Constant Runtime Complexity of Term Rewriting is Semi-Decidable

Jürgen Giesl

LuFG Informatik 2, RWTH Aachen University, Germany

joint work with Florian Frohn
Runtime Complexity

Example TRS $\mathcal{R}$

\[
\begin{align*}
f(d) &\rightarrow c(g(d)) \\
g(c(x)) &\rightarrow c(g(f(x)))
\end{align*}
\]

Rewrite Sequence

\[
\begin{align*}
g(c(d)) &\rightarrow_{\mathcal{R}} c(g(f(d))) \\
c(g(c(g(d)))) &\rightarrow_{\mathcal{R}} c(c(g(f(g(d)))))
\end{align*}
\]

Defined Symbols: roots of lhs $f, g$

Constructors: other function symbols $c, d$

Basic Term: defined symbol only at the root $f(x), g(c(x)), g(c(d))$

Runtime Complexity $rc_{\mathcal{R}}(n)$

length of longest $\rightarrow_{\mathcal{R}}$-sequence starting with basic term $t$ where $|t| \leq n$

Example: $rc_{\mathcal{R}}(3) \geq 3$ since $|g(c(d))| = 3$
Runtime Complexity

**Contribution:** Constant runtime complexity is semi-decidable

\[ \mathcal{R} \text{ has constant runtime complexity} \]

\[ \iff \quad \text{rc}_{\mathcal{R}}(n) \in O(1) \]

\[ \iff \quad \exists m \in \mathbb{N}. \quad \text{all } \rightarrow_{\mathcal{R}}\text{-evaluations of basic terms take at most } m \text{ steps} \]

**Motivation**

*Complexity Analysis:* bounds on program’s resource usage

*Constant Bounds:* detect bugs

**Runtime Complexity** \( \text{rc}_{\mathcal{R}}(n) \)

length of longest \( \rightarrow_{\mathcal{R}}\)-sequence starting with basic term \( t \) where \( |t| \leq n \)
Constructor-Based Narrowing

Example TRS $\mathcal{R}$

$f(d) \rightarrow c(g(d))$
$g(c(x)) \rightarrow c(g(f(x)))$

Constructor-Based Narrowing Sequence

$g(x) \xrightarrow{\{x/c(x')\}} c(g(f(x'))) \xrightarrow{\{x'/d\}} c(g(c(g(d)))) \xrightarrow{\emptyset} c(c(g(f(g(d)))))$

Narrowing sequence $t_0 \xrightarrow{\sigma_1} \cdots \xrightarrow{\sigma_n} t_n$ is constructor based if $t_0 \sigma_1 \cdots \sigma_n$ is basic.

Constructor-based $\mathcal{R}$-narrowing terminates:

- $g(x), g(c(x)), g(c(d))$ 3 steps
- $g(c(t))$ for other terms $t$ 1 step
- $g(t)$ for other terms $t$ 0 steps
- $f(x), f(d)$ 1 step
- $f(t)$ for other terms $t$ 0 steps

Goal: Show termination of cb narrowing by inspecting finitely many start terms

Main Theorem

$\mathcal{R}$ has constant runtime complexity iff constructor-based $\mathcal{R}$-narrowing terminates
**Constructor-Based Narrowing**

**Example TRS \( \mathcal{R} \)**

\[
\begin{align*}
  f(d) & \rightarrow c(g(d)) \\
  g(c(x)) & \rightarrow c(g(f(x)))
\end{align*}
\]

**Constructor-Based Narrowing Sequence**

\[
\begin{align*}
  g(x) & \xrightarrow{x/c(x')} c(g(f(x'))) & \xrightarrow{x'/d} c(g(c(g(d)))) & \xrightarrow{\emptyset} c(c(g(f(g(d)))))
\end{align*}
\]

Narrowing sequence \( t_0 \xrightarrow{\sigma_1} \cdots \xrightarrow{\sigma_n} t_n \) is constructor based if \( t_0 \sigma_1 \cdots \sigma_n \) is basic.

