On Confluence of Parallel-Innermost Term Rewriting

Carsten Fuhs[1]

joint work with Thaïs Baudon[2] and Laure Gonnord[3,2]

[1] Birkbeck, University of London, UK
[2] LIP (UMR CNRS/ENS Lyon/UCB Lyon1/INRIA), Lyon, France
[3] LCIS (UGA/Grenoble INP/Ésisar), Valence, France

11th International Workshop on Confluence, Haifa, Israel, 1st August 2022

Overview

1. What is **parallel**-innermost rewriting?

Overview

1. What is **parallel**-innermost rewriting?
2. Proving confluence of **parallel**-innermost rewriting: how?

Overview

1. What is **parallel**-innermost rewriting?
2. Proving confluence of **parallel**-innermost rewriting: how?
3. Proving confluence of **parallel**-innermost rewriting: why?

Overview

1. What is **parallel**-innermost rewriting?
2. Proving confluence of **parallel**-innermost rewriting: how?
3. Proving confluence of **parallel**-innermost rewriting: why?

...or, how we tried to parallelise programs on term-shaped data structures,
then searched for complexity bounds of term rewrite systems,
and ended up building a (simple) confluence prover

Parallel-innermost rewriting: How it works

TRS $\mathcal{R}$:

$$\mathsf{d}(\mathsf{Zero}) \to \mathsf{Zero}$$
$$\mathsf{d}(\mathsf{S}(x)) \to \mathsf{S}(\mathsf{S}(\mathsf{d}(x)))$$
$$\mathsf{doubles}(\mathsf{Zero}) \to \mathsf{Nil}$$
$$\mathsf{doubles}(\mathsf{S}(x)) \to \mathsf{Cons}(\mathsf{d}(\mathsf{S}(x)), \mathsf{doubles}(x))$$

Parallel-innermost rewriting: How it works

TRS $\mathcal{R}$:

$$\mathsf{d}(\mathsf{Zero}) \to \mathsf{Zero}$$
$$\mathsf{d}(\mathsf{S}(x)) \to \mathsf{S}(\mathsf{S}(\mathsf{d}(x)))$$
$$\mathsf{doubles}(\mathsf{Zero}) \to \mathsf{Nil}$$
$$\mathsf{doubles}(\mathsf{S}(x)) \to \mathsf{Cons}(\mathsf{d}(\mathsf{S}(x)), \mathsf{doubles}(x))$$

Reduction with (sequential) innermost rewriting:

$$\mathsf{doubles}(\mathsf{S}(\mathsf{Zero}))$$

Parallel-innermost rewriting: How it works

TRS $\mathcal{R}$:

$$\mathsf{d}(\mathsf{Zero}) \to \mathsf{Zero}$$
$$\mathsf{d}(\mathsf{S}(x)) \to \mathsf{S}(\mathsf{S}(\mathsf{d}(x)))$$
$$\mathsf{doubles}(\mathsf{Zero}) \to \mathsf{Nil}$$
$$\mathsf{doubles}(\mathsf{S}(x)) \to \mathsf{Cons}(\mathsf{d}(\mathsf{S}(x)), \mathsf{doubles}(x))$$

Reduction with (sequential) innermost rewriting:

$$\begin{aligned} & \mathsf{doubles}(\mathsf{S}(\mathsf{Zero})) \\ \xrightarrow{\mathrm{i}}_{\mathcal{R}} \; & \mathsf{Cons}(\, \mathsf{d}(\mathsf{S}(\mathsf{Zero})) \,,\, \mathsf{doubles}(\mathsf{Zero}) \,) \end{aligned}$$

Parallel-innermost rewriting: How it works

TRS $\mathcal{R}$:

$$\mathsf{d}(\mathsf{Zero}) \to \mathsf{Zero}$$
$$\mathsf{d}(\mathsf{S}(x)) \to \mathsf{S}(\mathsf{S}(\mathsf{d}(x)))$$
$$\mathsf{doubles}(\mathsf{Zero}) \to \mathsf{Nil}$$
$$\mathsf{doubles}(\mathsf{S}(x)) \to \mathsf{Cons}(\mathsf{d}(\mathsf{S}(x)), \mathsf{doubles}(x))$$

Reduction with (sequential) innermost rewriting:

$$\begin{aligned} & \mathsf{doubles}(\mathsf{S}(\mathsf{Zero})) \\ \xrightarrow{\mathrm{i}}_{\mathcal{R}} \; & \mathsf{Cons}(\, \mathsf{d}(\mathsf{S}(\mathsf{Zero}))\,, \,\mathsf{doubles}(\mathsf{Zero})\,) \\ \xrightarrow{\mathrm{i}}_{\mathcal{R}} \; & \mathsf{Cons}(\, \mathsf{S}(\mathsf{S}(\mathsf{d}(\mathsf{Zero})))\,, \mathsf{doubles}(\mathsf{Zero})) \end{aligned}$$

Parallel-innermost rewriting: How it works

TRS $\mathcal{R}$:

$$\mathsf{d}(\mathsf{Zero}) \to \mathsf{Zero}$$
$$\mathsf{d}(\mathsf{S}(x)) \to \mathsf{S}(\mathsf{S}(\mathsf{d}(x)))$$
$$\mathsf{doubles}(\mathsf{Zero}) \to \mathsf{Nil}$$
$$\mathsf{doubles}(\mathsf{S}(x)) \to \mathsf{Cons}(\mathsf{d}(\mathsf{S}(x)), \mathsf{doubles}(x))$$

Reduction with (sequential) innermost rewriting:

$$\begin{aligned}
& \mathsf{doubles}(\mathsf{S}(\mathsf{Zero})) \\
\xrightarrow{\mathsf{i}}_{\mathcal{R}} \; & \mathsf{Cons}(\, \mathsf{d}(\mathsf{S}(\mathsf{Zero}))\,, \,\mathsf{doubles}(\mathsf{Zero})\,) \\
\xrightarrow{\mathsf{i}}_{\mathcal{R}} \; & \mathsf{Cons}(\, \mathsf{S}(\mathsf{S}(\mathsf{d}(\mathsf{Zero})))\,, \mathsf{doubles}(\mathsf{Zero})) \\
\xrightarrow{\mathsf{i}}_{\mathcal{R}} \; & \mathsf{Cons}(\mathsf{S}(\mathsf{S}(\mathsf{Zero})), \mathsf{doubles}(\mathsf{Zero}))
\end{aligned}$$

Parallel-innermost rewriting: How it works

TRS $\mathcal{R}$:

$$\mathsf{d}(\mathsf{Zero}) \to \mathsf{Zero}$$
$$\mathsf{d}(\mathsf{S}(x)) \to \mathsf{S}(\mathsf{S}(\mathsf{d}(x)))$$
$$\mathsf{doubles}(\mathsf{Zero}) \to \mathsf{Nil}$$
$$\mathsf{doubles}(\mathsf{S}(x)) \to \mathsf{Cons}(\mathsf{d}(\mathsf{S}(x)), \mathsf{doubles}(x))$$

Reduction with (sequential) innermost rewriting:

$$\begin{aligned}
& \mathsf{doubles}(\mathsf{S}(\mathsf{Zero})) \\
\xrightarrow{\mathsf{i}}_{\mathcal{R}} \; & \mathsf{Cons}(\mathsf{d}(\mathsf{S}(\mathsf{Zero})), \mathsf{doubles}(\mathsf{Zero})) \\
\xrightarrow{\mathsf{i}}_{\mathcal{R}} \; & \mathsf{Cons}(\mathsf{S}(\mathsf{S}(\mathsf{d}(\mathsf{Zero}))), \mathsf{doubles}(\mathsf{Zero})) \\
\xrightarrow{\mathsf{i}}_{\mathcal{R}} \; & \mathsf{Cons}(\mathsf{S}(\mathsf{S}(\mathsf{Zero})), \mathsf{doubles}(\mathsf{Zero})) \\
\xrightarrow{\mathsf{i}}_{\mathcal{R}} \; & \mathsf{Cons}(\mathsf{S}(\mathsf{S}(\mathsf{Zero})), \mathsf{Nil})
\end{aligned}$$

Parallel-innermost rewriting: How it works

TRS $\mathcal{R}$:

$$\mathsf{d}(\mathsf{Zero}) \to \mathsf{Zero}$$
$$\mathsf{d}(\mathsf{S}(x)) \to \mathsf{S}(\mathsf{S}(\mathsf{d}(x)))$$
$$\mathsf{doubles}(\mathsf{Zero}) \to \mathsf{Nil}$$
$$\mathsf{doubles}(\mathsf{S}(x)) \to \mathsf{Cons}(\mathsf{d}(\mathsf{S}(x)), \mathsf{doubles}(x))$$

Reduction with (sequential) innermost rewriting:

$$\begin{aligned}
& \mathsf{doubles}(\mathsf{S}(\mathsf{Zero})) \\
\xrightarrow{\mathrm{i}}_{\mathcal{R}} \; & \mathsf{Cons}(\, \mathsf{d}(\mathsf{S}(\mathsf{Zero}))\,,\, \mathsf{doubles}(\mathsf{Zero})\,) \\
\xrightarrow{\mathrm{i}}_{\mathcal{R}} \; & \mathsf{Cons}(\, \mathsf{S}(\mathsf{S}(\mathsf{d}(\mathsf{Zero})))\,, \mathsf{doubles}(\mathsf{Zero})) \\
\xrightarrow{\mathrm{i}}_{\mathcal{R}} \; & \mathsf{Cons}(\mathsf{S}(\mathsf{S}(\mathsf{Zero})), \mathsf{doubles}(\mathsf{Zero})) \\
\xrightarrow{\mathrm{i}}_{\mathcal{R}} \; & \mathsf{Cons}(\mathsf{S}(\mathsf{S}(\mathsf{Zero})),\, \mathsf{Nil}\,)
\end{aligned}$$

4 steps.

