Proof Theory at Work: Complexity Analysis of Term Rewrite Systems

Georg Moser

Institute of Computer Science
University of Innsbruck

October 21, COMPLICE Workshop, Nancy
Overview

- A Crash Course in Rewriting
- Complexity Analysis of Term Rewriting Systems
- Discussion
- Open Issues
A Crash Course in Rewriting
A Simple Program

```haskell
let rec reverse l =
  match l with
  | [] -> []
  | h :: t ->
    (reverse t) @ [h];;

reverse(nil) = nil
reverse(n :: x) = app(reverse(x), n :: nil)

let rec shuffle l =
  match l with
  | [] -> []
  | h :: t ->
    h :: shuffle (reverse t);;

shuffle(nil) = nil
shuffle(n :: x) = n :: shuffle(reverse(x))

app(nil, y) = y
app(n :: x, y) = n :: app(x, y)
```

Example

shuffle[1, 2, 3, 4, 5] computes [1, 5, 2, 4, 3]
Signature
let signature $\mathcal{F}$ consists of

$$\text{reverse} \quad \text{shuffle} \quad \text{app} \quad :: \quad s \quad \text{nil} \quad 0$$

Rules
consider the TRS $\mathcal{R}$

$$\begin{align*}
\text{app}(\text{nil}, y) & \rightarrow y \\
\text{app}(n :: x, y) & \rightarrow n :: \text{app}(x, y) \\
\text{reverse}(\text{nil}) & \rightarrow \text{nil} \\
\text{reverse}(n :: x) & \rightarrow \text{app}(\text{reverse}(x), n :: \text{nil}) \\
\text{shuffle}(\text{nil}) & \rightarrow \text{nil} \\
\text{shuffle}(n :: x) & \rightarrow n :: \text{shuffle}(\text{reverse}(x))
\end{align*}$$

$$\begin{align*}
\text{shuffle}(1 :: 2 :: 3 :: 4 :: 5 :: \text{nil}) & \rightarrow_{\mathcal{R}} 1 :: \text{shuffle}(\text{reverse}(2 :: 3 :: 4 :: 5 :: \text{nil})) \\
& \rightarrow_{\mathcal{R}} 1 :: \text{shuffle}(\text{app}(\text{reverse}(3 :: 4 :: 5 :: \text{nil}), 2 :: \text{nil})) \\
& \rightarrow^{*}_{\mathcal{R}} 1 :: \text{shuffle}(\text{app}(\text{app}(\text{app}(\text{nil}, 5 :: \text{nil}), 4 :: \text{nil}), 3 :: \text{nil}), 2 :: \text{nil})) \\
& \rightarrow^{*}_{\mathcal{R}} 1 :: \text{shuffle}(5 :: 4 :: 3 :: 2 :: \text{nil}) \rightarrow^{*}_{\mathcal{R}} 1 :: 5 :: 2 :: 4 :: 3 :: \text{nil}
\end{align*}$$

(we abbreviate $s^n(0)$ as $n$)
What is Termination?

consider a TRS $\mathcal{R}$ and a term $t$ such that

$$t = t_1 \rightarrow_{\mathcal{R}} t_2 \rightarrow_{\mathcal{R}} t_3 \rightarrow_{\mathcal{R}} \cdots \rightarrow_{\mathcal{R}} t_n$$

Definition

a TRS is terminating if $\rightarrow_{\mathcal{R}}$ is well-founded

Definition

a reduction order $\succ$ is a well-founded order, such that if

$$\forall \ l \rightarrow r \in \mathcal{R} \Rightarrow l \succ r$$

compatibility

then $\mathcal{R}$ is terminating

Quest

find a compatible reduction order
The Multiset Path Order

let $>\,$ denote a precedence; $>\,$ induces order $>_{\text{mpo}}$:

$s = f(s_1, \ldots, s_n) >_{\text{mpo}} t$ if either

1. $\exists i \ s_i \geq_{\text{mpo}} t$

2. $t = f(t_1, \ldots, t_n)$ and
   
   $\{s_1, \ldots, s_n\} >_{\text{mpo}} \{t_1, \ldots, t_n\}$

3. $t = g(t_1, \ldots, t_m)$ with $f > g$ and
   
   $\forall i \ s >_{\text{mpo}} t_i$
**Example**

consider a precedence $>$ such that

$$\text{reverse} > \text{shuffle} > \text{app} > :: > s > \text{nil} > 0$$

consider the TRS $\mathcal{R}$

$$\begin{align*}
\text{app}(\text{nil}, y) & \rightarrow y & \text{app}(n :: x, y) & \rightarrow n :: \text{app}(x, y) \\
\text{reverse}(\text{nil}) & \rightarrow \text{nil} & \text{reverse}(n :: x) & \rightarrow \text{app}(\text{reverse}(x), n :: \text{nil}) \\
\text{shuffle}(\text{nil}) & \rightarrow \text{nil} & \text{shuffle}(n :: x) & \rightarrow n :: \text{shuffle}(\text{reverse}(x))
\end{align*}$$

$$\begin{align*}
\text{app}(n :: x, y) & >_{\text{mpo}} n :: \text{app}(x, y) \\
\text{shuffle}(n :: x) & \nless_{\text{mpo}} n :: \text{shuffle}(\text{reverse}(x))
\end{align*}$$
How-to Prove Termination of $\mathcal{R}$?

