of primitive recursion is not known.
  $f(c(x_1,\ldots,x_k),\vec{y}) \to h_c(x_1,\ldots,x_k,\vec{y},f(x_1,\vec{y}),\ldots,f(x_k,\vec{y}))$
- Unfolding graph rewriting: infinite graph rewriting by which the general recursion can be related to polytime computability.
- This work: Complexity analysis of infinite GRSs based on unfolding graph rewriting.

Proving Termination of Unfolding Graph Rewriting for General Safe Recursion
N. Eguchi. Preprint, arXiv:1404.6196 (will be replaced soon!).

Thank you for your listening!