Parallel-innermost rewriting: How it works

TRS $\mathcal{R}$:

$$\mathsf{d}(\mathsf{Zero}) \to \mathsf{Zero}$$
$$\mathsf{d}(\mathsf{S}(x)) \to \mathsf{S}(\mathsf{S}(\mathsf{d}(x)))$$
$$\mathsf{doubles}(\mathsf{Zero}) \to \mathsf{Nil}$$
$$\mathsf{doubles}(\mathsf{S}(x)) \to \mathsf{Cons}(\mathsf{d}(\mathsf{S}(x)), \mathsf{doubles}(x))$$

Reduction with (sequential) innermost rewriting:

$$\begin{aligned}
& \mathsf{doubles}(\mathsf{S}(\mathsf{Zero})) \\
\xrightarrow{\mathrm{i}}_{\mathcal{R}} \ & \mathsf{Cons}(\ \mathsf{d}(\mathsf{S}(\mathsf{Zero}))\ ,\ \mathsf{doubles}(\mathsf{Zero})\ ) \\
\xrightarrow{\mathrm{i}}_{\mathcal{R}} \ & \mathsf{Cons}(\ \mathsf{S}(\mathsf{S}(\mathsf{d}(\mathsf{Zero})))\ , \mathsf{doubles}(\mathsf{Zero})) \\
\xrightarrow{\mathrm{i}}_{\mathcal{R}} \ & \mathsf{Cons}(\mathsf{S}(\mathsf{S}(\mathsf{Zero})), \mathsf{doubles}(\mathsf{Zero})) \\
\xrightarrow{\mathrm{i}}_{\mathcal{R}} \ & \mathsf{Cons}(\mathsf{S}(\mathsf{S}(\mathsf{Zero})),\ \mathsf{Nil}\ )
\end{aligned}$$

4 steps.

Reduction with **parallel-**innermost rewriting:

$$\mathsf{doubles}(\mathsf{S}(\mathsf{Zero}))$$

Parallel-innermost rewriting: How it works

TRS $\mathcal{R}$:

$$\mathsf{d}(\mathsf{Zero}) \to \mathsf{Zero}$$
$$\mathsf{d}(\mathsf{S}(x)) \to \mathsf{S}(\mathsf{S}(\mathsf{d}(x)))$$
$$\mathsf{doubles}(\mathsf{Zero}) \to \mathsf{Nil}$$
$$\mathsf{doubles}(\mathsf{S}(x)) \to \mathsf{Cons}(\mathsf{d}(\mathsf{S}(x)), \mathsf{doubles}(x))$$

Reduction with (sequential) innermost rewriting:

$$\begin{aligned} & \mathsf{doubles}(\mathsf{S}(\mathsf{Zero})) \\ \xrightarrow{\mathsf{i}}_{\mathcal{R}} \; & \mathsf{Cons}(\, \mathsf{d}(\mathsf{S}(\mathsf{Zero})) \,, \, \mathsf{doubles}(\mathsf{Zero}) \,) \\ \xrightarrow{\mathsf{i}}_{\mathcal{R}} \; & \mathsf{Cons}(\, \mathsf{S}(\mathsf{S}(\mathsf{d}(\mathsf{Zero}))) \,, \mathsf{doubles}(\mathsf{Zero})) \\ \xrightarrow{\mathsf{i}}_{\mathcal{R}} \; & \mathsf{Cons}(\mathsf{S}(\mathsf{S}(\mathsf{Zero})), \mathsf{doubles}(\mathsf{Zero})) \\ \xrightarrow{\mathsf{i}}_{\mathcal{R}} \; & \mathsf{Cons}(\mathsf{S}(\mathsf{S}(\mathsf{Zero})), \, \mathsf{Nil} \,) \end{aligned}$$

4 steps.

Reduction with **parallel-**innermost rewriting:

$$\begin{aligned} & \mathsf{doubles}(\mathsf{S}(\mathsf{Zero})) \\ \mathrel{\shortparallel}\!\xrightarrow{\mathsf{i}}_{\mathcal{R}} \; & \mathsf{Cons}(\, \mathsf{d}(\mathsf{S}(\mathsf{Zero})) \,, \, \mathsf{doubles}(\mathsf{Zero}) \,) \end{aligned}$$

Parallel-innermost rewriting: How it works

TRS $\mathcal{R}$:

$$\mathsf{d}(\mathsf{Zero}) \to \mathsf{Zero}$$
$$\mathsf{d}(\mathsf{S}(x)) \to \mathsf{S}(\mathsf{S}(\mathsf{d}(x)))$$
$$\mathsf{doubles}(\mathsf{Zero}) \to \mathsf{Nil}$$
$$\mathsf{doubles}(\mathsf{S}(x)) \to \mathsf{Cons}(\mathsf{d}(\mathsf{S}(x)), \mathsf{doubles}(x))$$

Reduction with (sequential) innermost rewriting:

$$\begin{aligned}
&\mathsf{doubles}(\mathsf{S}(\mathsf{Zero})) \\
\xrightarrow{\mathsf{i}}_{\mathcal{R}}\ &\mathsf{Cons}(\ \mathsf{d}(\mathsf{S}(\mathsf{Zero}))\ ,\ \mathsf{doubles}(\mathsf{Zero})\ ) \\
\xrightarrow{\mathsf{i}}_{\mathcal{R}}\ &\mathsf{Cons}(\ \mathsf{S}(\mathsf{S}(\mathsf{d}(\mathsf{Zero})))\ , \mathsf{doubles}(\mathsf{Zero})) \\
\xrightarrow{\mathsf{i}}_{\mathcal{R}}\ &\mathsf{Cons}(\mathsf{S}(\mathsf{S}(\mathsf{Zero})), \mathsf{doubles}(\mathsf{Zero})) \\
\xrightarrow{\mathsf{i}}_{\mathcal{R}}\ &\mathsf{Cons}(\mathsf{S}(\mathsf{S}(\mathsf{Zero})),\ \mathsf{Nil}\ )
\end{aligned}$$

4 steps.

Reduction with **parallel-**innermost rewriting:

$$\begin{aligned}
&\mathsf{doubles}(\mathsf{S}(\mathsf{Zero})) \\
\mathrel{\#}\!\!\xrightarrow{\mathsf{i}}_{\mathcal{R}}\ &\mathsf{Cons}(\ \mathsf{d}(\mathsf{S}(\mathsf{Zero}))\ ,\ \mathsf{doubles}(\mathsf{Zero})\ ) \\
\mathrel{\#}\!\!\xrightarrow{\mathsf{i}}_{\mathcal{R}}\ &\mathsf{Cons}(\ \mathsf{S}(\mathsf{S}(\mathsf{d}(\mathsf{Zero})))\ ,\ \mathsf{Nil}\ )
\end{aligned}$$

Parallel-innermost rewriting: How it works

TRS $\mathcal{R}$:

$$\mathsf{d}(\mathsf{Zero}) \to \mathsf{Zero}$$
$$\mathsf{d}(\mathsf{S}(x)) \to \mathsf{S}(\mathsf{S}(\mathsf{d}(x)))$$
$$\mathsf{doubles}(\mathsf{Zero}) \to \mathsf{Nil}$$
$$\mathsf{doubles}(\mathsf{S}(x)) \to \mathsf{Cons}(\mathsf{d}(\mathsf{S}(x)), \mathsf{doubles}(x))$$

Reduction with (sequential) innermost rewriting:

$$\begin{aligned}
& \mathsf{doubles}(\mathsf{S}(\mathsf{Zero})) \\
\xrightarrow{\mathsf{i}}_{\mathcal{R}} \; & \mathsf{Cons}(\,\mathsf{d}(\mathsf{S}(\mathsf{Zero}))\,,\,\mathsf{doubles}(\mathsf{Zero})\,) \\
\xrightarrow{\mathsf{i}}_{\mathcal{R}} \; & \mathsf{Cons}(\,\mathsf{S}(\mathsf{S}(\mathsf{d}(\mathsf{Zero})))\,, \mathsf{doubles}(\mathsf{Zero})) \\
\xrightarrow{\mathsf{i}}_{\mathcal{R}} \; & \mathsf{Cons}(\mathsf{S}(\mathsf{S}(\mathsf{Zero})), \mathsf{doubles}(\mathsf{Zero})) \\
\xrightarrow{\mathsf{i}}_{\mathcal{R}} \; & \mathsf{Cons}(\mathsf{S}(\mathsf{S}(\mathsf{Zero})),\,\mathsf{Nil}\,)
\end{aligned}$$

4 steps.