- $>_\text{mpo}$ not enough

Incomplete List
approximated dependency graph, argument filtering, bounds, dependency pair method, Knuth-Bendix order, lexicographic path order, loop detection, matrix interpretation, polynomial interpretation, predictive labeling, recursive SCC, root labeling, semantic labeling, simple projection and subterm criterion, uncurrying, usable rules

demo $T_1 T_2$
./ttt2 -p 3.12.trs
Complexity Analysis of Term Rewriting Systems
Consider

1. $\exists$ reduction order $\succ$ such that

2. $\mathcal{R} \subseteq \succ$

**Observation**

- $\mathcal{R} \subseteq \succ$ certifies termination of $\mathcal{R}$
- $\succ$ can be used to measure the derivation length
Definition

derivation length

\[ dl(t, \rightarrow) = \max\{ n \mid \exists u \ t \rightarrow^n u \} \]
\[ dl(n, T, \rightarrow) = \max\{ dl(t, \rightarrow) \mid \exists t \in T \text{ and } |t| \leq n \} \]

Definition

derivational complexity

\[ dc_R(n) = dl(n, "\text{all terms}", \rightarrow_R) \]

Definition

runtime complexity

\[ rc_R(n) = dl(n, "\text{basic terms}", \rightarrow_R) \]

term \( f(t_1, \ldots, t_n) \) is basic if
- \( f \) is defined
- \( t_1, \ldots, t_n \) contain no defined symbols
The Polytime Computable Functions

Definition

the class of polytime computable functions is the class of functions computable by a TM $M$ running in polynomial time. The class of polytime computable functions is the smallest class containing a certain set of initial functions and closed under predicative recursion on notation

1. containing a certain set of initial functions
2. closed under predicative recursion on notation

$$f(0, \bar{x}; \bar{y}) = g(\bar{x}; \bar{y})$$
$$f(z0, \bar{x}; \bar{y}) = h_0(z, \bar{x}; \bar{y}, f(u, \bar{x}; \bar{y}))$$
$$f(z1, \bar{x}; \bar{y}) = h_1(z, \bar{x}; \bar{y}, f(u, \bar{x}; \bar{y}))$$

3. closed under safe composition

$$f(\bar{x}; \bar{y}) = h(r_1(\bar{x};), \ldots, r_m(\bar{x};); s_1(\bar{x}; \bar{y}), \ldots, s_n(\bar{x}; \bar{y}))$$

Notation

$$f(\underbrace{x_1, \ldots, x_m}_{\text{normal}}, \underbrace{y_1, \ldots, y_n}_{\text{safe}})$$
Polynomial Path Orders \( \succ_{\text{pop}} \)

- \( \succ_{\text{pop}} \) is a restriction of \( \succ_{\text{mpo}} \): \( \succ_{\text{pop}} \subseteq \succ_{\text{mpo}} \)
- \( \succ_{\text{pop}} = \succ_{\text{mpo}} \cap \text{predicative recursion} \)

**predicative recursion:**

\[
\begin{align*}
  f(0, \vec{x}; \vec{y}) & \succ_{\text{pop}} g(\vec{x}; \vec{y}) \\
  f(z0, \vec{x}; \vec{y}) & \succ_{\text{pop}} h_0(z, \vec{x}; \vec{y}, f(z, \vec{x}; \vec{y})) \\
  f(z1, \vec{x}; \vec{y}) & \succ_{\text{pop}} h_1(z, \vec{x}; \vec{y}, f(z, \vec{x}; \vec{y}))
\end{align*}
\]

**Theorem**

let \( \mathcal{R} \) be constructor and right-linear

\[
\mathcal{R} \subseteq \succ_{\text{pop}} \quad \implies \quad \text{rc}_{\mathcal{R}} \text{ polynomially bounded}
\]
consider the TRS $S$

1: \( \text{half}(0) \rightarrow 0 \)
2: \( \text{half}(s(0)) \rightarrow 0 \)
3: \( \text{half}(s(s(x))) \rightarrow s(\text{half}(x)) \)
4: \( \text{bits}(0) \rightarrow 0 \)
5: \( \text{bits}(s(0)) \rightarrow s(0) \)
6: \( \text{bits}(s(s(x))) \rightarrow s(\text{bits}(s(\text{half}(x)))) \)