For \( g(c(x)) \rightarrow c(g(x)) \), cb narrowing would not terminate:

\[
\begin{align*}
  g(x) & \xrightarrow{x/c(x')} c(g(x')) & \xrightarrow{x'/c(x'')} c(c(g(x''))) & \narrowing
\end{align*}
\]

**Goal:** Show termination of cb narrowing by inspecting finitely many start terms

**Main Theorem**

\( \mathcal{R} \) has constant runtime complexity iff constructor-based \( \mathcal{R} \)-narrowing terminates
Constructor-Based Narrowing

Example TRS $\mathcal{R}$

\[
f(d) \rightarrow c(g(d)) \quad g(c(x)) \rightarrow c(g(f(x)))
\]

Constructor-Based Narrowing Sequence

\[
\begin{align*}
g(x) & \xrightarrow{\{x/c(x')\}} c(g(f(x'))) \xrightarrow{\{x'/d\}} c(g(c(g(d)))) \xrightarrow{\emptyset} c(c(g(f(g(d))))) \\
g(c(x')) & \xrightarrow{\emptyset} c(g(f(x'))) \xrightarrow{\{x'/d\}} c(g(c(g(d)))) \xrightarrow{\emptyset} c(c(g(f(g(d))))) \\
g(c(d)) & \xrightarrow{\emptyset} c(g(f(d))) \xrightarrow{\emptyset} c(g(c(g(d)))) \xrightarrow{\emptyset} c(c(g(f(g(d))))) \\
\end{align*}
\]

Def: $s_0 \xrightarrow{\sigma_1_{\pi_1}} \ldots \xrightarrow{\sigma_n_{\pi_n}} s_n$ is more general than $t_0 \xrightarrow{\theta_1_{\pi_1}} \ldots \xrightarrow{\theta_n_{\pi_n}} t_n$ if there is a substitution $\eta$ with

\[
\begin{align*}
s_0 & \sigma_1 \sigma_2 \ldots \sigma_n \eta = t_0 \theta_1 \theta_2 \ldots \theta_n \\
s_1 & \sigma_2 \ldots \sigma_n \eta = t_1 \theta_2 \ldots \theta_n \\
& \ldots \\
s_n & \eta = t_n
\end{align*}
\]

Goal: Show termination of cb narrowing by inspecting finitely many start terms

Narrowing Lemma

For every $f(\ldots) \rightsquigarrow^n t$ there is a more general sequence $f(x_1, \ldots, x_k) \rightsquigarrow^n s$. 
Main Theorem

\( \mathcal{R} \) has constant runtime complexity iff constructor-based \( \mathcal{R} \)-narrowing terminates

Proof of “\( \Leftarrow \)”

Assume that \( \mathcal{R} \) does not have constant runtime complexity

\[
\Rightarrow \quad f(\vec{q}_1) \rightarrow^{n_1} t_1, \quad f(\vec{q}_2) \rightarrow^{n_2} t_2, \ldots \quad \text{with } n_1 < n_2 < \ldots
\]

\[
\Rightarrow \quad f(\vec{q}_1) \overset{\sigma}{\Rightarrow}^{n_1} t_1, \quad f(\vec{q}_2) \overset{\sigma}{\Rightarrow}^{n_2} t_2, \ldots
\]

\[
\Rightarrow \quad f(x_1, \ldots, x_k) \overset{\sigma_1}{\Rightarrow}^{n_1} s_1, \quad f(x_1, \ldots, x_k) \overset{\sigma_2}{\Rightarrow}^{n_2} s_2, \ldots \quad \text{by Narrowing Lemma}
\]

\[
\Rightarrow \quad \text{cb narrowing tree with root } f(x_1, \ldots, x_k)
\]

- has infinitely many nodes
- is finitely branching (as \( \mathcal{R} \) is finite)
- has infinite path (by König’s Lemma), i.e., infinite cb narrowing sequence