Reduction with **parallel-**innermost rewriting:

$$\begin{aligned}
& \mathsf{doubles}(\mathsf{S}(\mathsf{Zero})) \\
\parallel\!\xrightarrow{\mathsf{i}}_{\mathcal{R}} \; & \mathsf{Cons}(\,\mathsf{d}(\mathsf{S}(\mathsf{Zero}))\,,\,\mathsf{doubles}(\mathsf{Zero})\,) \\
\parallel\!\xrightarrow{\mathsf{i}}_{\mathcal{R}} \; & \mathsf{Cons}(\,\mathsf{S}(\mathsf{S}(\mathsf{d}(\mathsf{Zero})))\,,\,\mathsf{Nil}\,) \\
\parallel\!\xrightarrow{\mathsf{i}}_{\mathcal{R}} \; & \mathsf{Cons}(\mathsf{S}(\mathsf{S}(\mathsf{Zero})), \mathsf{Nil})
\end{aligned}$$

Parallel-innermost rewriting: How it works

TRS $\mathcal{R}$:

$$\mathsf{d}(\mathsf{Zero}) \to \mathsf{Zero}$$
$$\mathsf{d}(\mathsf{S}(x)) \to \mathsf{S}(\mathsf{S}(\mathsf{d}(x)))$$
$$\mathsf{doubles}(\mathsf{Zero}) \to \mathsf{Nil}$$
$$\mathsf{doubles}(\mathsf{S}(x)) \to \mathsf{Cons}(\mathsf{d}(\mathsf{S}(x)), \mathsf{doubles}(x))$$

Reduction with (sequential) innermost rewriting:

$$\begin{aligned} & \mathsf{doubles}(\mathsf{S}(\mathsf{Zero})) \\ \xrightarrow{\mathsf{i}}_{\mathcal{R}} \ & \mathsf{Cons}(\ \mathsf{d}(\mathsf{S}(\mathsf{Zero}))\ ,\ \mathsf{doubles}(\mathsf{Zero})\ ) \\ \xrightarrow{\mathsf{i}}_{\mathcal{R}} \ & \mathsf{Cons}(\ \mathsf{S}(\mathsf{S}(\mathsf{d}(\mathsf{Zero})))\ , \mathsf{doubles}(\mathsf{Zero})) \\ \xrightarrow{\mathsf{i}}_{\mathcal{R}} \ & \mathsf{Cons}(\mathsf{S}(\mathsf{S}(\mathsf{Zero})), \mathsf{doubles}(\mathsf{Zero})) \\ \xrightarrow{\mathsf{i}}_{\mathcal{R}} \ & \mathsf{Cons}(\mathsf{S}(\mathsf{S}(\mathsf{Zero})),\ \mathsf{Nil}\ ) \end{aligned}$$

4 steps.

Reduction with **parallel-**innermost rewriting:

$$\begin{aligned} & \mathsf{doubles}(\mathsf{S}(\mathsf{Zero})) \\ \overset{\mathsf{i}}{\Rightarrow}_{\mathcal{R}} \ & \mathsf{Cons}(\ \mathsf{d}(\mathsf{S}(\mathsf{Zero}))\ ,\ \mathsf{doubles}(\mathsf{Zero})\ ) \\ \overset{\mathsf{i}}{\Rightarrow}_{\mathcal{R}} \ & \mathsf{Cons}(\ \mathsf{S}(\mathsf{S}(\mathsf{d}(\mathsf{Zero})))\ ,\ \mathsf{Nil}\ ) \\ \overset{\mathsf{i}}{\Rightarrow}_{\mathcal{R}} \ & \mathsf{Cons}(\mathsf{S}(\mathsf{S}(\mathsf{Zero})), \mathsf{Nil}) \end{aligned}$$

Must reduce **all** innermost redexes. Only 3 steps!

Parallel-innermost rewriting: How it works

TRS $\mathcal{R}$:

$$\mathsf{d}(\mathsf{Zero}) \to \mathsf{Zero}$$
$$\mathsf{d}(\mathsf{S}(x)) \to \mathsf{S}(\mathsf{S}(\mathsf{d}(x)))$$
$$\mathsf{doubles}(\mathsf{Zero}) \to \mathsf{Nil}$$
$$\mathsf{doubles}(\mathsf{S}(x)) \to \mathsf{Cons}(\mathsf{d}(\mathsf{S}(x)), \mathsf{doubles}(x))$$

Reduction with (sequential) innermost rewriting:

$$\begin{aligned}
&\mathsf{doubles}(\mathsf{S}(\mathsf{Zero})) \\
\xrightarrow{\mathsf{i}}_{\mathcal{R}}\ &\mathsf{Cons}(\ \mathsf{d}(\mathsf{S}(\mathsf{Zero}))\ ,\ \mathsf{doubles}(\mathsf{Zero})\ ) \\
\xrightarrow{\mathsf{i}}_{\mathcal{R}}\ &\mathsf{Cons}(\ \mathsf{S}(\mathsf{S}(\mathsf{d}(\mathsf{Zero})))\ , \mathsf{doubles}(\mathsf{Zero})) \\
\xrightarrow{\mathsf{i}}_{\mathcal{R}}\ &\mathsf{Cons}(\mathsf{S}(\mathsf{S}(\mathsf{Zero})), \mathsf{doubles}(\mathsf{Zero})) \\
\xrightarrow{\mathsf{i}}_{\mathcal{R}}\ &\mathsf{Cons}(\mathsf{S}(\mathsf{S}(\mathsf{Zero})),\ \mathsf{Nil}\ )
\end{aligned}$$

4 steps.

Reduction with **parallel-**innermost rewriting:

$$\begin{aligned}
&\mathsf{doubles}(\mathsf{S}(\mathsf{Zero})) \\
\parallel\!\!\xrightarrow{\mathsf{i}}_{\mathcal{R}}\ &\mathsf{Cons}(\ \mathsf{d}(\mathsf{S}(\mathsf{Zero}))\ ,\ \mathsf{doubles}(\mathsf{Zero})\ ) \\
\parallel\!\!\xrightarrow{\mathsf{i}}_{\mathcal{R}}\ &\mathsf{Cons}(\ \mathsf{S}(\mathsf{S}(\mathsf{d}(\mathsf{Zero})))\ ,\ \mathsf{Nil}\ ) \\
\parallel\!\!\xrightarrow{\mathsf{i}}_{\mathcal{R}}\ &\mathsf{Cons}(\mathsf{S}(\mathsf{S}(\mathsf{Zero})), \mathsf{Nil})
\end{aligned}$$

Must reduce **all** innermost redexes. Only 3 steps!

Here: (max-)parallel-innermost rewriting – all innermost redexes **must** be reduced

Literature: also variants of parallel rewriting where one or more innermost redexes **may** be reduced

Proving confluence of parallel-innermost rewriting (1/2)

Proving **confluence** of $\Rightarrow^{\mathsf{i}}_{\mathcal{R}}$:

Proving confluence of parallel-innermost rewriting (1/2)

Proving **confluence** of $\parallel\!\!\rightarrow^{\mathsf{i}}_{\mathcal{R}}$:

Attempt 1:

- use a tool from CoCo to prove confluence of $\parallel\!\!\rightarrow^{\mathsf{i}}_{\mathcal{R}}$ directly

Proving confluence of parallel-innermost rewriting (1/2)

Proving **confluence** of $\parallel\!\!\rightarrow^{\mathsf{i}}_{\mathcal{R}}$:

Attempt 1:

- use a tool from CoCo to prove confluence of $\parallel\!\!\rightarrow^{\mathsf{i}}_{\mathcal{R}}$ directly

But: **parallel**-innermost rewriting seems not to be handled (yet!)

Proving confluence of parallel-innermost rewriting (1/2)

Proving **confluence** of $\parallel^{\mathsf{i}}\!\!\rightarrow_{\mathcal{R}}$:

Attempt 1:

- use a tool from CoCo to prove confluence of $\parallel^{\mathsf{i}}\!\!\rightarrow_{\mathcal{R}}$ directly

But: **parallel**-innermost rewriting seems not to be handled (yet!)

Attempt 2:

- use a tool from CoCo for full or innermost rewriting
- get sufficient criterion for confluence of $\parallel^{\mathsf{i}}\!\!\rightarrow_{\mathcal{R}}$

Proving confluence of parallel-innermost rewriting (1/2)

Proving **confluence** of $\parallel\!\!\rightarrow^{\mathsf{i}}_{\mathcal{R}}$:

Attempt 1:

- use a tool from CoCo to prove confluence of $\parallel\!\!\rightarrow^{\mathsf{i}}_{\mathcal{R}}$ directly

But: **parallel**-innermost rewriting seems not to be handled (yet!)

Attempt 2:

- use a tool from CoCo for full or innermost rewriting
- get sufficient criterion for confluence of $\parallel\!\!\rightarrow^{\mathsf{i}}_{\mathcal{R}}$

But:

Example (Full/innermost confluence does *not* imply parallel-innermost confluence)

$\mathcal{R} = \{ \quad \mathsf{a} \to \mathsf{f}(\mathsf{b},\mathsf{c}), \quad \mathsf{a} \to \mathsf{f}(\mathsf{b},\mathsf{b}), \quad \mathsf{b} \to \mathsf{c}, \quad \mathsf{c} \to \mathsf{b} \quad \}$. Confluent: $\rightarrow^{\mathsf{i}}_{\mathcal{R}}$ and $\rightarrow_{\mathcal{R}}$ But...

Proving confluence of parallel-innermost rewriting (1/2)

Proving **confluence** of $\parallel\!\!\rightarrow^{\mathsf{i}}_{\mathcal{R}}$:

Attempt 1:

- use a tool from CoCo to prove confluence of $\parallel\!\!\rightarrow^{\mathsf{i}}_{\mathcal{R}}$ directly

But: **parallel**-innermost rewriting seems not to be handled (yet!)