- \( \text{bits} \) computes length of binary representation
- \( S \) is compatible with \( >_{\text{mpo}} \), but not compatible with \( >_{\text{pop}^*} \)

**Incomplete List**
- approximated dependency graph
- arctic
- argument filtering
- linear polynomial interpretation
- matrix interpretation
- \( \text{pop}^* \)
- root labeling
- weak dependency pair method
- weak dependency graphs
- usable rules

**demo TCT**

tct -a irc -s ’wdg pop*’ rta09b.trs
Observation
POP* = LMPO - multiple recursive calls

Corollary
POP* characterises the polytime computable functions

Definition
- ∃ extension of POP* including predicative parameter substitution

\[
f(0, \vec{x}; \vec{y}) = g(\vec{x}; \vec{y}) \\
f(z, \vec{x}; \vec{y}) = h_i(z, \vec{x}; \vec{y}, f(z, \vec{x}; p_1(z, \vec{x}; \vec{y}), \ldots, p_m(z, \vec{x}; \vec{y})))
\]

- characterises polytime computable functions

Question
∃ similar extension of LMPO, I suppose?
Automated Complexity Analysis: A Snapshot
polynomial runtime complexity on 1404 TRSs

- TCT (304)
- transf + POP* (214)
- POP* (73)
- POP (40) fast

- it is still open how many of the examples in the testbed admit polynomial runtime complexity
## A More Detailed Analysis

<table>
<thead>
<tr>
<th></th>
<th>POP*</th>
<th>POP*&lt;sub&gt;ps&lt;/sub&gt;</th>
<th>LMPO</th>
<th>MPO</th>
<th>Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>yes</td>
<td>46</td>
<td>57</td>
<td>60</td>
<td>110</td>
<td>155</td>
</tr>
<tr>
<td>maybe</td>
<td>1262</td>
<td>1251</td>
<td>1241</td>
<td>1170</td>
<td>469</td>
</tr>
<tr>
<td>timeout</td>
<td>4</td>
<td>4</td>
<td>11</td>
<td>32</td>
<td>688</td>
</tr>
<tr>
<td>TC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>yes</td>
<td>40</td>
<td>51</td>
<td>54</td>
<td>66</td>
<td>89</td>
</tr>
<tr>
<td>maybe</td>
<td>530</td>
<td>519</td>
<td>513</td>
<td>493</td>
<td>288</td>
</tr>
<tr>
<td>timeout</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>11</td>
<td>193</td>
</tr>
<tr>
<td>TCO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>yes</td>
<td>20</td>
<td>23</td>
<td>25</td>
<td>30</td>
<td>47</td>
</tr>
<tr>
<td>maybe</td>
<td>280</td>
<td>277</td>
<td>275</td>
<td>270</td>
<td>157</td>
</tr>
<tr>
<td>timeout</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>96</td>
</tr>
<tr>
<td>average</td>
<td>0.04</td>
<td>0.06</td>
<td>0.05</td>
<td>0.08</td>
<td>6.46</td>
</tr>
</tbody>
</table>
Open Issues

as decided in the last Termination Competition Business meeting, the Termination Problem Data Base (TPDB) has been completely reorganized and will be soon public.

Observation
still the TPDB is a (very) bad database for complexity, but it is our only larger one.

Competition
termcomp.uibk.ac.at

the next competition takes place end of November
we are happy for any competition and any new example ...
Thank You for Your Attention!
Definition
we define \( s = f(s_1, \ldots, s_n) \stackrel{\text{pop}}{>}_t \) if \( s >_{\text{pop}} t \) or

1. \( \exists \ i \) such that \( s_i \stackrel{\text{pop}}{>}_t \)

2. \( t = f(t_1, \ldots, t_m) \) with \( f \in \mathcal{D} \) such that \( \text{nrm}(s) \stackrel{\text{mul}}{>}_{\text{pop}} \text{nrm}(t) \) and \( \text{safe}(s) \stackrel{\text{mul}}{>}_{\text{pop}} \text{safe}(t) \) holds

3. \( t = g(t_1, \ldots, t_m) \) with \( f > g, f \in \mathcal{D} \), and
   - \( s >_{\text{pop}} t_{j_0} \) for some \( j_0 \in \text{safe}(g) \), and
   - for all \( j \neq j_0 \) either \( s >_{\text{pop}} t_j \), or \( s \triangleright t_j \) and \( j \in \text{safe}(g) \), or

Theorem
POP* characterises the polytime computable functions
- in particular for constructor TRS \( \mathcal{R} \), if \( \mathcal{R} \subseteq >_{\text{pop}} \), then \( rc^i_{\mathcal{R}} \) is polynomially bounded