Narrowing Lemma

For every \( f(\ldots) \Rightarrow^n t \) there is a more general sequence \( f(x_1, \ldots, x_k) \Rightarrow^n s \).
Main Theorem

$\mathcal{R}$ has constant runtime complexity iff constructor-based $\mathcal{R}$-narrowing terminates

Proof of “⇒”

Assume that there is an infinite sequence $t_0 \xrightarrow{\sigma_1} t_1 \xrightarrow{\sigma_2} \cdots$

$\Rightarrow t_0 \sigma_1 \cdots \sigma_{m+1} \rightarrow^{m+1} t_{m+1}$ for all $m \in \mathbb{N}$

$\Rightarrow \forall m. \text{ there is an } \rightarrow_{\mathcal{R}}\text{-evaluation of a basic term with more than } m \text{ steps}$

$\Leftarrow \mathcal{R}$ does not have constant runtime complexity

Narrowing Lemma

For every $f(\ldots) \leadsto^n t$ there is a more general sequence $f(x_1, \ldots, x_k) \leadsto^n s$. 
Semi-Decision Procedure for Constant Runtime Complexity

Example TRS $\mathcal{R}$

\[
f(d) \rightarrow c(g(d)) \quad \quad \quad g(c(x)) \rightarrow c(g(f(x)))
\]

CB Narrowing Trees

\[
\begin{align*}
\overline{f(x)} & \xrightarrow{\{x/d\}} c(g(d)) \\
\overline{g(x)} & \xrightarrow{\{x/c(x')\}} c(g(f(x'))) & \xrightarrow{\{x'/d\}} c(g(c(g(d)))) & \xrightarrow{\emptyset} c(c(g(f(g(d)))))
\end{align*}
\]

Semi-Decision Procedure

- For all defined symbols $f$, build cb narrowing tree for $f(x_1, \ldots, x_k)$.
- If constructing the trees terminates, then return "constant runtime".
Theorem

Constant runtime complexity of TRSs is semi-decidable, but not decidable.

Proof

Turing machine $M$ is immortal

$\Leftrightarrow$ rewriting infinite basic terms with $R_M$ does not terminate

$\Leftrightarrow$ narrowing basic terms $f(x_1, \ldots, x_k)$ with $R_M$ does not terminate

$\Leftrightarrow$ $R_M$ does not have constant runtime complexity

(Im)mortality undecidable $\Rightarrow$ constant runtime complexity undecidable
Experiments

Theorem

Constant runtime complexity of TRSs is semi-decidable, but not decidable.

- Implementation and integration of semi-decision procedure in AProVE

- Full rewriting (959 examples from the TPDB, 60 s per example)
  - 57 TRSs with constant runtime
  - 57 TRSs detected by semi-decision procedure, 1.8 s avg. on successes
  - 51 TRSs detected by TcT or AProVE without semi-decision procedure

- Innermost rewriting (1022 examples from the TPDB, 60 s per example)
  - 59 TRSs with constant runtime
  - 58 TRSs detected by semi-decision procedure, 1.4 s avg. on successes
  - 1 TRS with relative rules not detected by semi-decision procedure
  - 55 TRSs detected by TcT or AProVE without semi-decision procedure
Constant Runtime Complexity of TRSs is Semi-Decidable

\[ \mathcal{R} \text{ has constant runtime complexity} \]
\[ \iff \exists m \in \mathbb{N}. \text{ all } \rightarrow_{\mathcal{R}}\text{-evaluations of basic terms take at most } m \text{ steps} \]

**Main Theorem**

\[ \mathcal{R} \text{ has constant runtime complexity} \iff \text{constructor-based } \mathcal{R}\text{-narrowing of all terms } f(x_1, \ldots, x_k) \text{ terminates} \]

**Semi-Decision Procedure (implemented in AProVE)**

- For all defined symbols \( f \), build cb narrowing tree for \( f(x_1, \ldots, x_k) \).
- If constructing the trees terminates, then return "constant runtime".

**Theorem**

Constant runtime complexity of TRSs is semi-decidable, but not decidable.