Attempt 2:

- use a tool from CoCo for full or innermost rewriting
- get sufficient criterion for confluence of $\parallel\!\!\rightarrow^{\mathsf{i}}_{\mathcal{R}}$

But:

Example (Full/innermost confluence does *not* imply parallel-innermost confluence)

$\mathcal{R} = \{ \quad \mathsf{a} \to \mathsf{f}(\mathsf{b},\mathsf{c}), \quad \mathsf{a} \to \mathsf{f}(\mathsf{b},\mathsf{b}), \quad \mathsf{b} \to \mathsf{c}, \quad \mathsf{c} \to \mathsf{b} \quad \}$. Confluent: $\rightarrow^{\mathsf{i}}_{\mathcal{R}}$ and $\rightarrow_{\mathcal{R}}$ But...

a

Proving confluence of parallel-innermost rewriting (1/2)

Proving **confluence** of $\parallel^{\mathsf{i}}\!\!\rightarrow_{\mathcal{R}}$:

Attempt 1:

- use a tool from CoCo to prove confluence of $\parallel^{\mathsf{i}}\!\!\rightarrow_{\mathcal{R}}$ directly

But: **parallel**-innermost rewriting seems not to be handled (yet!)

Attempt 2:

- use a tool from CoCo for full or innermost rewriting
- get sufficient criterion for confluence of $\parallel^{\mathsf{i}}\!\!\rightarrow_{\mathcal{R}}$

But:

Example (Full/innermost confluence does *not* imply parallel-innermost confluence)

$\mathcal{R} = \{ \quad \mathsf{a} \to \mathsf{f}(\mathsf{b},\mathsf{c}), \quad \mathsf{a} \to \mathsf{f}(\mathsf{b},\mathsf{b}), \quad \mathsf{b} \to \mathsf{c}, \quad \mathsf{c} \to \mathsf{b} \quad \}$. Confluent: $\xrightarrow{\mathsf{i}}_{\mathcal{R}}$ and $\rightarrow_{\mathcal{R}}$ But...

$\mathsf{a} \parallel^{\mathsf{i}}\!\!\rightarrow_{\mathcal{R}} \mathsf{f}(\mathsf{b},\mathsf{c})$

Proving confluence of parallel-innermost rewriting (1/2)

Proving **confluence** of $\parallel^{\mathsf{i}}\!\!\rightarrow_{\mathcal{R}}$:

Attempt 1:

- use a tool from CoCo to prove confluence of $\parallel^{\mathsf{i}}\!\!\rightarrow_{\mathcal{R}}$ directly

But: **parallel**-innermost rewriting seems not to be handled (yet!)

Attempt 2:

- use a tool from CoCo for full or innermost rewriting
- get sufficient criterion for confluence of $\parallel^{\mathsf{i}}\!\!\rightarrow_{\mathcal{R}}$

But:

Example (Full/innermost confluence does *not* imply parallel-innermost confluence)

$\mathcal{R} = \{ \quad \mathsf{a} \to \mathsf{f}(\mathsf{b},\mathsf{c}), \quad \mathsf{a} \to \mathsf{f}(\mathsf{b},\mathsf{b}), \quad \mathsf{b} \to \mathsf{c}, \quad \mathsf{c} \to \mathsf{b} \quad \}$. Confluent: $\xrightarrow{\mathsf{i}}_{\mathcal{R}}$ and $\rightarrow_{\mathcal{R}}$ But...

$$\mathsf{f}(\mathsf{b},\mathsf{b}) \; \mathcal{R}^{\leftarrow}\!\!\parallel^{\mathsf{i}} \; \mathsf{a} \; \parallel^{\mathsf{i}}\!\!\rightarrow_{\mathcal{R}} \mathsf{f}(\mathsf{b},\mathsf{c})$$

Proving confluence of parallel-innermost rewriting (1/2)

Proving **confluence** of $\parallel^{\mathsf{i}}\!\!\rightarrow_{\mathcal{R}}$:

Attempt 1:

- use a tool from CoCo to prove confluence of $\parallel^{\mathsf{i}}\!\!\rightarrow_{\mathcal{R}}$ directly

But: **parallel**-innermost rewriting seems not to be handled (yet!)

Attempt 2:

- use a tool from CoCo for full or innermost rewriting
- get sufficient criterion for confluence of $\parallel^{\mathsf{i}}\!\!\rightarrow_{\mathcal{R}}$

But:

Example (Full/innermost confluence does *not* imply parallel-innermost confluence)

$\mathcal{R} = \{ \quad \mathsf{a} \to \mathsf{f}(\mathsf{b},\mathsf{c}), \quad \mathsf{a} \to \mathsf{f}(\mathsf{b},\mathsf{b}), \quad \mathsf{b} \to \mathsf{c}, \quad \mathsf{c} \to \mathsf{b} \quad \}$. Confluent: $\xrightarrow{\mathsf{i}}_{\mathcal{R}}$ and $\rightarrow_{\mathcal{R}}$ But...

$\mathsf{f}(\mathsf{b},\mathsf{b}) \; \mathcal{R} \leftarrow \parallel^{\mathsf{i}} \; \mathsf{a} \; \parallel^{\mathsf{i}}\!\!\rightarrow_{\mathcal{R}} \mathsf{f}(\mathsf{b},\mathsf{c}) \; \parallel^{\mathsf{i}}\!\!\rightarrow_{\mathcal{R}} \mathsf{f}(\mathsf{c},\mathsf{b}) \; \parallel^{\mathsf{i}}\!\!\rightarrow_{\mathcal{R}} \mathsf{f}(\mathsf{b},\mathsf{c}) \; \parallel^{\mathsf{i}}\!\!\rightarrow_{\mathcal{R}} \ldots$

Proving confluence of parallel-innermost rewriting (1/2)

Proving **confluence** of $\parallel^{\mathsf{i}}\!\!\rightarrow_{\mathcal{R}}$:

Attempt 1:

- use a tool from CoCo to prove confluence of $\parallel^{\mathsf{i}}\!\!\rightarrow_{\mathcal{R}}$ directly

But: **parallel**-innermost rewriting seems not to be handled (yet!)

Attempt 2:

- use a tool from CoCo for full or innermost rewriting
- get sufficient criterion for confluence of $\parallel^{\mathsf{i}}\!\!\rightarrow_{\mathcal{R}}$

But:

Example (Full/innermost confluence does *not* imply parallel-innermost confluence)

$\mathcal{R} = \{ \quad \mathsf{a} \to \mathsf{f}(\mathsf{b},\mathsf{c}), \quad \mathsf{a} \to \mathsf{f}(\mathsf{b},\mathsf{b}), \quad \mathsf{b} \to \mathsf{c}, \quad \mathsf{c} \to \mathsf{b} \quad \}$. Confluent: $\xrightarrow{\mathsf{i}}_{\mathcal{R}}$ and $\rightarrow_{\mathcal{R}}$ But...

$\ldots \; \mathcal{R}\!\leftarrow\!\parallel^{\mathsf{i}} \; \mathsf{f}(\mathsf{b},\mathsf{b}) \; \mathcal{R}\!\leftarrow\!\parallel^{\mathsf{i}} \; \mathsf{f}(\mathsf{c},\mathsf{c}) \; \mathcal{R}\!\leftarrow\!\parallel^{\mathsf{i}} \; \mathsf{f}(\mathsf{b},\mathsf{b}) \; \mathcal{R}\!\leftarrow\!\parallel^{\mathsf{i}} \; \mathsf{a} \; \parallel^{\mathsf{i}}\!\!\rightarrow_{\mathcal{R}} \mathsf{f}(\mathsf{b},\mathsf{c}) \; \parallel^{\mathsf{i}}\!\!\rightarrow_{\mathcal{R}} \mathsf{f}(\mathsf{c},\mathsf{b}) \; \parallel^{\mathsf{i}}\!\!\rightarrow_{\mathcal{R}} \mathsf{f}(\mathsf{b},\mathsf{c}) \; \parallel^{\mathsf{i}}\!\!\rightarrow_{\mathcal{R}} \ldots$

Proving confluence of parallel-innermost rewriting (1/2)

Proving **confluence** of $\parallel^{\mathsf{i}}\!\!\rightarrow_{\mathcal{R}}$:

Attempt 1:

- use a tool from CoCo to prove confluence of $\parallel^{\mathsf{i}}\!\!\rightarrow_{\mathcal{R}}$ directly

But: **parallel**-innermost rewriting seems not to be handled (yet!)

Attempt 2:

- use a tool from CoCo for full or innermost rewriting
- get sufficient criterion for confluence of $\parallel^{\mathsf{i}}\!\!\rightarrow_{\mathcal{R}}$

But:

Example (Full/innermost confluence does *not* imply parallel-innermost confluence)

$\mathcal{R} = \{ \quad \mathsf{a} \to \mathsf{f}(\mathsf{b},\mathsf{c}), \quad \mathsf{a} \to \mathsf{f}(\mathsf{b},\mathsf{b}), \quad \mathsf{b} \to \mathsf{c}, \quad \mathsf{c} \to \mathsf{b} \quad \}$. Confluent: $\xrightarrow{\mathsf{i}}_{\mathcal{R}}$ and $\rightarrow_{\mathcal{R}}$ But...

$\ldots \; {}_{\mathcal{R}}\!\leftarrow\!\!\parallel^{\mathsf{i}} \; \mathsf{f}(\mathsf{b},\mathsf{b}) \; {}_{\mathcal{R}}\!\leftarrow\!\!\parallel^{\mathsf{i}} \; \mathsf{f}(\mathsf{c},\mathsf{c}) \; {}_{\mathcal{R}}\!\leftarrow\!\!\parallel^{\mathsf{i}} \; \mathsf{f}(\mathsf{b},\mathsf{b}) \; {}_{\mathcal{R}}\!\leftarrow\!\!\parallel^{\mathsf{i}} \; \mathsf{a} \; \parallel^{\mathsf{i}}\!\!\rightarrow_{\mathcal{R}} \mathsf{f}(\mathsf{b},\mathsf{c}) \; \parallel^{\mathsf{i}}\!\!\rightarrow_{\mathcal{R}} \mathsf{f}(\mathsf{c},\mathsf{b}) \; \parallel^{\mathsf{i}}\!\!\rightarrow_{\mathcal{R}} \mathsf{f}(\mathsf{b},\mathsf{c}) \; \parallel^{\mathsf{i}}\!\!\rightarrow_{\mathcal{R}} \ldots$

Conjecture

Let $\mathcal{R}$ be a TRS whose innermost rewrite relation $\xrightarrow{\mathsf{i}}_{\mathcal{R}}$ is terminating.
Then $\xrightarrow{\mathsf{i}}_{\mathcal{R}}$ is confluent iff $\parallel^{\mathsf{i}}\!\!\rightarrow_{\mathcal{R}}$ is confluent.

Proving confluence of parallel-innermost rewriting (2/2)

Proving **confluence** of $\Rightarrow^{\mathsf{i}}_{\mathcal{R}}$:

Proving confluence of parallel-innermost rewriting (2/2)

Proving **confluence** of $\Rightarrow^{\mathsf{i}}_{\mathcal{R}}$:

Attempt 3:

- devise and implement a simple sufficient criterion

Proving confluence of parallel-innermost rewriting (2/2)

Proving **confluence** of $\Rightarrow^{\mathsf{i}}_{\mathcal{R}}$:

Attempt 3:

- devise and implement a simple sufficient criterion

Lemma (Baader, Nipkow, *Term Rewriting and All That*, 1998; Lemma 6.3.9)

If $\mathcal{R}$ is non-overlapping, rewriting any redex will have a unique result.

Proving confluence of parallel-innermost rewriting (2/2)

Proving **confluence** of $\Rightarrow^{\mathsf{i}}_{\mathcal{R}}$:

Attempt 3:

- devise and implement a simple sufficient criterion

Lemma (Baader, Nipkow, *Term Rewriting and All That*, 1998; Lemma 6.3.9)

If $\mathcal{R}$ is non-overlapping, rewriting any redex will have a unique result.

$\Rightarrow^{\mathsf{i}}_{\mathcal{R}}$ fixes the used redexes: all innermost redexes at the same time

Proving confluence of parallel-innermost rewriting (2/2)

Proving **confluence** of $\Rightarrow^{\mathsf{i}}_{\mathcal{R}}$:

Attempt 3:

- devise and implement a simple sufficient criterion

Lemma (Baader, Nipkow, *Term Rewriting and All That*, 1998; Lemma 6.3.9)

If $\mathcal{R}$ is non-overlapping, rewriting any redex will have a unique result.

$\Rightarrow^{\mathsf{i}}_{\mathcal{R}}$ fixes the used redexes: all innermost redexes at the same time

Corollary (Baudon, Fuhs, Gonnord, LOPSTR 2022)

If $\mathcal{R}$ is non-overlapping, $\Rightarrow^{\mathsf{i}}_{\mathcal{R}}$ is confluent.

Proving confluence of parallel-innermost rewriting (2/2)

Proving **confluence** of $\Rightarrow^{\mathsf{i}}_{\mathcal{R}}$:

Attempt 3:

- devise and implement a simple sufficient criterion

Lemma (Baader, Nipkow, *Term Rewriting and All That*, 1998; Lemma 6.3.9)

If $\mathcal{R}$ is non-overlapping, rewriting any redex will have a unique result.

$\Rightarrow^{\mathsf{i}}_{\mathcal{R}}$ fixes the used redexes: all innermost redexes at the same time

Corollary (Baudon, Fuhs, Gonnord, LOPSTR 2022)

If $\mathcal{R}$ is non-overlapping, $\Rightarrow^{\mathsf{i}}_{\mathcal{R}}$ is confluent.

Only a simple criterion: if $\mathcal{R}$ is non-overlapping, $\Rightarrow^{\mathsf{i}}_{\mathcal{R}}$ is deterministic!

Proving confluence of parallel-innermost rewriting (2/2)

Proving **confluence** of $\Rightarrow^{\mathsf{i}}_{\mathcal{R}}$:

Attempt 3:

- devise and implement a simple sufficient criterion

Lemma (Baader, Nipkow, *Term Rewriting and All That*, 1998; Lemma 6.3.9)

If $\mathcal{R}$ is non-overlapping, rewriting any redex will have a unique result.

$\Rightarrow^{\mathsf{i}}_{\mathcal{R}}$ fixes the used redexes: all innermost redexes at the same time

Corollary (Baudon, Fuhs, Gonnord, LOPSTR 2022)

If $\mathcal{R}$ is non-overlapping, $\Rightarrow^{\mathsf{i}}_{\mathcal{R}}$ is confluent.

Only a simple criterion: if $\mathcal{R}$ is non-overlapping, $\Rightarrow^{\mathsf{i}}_{\mathcal{R}}$ is deterministic!

Applies also to **parallel-outermost** rewriting (and other multistep strategies with fixed redexes)

Experiments

How powerful is our criterion for confluence of $\#\!\!\#^{\mathsf{i}}_{\mathcal{R}}$ in practice?

Experiments

How powerful is our criterion for confluence of $\parallel^{\mathsf{i}}\!\!\rightarrow_{\mathcal{R}}$ in practice?

- Implementation in program analysis tool AProVE

Experiments

How powerful is our criterion for confluence of $\not\parallel^{i}_{\mathcal{R}}$ in practice?

- Implementation in program analysis tool AProVE
- Benchmark set 1:
  663 TRSs from Termination Problem Database (TPDB),
  category `Runtime_Complexity_Innermost_Rewriting`

Experiments

How powerful is our criterion for confluence of $\#\!\!\#^{\mathsf{i}}\!\!\rightarrow_{\mathcal{R}}$ in practice?

- Implementation in program analysis tool AProVE
- Benchmark set 1:
  663 TRSs from Termination Problem Database (TPDB),
  category `Runtime_Complexity_Innermost_Rewriting`

  $\Rightarrow$ proof of confluence of $\#\!\!\#^{\mathsf{i}}\!\!\rightarrow_{\mathcal{R}}$ found for 447 TRSs $\mathcal{R}$: 67.4%

Experiments

How powerful is our criterion for confluence of $\#\!\!\#^{\mathsf{i}}\!\!\rightarrow_{\mathcal{R}}$ in practice?

- Implementation in program analysis tool AProVE
- Benchmark set 1:
  663 TRSs from Termination Problem Database (TPDB),
  category `Runtime_Complexity_Innermost_Rewriting`

  $\Rightarrow$ proof of confluence of $\#\!\!\#^{\mathsf{i}}\!\!\rightarrow_{\mathcal{R}}$ found for 447 TRSs $\mathcal{R}$: 67.4%
- Benchmark set 2:
  570 unsorted unconditional TRSs from COPS

Experiments

How powerful is our criterion for confluence of $\#\!\!\#^{\mathsf{i}}\!\!\rightarrow_{\mathcal{R}}$ in practice?

- Implementation in program analysis tool AProVE
- Benchmark set 1:
  663 TRSs from Termination Problem Database (TPDB),
  category `Runtime_Complexity_Innermost_Rewriting`

  $\Rightarrow$ proof of confluence of $\#\!\!\#^{\mathsf{i}}\!\!\rightarrow_{\mathcal{R}}$ found for 447 TRSs $\mathcal{R}$: 67.4%
- Benchmark set 2:
  570 unsorted unconditional TRSs from COPS

  $\Rightarrow$ proof of confluence of $\#\!\!\#^{\mathsf{i}}\!\!\rightarrow_{\mathcal{R}}$ found for 115 TRSs $\mathcal{R}$: 20.2%

Experiments

How powerful is our criterion for confluence of $\#\!\!\#^{\mathsf{i}}\!\!\rightarrow_{\mathcal{R}}$ in practice?

- Implementation in program analysis tool AProVE
- Benchmark set 1:
  663 TRSs from Termination Problem Database (TPDB),
  category `Runtime_Complexity_Innermost_Rewriting`

  $\Rightarrow$ proof of confluence of $\#\!\!\#^{\mathsf{i}}\!\!\rightarrow_{\mathcal{R}}$ found for 447 TRSs $\mathcal{R}$: 67.4%
- Benchmark set 2:
  570 unsorted unconditional TRSs from COPS

  $\Rightarrow$ proof of confluence of $\#\!\!\#^{\mathsf{i}}\!\!\rightarrow_{\mathcal{R}}$ found for 115 TRSs $\mathcal{R}$: 20.2%

Can we not do better than this?

Future Work

- Adapt criteria for full rewriting (or specific rewrite strategies)
- Specifically, adapt Knuth-Bendix criterion: prove termination and joinability of critical pairs

Future Work

- Adapt criteria for full rewriting (or specific rewrite strategies)
- Specifically, adapt Knuth-Bendix criterion: prove termination and joinability of critical pairs

Sayaka Ishizuki, Michio Oyamaguchi, Masahiko Sakai
Conditions for confluence of innermost terminating term rewriting systems
Appl. Algebra Eng. Commun. Comput. 30(4): 349-360 (2019)

Future Work

- Adapt criteria for full rewriting (or specific rewrite strategies)
- Specifically, adapt Knuth-Bendix criterion: prove termination and joinability of critical pairs

Sayaka Ishizuki, Michio Oyamaguchi, Masahiko Sakai
Conditions for confluence of innermost terminating term rewriting systems
Appl. Algebra Eng. Commun. Comput. 30(4): 349-360 (2019)

Lemma (Ishizuki, Oyamaguchi, Sakai, AAECC, Lemma 5)

If $\xrightarrow{\mathrm{i}}_{\mathcal{R}}$ is terminating and all critical pairs are innermost joinable ***for normalised instances****, then $\xrightarrow{\mathrm{i}}_{\mathcal{R}}$ is* ***confluent****.*

Future Work

- Adapt criteria for full rewriting (or specific rewrite strategies)
- Specifically, adapt Knuth-Bendix criterion: prove termination and joinability of critical pairs

Sayaka Ishizuki, Michio Oyamaguchi, Masahiko Sakai
Conditions for confluence of innermost terminating term rewriting systems
Appl. Algebra Eng. Commun. Comput. 30(4): 349-360 (2019)

Lemma (Ishizuki, Oyamaguchi, Sakai, AAECC, Lemma 5)

If $\xrightarrow{\mathrm{i}}_{\mathcal{R}}$ is terminating and all critical pairs are innermost joinable ***for normalised instances****, then $\xrightarrow{\mathrm{i}}_{\mathcal{R}}$ is* ***confluent****.*

If the conjecture (Attempt 2) holds, we have another criterion for confluence of $\overset{\mathrm{i}}{\Longrightarrow}_{\mathcal{R}}$!

Conclusion for "Proving confluence of parallel-innermost rewriting: how?"

- First (to the best of our knowledge) automated technique to prove confluence of **parallel-**innermost rewriting

Conclusion for "Proving confluence of parallel-innermost rewriting: how?"

- First (to the best of our knowledge) automated technique to prove confluence of **parallel-**innermost rewriting
- Works well for our application (motivation: deterministic inputs)

Conclusion for "Proving confluence of parallel-innermost rewriting: how?"

- First (to the best of our knowledge) automated technique to prove confluence of **parallel-**innermost rewriting
- Works well for our application (motivation: deterministic inputs)
- Works less well for TRSs from COPS

Conclusion for "Proving confluence of parallel-innermost rewriting: how?"

- First (to the best of our knowledge) automated technique to prove confluence of **parallel-**innermost rewriting
- Works well for our application (motivation: deterministic inputs)
- Works less well for TRSs from COPS
- Evaluation page with details of experiments:

https://www.dcs.bbk.ac.uk/~carsten/eval/parallel_confluence/

Motivation: Complexity analysis for automated scheduling (1/2)

What applications do confluence proofs for $\not\parallel^{\mathsf{i}}_{\mathcal{R}}$ have?

Motivation: Complexity analysis for automated scheduling (1/2)

What applications do confluence proofs for $\parallel^{\mathsf{i}}_{\mathcal{R}}$ have?

Goal: use complexity analysis to improve compilers

Inspiration

- Christophe Alias, Alain Darte, Paul Feautrier, Laure Gonnord
  Multi-dimensional Rankings, Program Termination, and Complexity Bounds of Flowchart Programs
  In *Proc. 17th Static Analysis Symposium*, pages 117–133, 2010.

Motivation: Complexity analysis for automated scheduling (1/2)

What applications do confluence proofs for $\#^{\mathsf{i}}\!\!\rightarrow_{\mathcal{R}}$ have?

Goal: use complexity analysis to improve compilers

Inspiration

- Christophe Alias, Alain Darte, Paul Feautrier, Laure Gonnord
  Multi-dimensional Rankings, Program Termination, and Complexity Bounds of Flowchart Programs
  In *Proc. 17th Static Analysis Symposium*, pages 117–133, 2010.
- Adapts static scheduling/parallelisation technique to find ranking functions ($\approx$ polynomial interpretations) for termination and complexity analysis

Motivation: Complexity analysis for automated scheduling (1/2)

What applications do confluence proofs for $\not\parallel^{\mathsf{i}}_{\mathcal{R}}$ have?

Goal: use complexity analysis to improve compilers

Inspiration

- Christophe Alias, Alain Darte, Paul Feautrier, Laure Gonnord
  Multi-dimensional Rankings, Program Termination, and Complexity Bounds of Flowchart Programs
  In *Proc. 17th Static Analysis Symposium*, pages 117–133, 2010.
- Adapts static scheduling/parallelisation technique to find ranking functions ($\approx$ polynomial interpretations) for termination and complexity analysis
- Idea: schedule assigns symbolic timestamps to instructions counting **up** from start of program

Motivation: Complexity analysis for automated scheduling (1/2)

What applications do confluence proofs for $\not\parallel^{\mathsf{i}}_{\mathcal{R}}$ have?

Goal: use complexity analysis to improve compilers

Inspiration

- Christophe Alias, Alain Darte, Paul Feautrier, Laure Gonnord
  Multi-dimensional Rankings, Program Termination, and Complexity Bounds of Flowchart Programs
  In *Proc. 17th Static Analysis Symposium*, pages 117–133, 2010.
- Adapts static scheduling/parallelisation technique to find ranking functions ($\approx$ polynomial interpretations) for termination and complexity analysis
- Idea: schedule assigns symbolic timestamps to instructions counting **up** from start of program
- Ranking function measures program states by number of steps until termination, counts **down** $\rightarrow$ get ranking function from schedule

Motivation: Complexity analysis for automated scheduling (1/2)

What applications do confluence proofs for $\mathcal{R}$ have?

Goal: use complexity analysis to improve compilers

Inspiration

- Christophe Alias, Alain Darte, Paul Feautrier, Laure Gonnord
  Multi-dimensional Rankings, Program Termination, and Complexity Bounds of Flowchart Programs
  In *Proc. 17th Static Analysis Symposium*, pages 117–133, 2010.
- Adapts static scheduling/parallelisation technique to find ranking functions ($\approx$ polynomial interpretations) for termination and complexity analysis
- Idea: schedule assigns symbolic timestamps to instructions counting **up** from start of program
- Ranking function measures program states by number of steps until termination, counts **down** $\rightarrow$ get ranking function from schedule
- But: restricted to programs with integer arithmetic and arrays

Motivation: Complexity analysis for automated scheduling (2/2)

How about data structures like lists or trees?

Motivation: Complexity analysis for automated scheduling (2/2)

How about data structures like lists or trees?

- We **know** how to find termination proofs and complexity bounds for TRSs...

Motivation: Complexity analysis for automated scheduling (2/2)

How about data structures like lists or trees?

- We **know** how to find termination proofs and complexity bounds for TRSs...
- Want: static scheduling for TRSs (goal: languages with pattern matching)

Motivation: Complexity analysis for automated scheduling (2/2)

How about data structures like lists or trees?

- We **know** how to find termination proofs and complexity bounds for TRSs...
- Want: static scheduling for TRSs (goal: languages with pattern matching)
- Get complexity bound for TRS, schedule function calls on data structures accordingly

Motivation: Complexity analysis for automated scheduling (2/2)

How about data structures like lists or trees?

- We **know** how to find termination proofs and complexity bounds for TRSs...
- Want: static scheduling for TRSs (goal: languages with pattern matching)
- Get complexity bound for TRS, schedule function calls on data structures accordingly

Complexity bound for which rewrite relation?!

Motivation: Complexity analysis for automated scheduling (2/2)

How about data structures like lists or trees?

- We **know** how to find termination proofs and complexity bounds for TRSs...
- Want: static scheduling for TRSs (goal: languages with pattern matching)
- Get complexity bound for TRS, schedule function calls on data structures accordingly

Complexity bound for which rewrite relation?!

- **Parallel-innermost** rewriting evaluates all innermost redexes simultaneously

Motivation: Complexity analysis for automated scheduling (2/2)

How about data structures like lists or trees?

- We **know** how to find termination proofs and complexity bounds for TRSs...
- Want: static scheduling for TRSs (goal: languages with pattern matching)
- Get complexity bound for TRS, schedule function calls on data structures accordingly

Complexity bound for which rewrite relation?!

- **Parallel-innermost** rewriting evaluates all innermost redexes simultaneously
- Captures which redexes can be evaluated independently by call-by-value strategy

Motivation: Complexity analysis for automated scheduling (2/2)

How about data structures like lists or trees?

- We **know** how to find termination proofs and complexity bounds for TRSs...
- Want: static scheduling for TRSs (goal: languages with pattern matching)
- Get complexity bound for TRS, schedule function calls on data structures accordingly

Complexity bound for which rewrite relation?!

- **Parallel-innermost** rewriting evaluates all innermost redexes simultaneously
- Captures which redexes can be evaluated independently by call-by-value strategy
- Standard assumption in parallel computing: machine with unbounded parallelism
  $\Rightarrow$ assess **potential** for parallelism

Runtime complexity of parallel-innermost rewriting

TRS $\mathcal{R}$:

$$\mathsf{d}(\mathsf{Zero}) \to \mathsf{Zero}$$
$$\mathsf{d}(\mathsf{S}(x)) \to \mathsf{S}(\mathsf{S}(\mathsf{d}(x)))$$
$$\mathsf{doubles}(\mathsf{Zero}) \to \mathsf{Nil}$$
$$\mathsf{doubles}(\mathsf{S}(x)) \to \mathsf{Cons}(\mathsf{d}(\mathsf{S}(x)), \mathsf{doubles}(x))$$

Reduction with (sequential) innermost rewriting:

$$\begin{aligned}
&\mathsf{doubles}(\mathsf{S}(\mathsf{Zero})) \\
\xrightarrow{\mathsf{i}}_{\mathcal{R}}\ &\mathsf{Cons}(\mathsf{d}(\mathsf{S}(\mathsf{Zero})), \mathsf{doubles}(\mathsf{Zero})) \\
\xrightarrow{\mathsf{i}}_{\mathcal{R}}\ &\mathsf{Cons}(\mathsf{S}(\mathsf{S}(\mathsf{d}(\mathsf{Zero}))), \mathsf{doubles}(\mathsf{Zero})) \\
\xrightarrow{\mathsf{i}}_{\mathcal{R}}\ &\mathsf{Cons}(\mathsf{S}(\mathsf{S}(\mathsf{Zero})), \mathsf{doubles}(\mathsf{Zero})) \\
\xrightarrow{\mathsf{i}}_{\mathcal{R}}\ &\mathsf{Cons}(\mathsf{S}(\mathsf{S}(\mathsf{Zero})), \mathsf{Nil})
\end{aligned}$$

4 steps.

Reduction with **parallel-**innermost rewriting:

$$\begin{aligned}
&\mathsf{doubles}(\mathsf{S}(\mathsf{Zero})) \\
\parallel\!\!\xrightarrow{\mathsf{i}}_{\mathcal{R}}\ &\mathsf{Cons}(\mathsf{d}(\mathsf{S}(\mathsf{Zero})), \mathsf{doubles}(\mathsf{Zero})) \\
\parallel\!\!\xrightarrow{\mathsf{i}}_{\mathcal{R}}\ &\mathsf{Cons}(\mathsf{S}(\mathsf{S}(\mathsf{d}(\mathsf{Zero}))), \mathsf{Nil}) \\
\parallel\!\!\xrightarrow{\mathsf{i}}_{\mathcal{R}}\ &\mathsf{Cons}(\mathsf{S}(\mathsf{S}(\mathsf{Zero})), \mathsf{Nil})
\end{aligned}$$

Must reduce **all** innermost redexes. Only 3 steps!

Runtime complexity of parallel-innermost rewriting

TRS $\mathcal{R}$:

$$\mathsf{d}(\mathsf{Zero}) \to \mathsf{Zero}$$
$$\mathsf{d}(\mathsf{S}(x)) \to \mathsf{S}(\mathsf{S}(\mathsf{d}(x)))$$
$$\mathsf{doubles}(\mathsf{Zero}) \to \mathsf{Nil}$$
$$\mathsf{doubles}(\mathsf{S}(x)) \to \mathsf{Cons}(\mathsf{d}(\mathsf{S}(x)), \mathsf{doubles}(x))$$

Reduction with (sequential) innermost rewriting:

$$\begin{aligned} & \mathsf{doubles}(\mathsf{S}(\mathsf{Zero})) \\ \xrightarrow{\mathsf{i}}_{\mathcal{R}} \ & \mathsf{Cons}(\ \mathsf{d}(\mathsf{S}(\mathsf{Zero}))\ ,\ \mathsf{doubles}(\mathsf{Zero})\ ) \\ \xrightarrow{\mathsf{i}}_{\mathcal{R}} \ & \mathsf{Cons}(\ \mathsf{S}(\mathsf{S}(\mathsf{d}(\mathsf{Zero})))\ , \mathsf{doubles}(\mathsf{Zero})) \\ \xrightarrow{\mathsf{i}}_{\mathcal{R}} \ & \mathsf{Cons}(\mathsf{S}(\mathsf{S}(\mathsf{Zero})), \mathsf{doubles}(\mathsf{Zero})) \\ \xrightarrow{\mathsf{i}}_{\mathcal{R}} \ & \mathsf{Cons}(\mathsf{S}(\mathsf{S}(\mathsf{Zero})),\ \mathsf{Nil}\ ) \end{aligned}$$

4 steps.

Reduction with **parallel-**innermost rewriting:

$$\begin{aligned} & \mathsf{doubles}(\mathsf{S}(\mathsf{Zero})) \\ \parallel\!\!\xrightarrow{\mathsf{i}}_{\mathcal{R}} \ & \mathsf{Cons}(\ \mathsf{d}(\mathsf{S}(\mathsf{Zero}))\ ,\ \mathsf{doubles}(\mathsf{Zero})\ ) \\ \parallel\!\!\xrightarrow{\mathsf{i}}_{\mathcal{R}} \ & \mathsf{Cons}(\ \mathsf{S}(\mathsf{S}(\mathsf{d}(\mathsf{Zero})))\ ,\ \mathsf{Nil}\ ) \\ \parallel\!\!\xrightarrow{\mathsf{i}}_{\mathcal{R}} \ & \mathsf{Cons}(\mathsf{S}(\mathsf{S}(\mathsf{Zero})), \mathsf{Nil}) \end{aligned}$$

Must reduce **all** innermost redexes. Only 3 steps!

Runtime Complexity: How long can a reduction from a **basic term** of size $\leq n$ get? (worst case)
Basic term: $f(t_1, \ldots, t_n)$ with f a defined symbol, t_i constructor terms

Sequential complexity: Example 1

TRS $\mathcal{R}$:

$$\mathsf{d}(\mathsf{Zero}) \to \mathsf{Zero}$$
$$\mathsf{d}(\mathsf{S}(x)) \to \mathsf{S}(\mathsf{S}(\mathsf{d}(x)))$$
$$\mathsf{doubles}(\mathsf{Zero}) \to \mathsf{Nil}$$
$$\mathsf{doubles}(\mathsf{S}(x)) \to \mathsf{Cons}(\mathsf{d}(\mathsf{S}(x)), \mathsf{doubles}(x))$$

Sequential complexity: Example 1

TRS $\mathcal{R}$:

$$\mathsf{d}(\mathsf{Zero}) \to \mathsf{Zero}$$
$$\mathsf{d}(\mathsf{S}(x)) \to \mathsf{S}(\mathsf{S}(\mathsf{d}(x)))$$
$$\mathsf{doubles}(\mathsf{Zero}) \to \mathsf{Nil}$$
$$\mathsf{doubles}(\mathsf{S}(x)) \to \mathsf{Cons}(\mathsf{d}(\mathsf{S}(x)), \mathsf{doubles}(x))$$

$\Rightarrow$ sum up costs of all function calls of a rule **together**

Sequential complexity: Example 1

TRS $\mathcal{R}$:

$$\mathsf{d}(\mathsf{Zero}) \to \mathsf{Zero}$$
$$\mathsf{d}(\mathsf{S}(x)) \to \mathsf{S}(\mathsf{S}(\mathsf{d}(x)))$$
$$\mathsf{doubles}(\mathsf{Zero}) \to \mathsf{Nil}$$
$$\mathsf{doubles}(\mathsf{S}(x)) \to \mathsf{Cons}(\mathsf{d}(\mathsf{S}(x)), \mathsf{doubles}(x))$$

Dependency Tuples $\mathbf{DT}(\mathcal{R})$ for function calls:

$$\mathsf{d}^\sharp(\mathsf{Zero}) \to \mathsf{Com}_0$$
$$\mathsf{d}^\sharp(\mathsf{S}(x)) \to \mathsf{Com}_1(\mathsf{d}^\sharp(x))$$
$$\mathsf{doubles}^\sharp(\mathsf{Zero}) \to \mathsf{Com}_0$$
$$\mathsf{doubles}^\sharp(\mathsf{S}(x)) \to \mathsf{Com}_2(\mathsf{d}^\sharp(\mathsf{S}(x)), \mathsf{doubles}^\sharp(x))$$

$\Rightarrow$ sum up costs of all function calls of a rule **together**

$$\mathrm{cost}(\mathsf{doubles}^\sharp(\mathsf{S}(x))) = 1 + \mathrm{cost}(\mathsf{d}^\sharp(\mathsf{S}(x))) + \mathrm{cost}(\mathsf{doubles}^\sharp(x))$$

Sequential complexity: Example 1

TRS $\mathcal{R}$:

$$\mathsf{d}(\mathsf{Zero}) \to \mathsf{Zero}$$
$$\mathsf{d}(\mathsf{S}(x)) \to \mathsf{S}(\mathsf{S}(\mathsf{d}(x)))$$
$$\mathsf{doubles}(\mathsf{Zero}) \to \mathsf{Nil}$$
$$\mathsf{doubles}(\mathsf{S}(x)) \to \mathsf{Cons}(\mathsf{d}(\mathsf{S}(x)), \mathsf{doubles}(x))$$

Dependency Tuples $\mathbf{DT}(\mathcal{R})$ for function calls:

$$\mathsf{d}^\sharp(\mathsf{Zero}) \to \mathsf{Com}_0$$
$$\mathsf{d}^\sharp(\mathsf{S}(x)) \to \mathsf{Com}_1(\mathsf{d}^\sharp(x))$$
$$\mathsf{doubles}^\sharp(\mathsf{Zero}) \to \mathsf{Com}_0$$
$$\mathsf{doubles}^\sharp(\mathsf{S}(x)) \to \mathsf{Com}_2(\mathsf{d}^\sharp(\mathsf{S}(x)), \mathsf{doubles}^\sharp(x))$$

$\Rightarrow$ sum up costs of all function calls of a rule **together**

$$\mathrm{cost}(\mathsf{doubles}^\sharp(\mathsf{S}(x))) = 1 + \mathrm{cost}(\mathsf{d}^\sharp(\mathsf{S}(x))) + \mathrm{cost}(\mathsf{doubles}^\sharp(x))$$

Theorem (Noschinski, Emmes, Giesl, J. Autom. Reasoning 2013)

Innermost complexity of $\mathcal{R}$ (aka $\mathbf{irc}_\mathcal{R}$) $\leq$ innermost complexity for DT problem for $\mathbf{DT}(\mathcal{R})$.

Sequential complexity: Example 1

TRS $\mathcal{R}$:

$$\mathsf{d}(\mathsf{Zero}) \to \mathsf{Zero}$$
$$\mathsf{d}(\mathsf{S}(x)) \to \mathsf{S}(\mathsf{S}(\mathsf{d}(x)))$$
$$\mathsf{doubles}(\mathsf{Zero}) \to \mathsf{Nil}$$
$$\mathsf{doubles}(\mathsf{S}(x)) \to \mathsf{Cons}(\mathsf{d}(\mathsf{S}(x)), \mathsf{doubles}(x))$$

Dependency Tuples $\mathbf{DT}(\mathcal{R})$ for function calls:

$$\mathsf{d}^\sharp(\mathsf{Zero}) \to \mathsf{Com}_0$$
$$\mathsf{d}^\sharp(\mathsf{S}(x)) \to \mathsf{Com}_1(\mathsf{d}^\sharp(x))$$
$$\mathsf{doubles}^\sharp(\mathsf{Zero}) \to \mathsf{Com}_0$$
$$\mathsf{doubles}^\sharp(\mathsf{S}(x)) \to \mathsf{Com}_2(\mathsf{d}^\sharp(\mathsf{S}(x)), \mathsf{doubles}^\sharp(x))$$

$\Rightarrow$ sum up costs of all function calls of a rule **together**

$$\mathrm{cost}(\mathsf{doubles}^\sharp(\mathsf{S}(x))) \;=\; 1 + \mathrm{cost}(\mathsf{d}^\sharp(\mathsf{S}(x))) + \mathrm{cost}(\mathsf{doubles}^\sharp(x))$$

Theorem (Noschinski, Emmes, Giesl, J. Autom. Reasoning 2013)

Innermost complexity of $\mathcal{R}$ (aka $\mathbf{irc}_\mathcal{R}$) $\leq$ innermost complexity for DT problem for $\mathbf{DT}(\mathcal{R})$.

$\Rightarrow$ DT framework finds (sequential) $\mathbf{irc}_\mathcal{R}(n) \in \mathcal{O}(n^2)$ via polynomial interpretation $\mathcal{P}ol$ of degree 2:

$$\mathcal{P}ol(\mathsf{doubles}^\sharp(x)) = x^2 + 2x, \quad \mathcal{P}ol(\mathsf{d}^\sharp(x)) = x, \quad \mathcal{P}ol(\mathsf{S}(x)) = 1 + x, \quad \ldots$$

Parallel-innermost complexity: Example 1

TRS $\mathcal{R}$:

$$\mathsf{d}(\mathsf{Zero}) \to \mathsf{Zero}$$
$$\mathsf{d}(\mathsf{S}(x)) \to \mathsf{S}(\mathsf{S}(\mathsf{d}(x)))$$
$$\mathsf{doubles}(\mathsf{Zero}) \to \mathsf{Nil}$$
$$\mathsf{doubles}(\mathsf{S}(x)) \to \mathsf{Cons}(\mathsf{d}(\mathsf{S}(x)), \mathsf{doubles}(x))$$

Parallel-innermost complexity: Example 1

TRS $\mathcal{R}$:

$$\mathsf{d}(\mathsf{Zero}) \to \mathsf{Zero}$$
$$\mathsf{d}(\mathsf{S}(x)) \to \mathsf{S}(\mathsf{S}(\mathsf{d}(x)))$$
$$\mathsf{doubles}(\mathsf{Zero}) \to \mathsf{Nil}$$
$$\mathsf{doubles}(\mathsf{S}(x)) \to \mathsf{Cons}(\mathsf{d}(\mathsf{S}(x)), \mathsf{doubles}(x))$$

$\Rightarrow$ function calls for last rule **in parallel**

Parallel-innermost complexity: Example 1

TRS $\mathcal{R}$:

$$\mathsf{d}(\mathsf{Zero}) \to \mathsf{Zero}$$
$$\mathsf{d}(\mathsf{S}(x)) \to \mathsf{S}(\mathsf{S}(\mathsf{d}(x)))$$
$$\mathsf{doubles}(\mathsf{Zero}) \to \mathsf{Nil}$$
$$\mathsf{doubles}(\mathsf{S}(x)) \to \mathsf{Cons}(\mathsf{d}(\mathsf{S}(x)), \mathsf{doubles}(x))$$

$\Rightarrow$ function calls for last rule **in parallel**
$\Rightarrow$ consider them **separately**, get the maximum of the costs

Parallel-innermost complexity: Example 1

TRS $\mathcal{R}$:

$$\mathsf{d}(\mathsf{Zero}) \to \mathsf{Zero}$$
$$\mathsf{d}(\mathsf{S}(x)) \to \mathsf{S}(\mathsf{S}(\mathsf{d}(x)))$$
$$\mathsf{doubles}(\mathsf{Zero}) \to \mathsf{Nil}$$
$$\mathsf{doubles}(\mathsf{S}(x)) \to \mathsf{Cons}(\mathsf{d}(\mathsf{S}(x)), \mathsf{doubles}(x))$$

Parallel Dependency Tuples PDT$(\mathcal{R})$ for **independent** function calls:

$$\mathsf{d}^\sharp(\mathsf{Zero}) \to \mathsf{Com}_0$$
$$\mathsf{d}^\sharp(\mathsf{S}(x)) \to \mathsf{Com}_1(\mathsf{d}^\sharp(x))$$
$$\mathsf{doubles}^\sharp(\mathsf{Zero}) \to \mathsf{Com}_0$$
$$\mathsf{doubles}^\sharp(\mathsf{S}(x)) \to \mathsf{Com}_1(\mathsf{d}^\sharp(\mathsf{S}(x)))$$
$$\mathsf{doubles}^\sharp(\mathsf{S}(x)) \to \mathsf{Com}_1(\mathsf{doubles}^\sharp(x))$$

$\Rightarrow$ function calls for last rule **in parallel**

$\Rightarrow$ consider them **separately**, get the maximum of the costs

Parallel-innermost complexity: Example 1

TRS $\mathcal{R}$:

$$\mathsf{d}(\mathsf{Zero}) \to \mathsf{Zero}$$
$$\mathsf{d}(\mathsf{S}(x)) \to \mathsf{S}(\mathsf{S}(\mathsf{d}(x)))$$
$$\mathsf{doubles}(\mathsf{Zero}) \to \mathsf{Nil}$$
$$\mathsf{doubles}(\mathsf{S}(x)) \to \mathsf{Cons}(\mathsf{d}(\mathsf{S}(x)), \mathsf{doubles}(x))$$

Parallel Dependency Tuples PDT$(\mathcal{R})$ for **independent** function calls:

$$\mathsf{d}^\sharp(\mathsf{Zero}) \to \mathsf{Com}_0$$
$$\mathsf{d}^\sharp(\mathsf{S}(x)) \to \mathsf{Com}_1(\mathsf{d}^\sharp(x))$$
$$\mathsf{doubles}^\sharp(\mathsf{Zero}) \to \mathsf{Com}_0$$
$$\mathsf{doubles}^\sharp(\mathsf{S}(x)) \to \mathsf{Com}_1(\mathsf{d}^\sharp(\mathsf{S}(x)))$$
$$\mathsf{doubles}^\sharp(\mathsf{S}(x)) \to \mathsf{Com}_1(\mathsf{doubles}^\sharp(x))$$

$\Rightarrow$ function calls for last rule **in parallel**

$\Rightarrow$ consider them **separately**, get the maximum of the costs

$$\mathrm{cost}(\mathsf{doubles}^\sharp(\mathsf{S}(x))) \quad = \quad 1 + \max(\ \mathrm{cost}(\mathsf{d}^\sharp(\mathsf{S}(x))),\ \mathrm{cost}(\mathsf{doubles}^\sharp(x))\ )$$

Parallel-innermost complexity: Example 1

TRS $\mathcal{R}$:

$$\mathsf{d}(\mathsf{Zero}) \to \mathsf{Zero}$$
$$\mathsf{d}(\mathsf{S}(x)) \to \mathsf{S}(\mathsf{S}(\mathsf{d}(x)))$$
$$\mathsf{doubles}(\mathsf{Zero}) \to \mathsf{Nil}$$
$$\mathsf{doubles}(\mathsf{S}(x)) \to \mathsf{Cons}(\mathsf{d}(\mathsf{S}(x)), \mathsf{doubles}(x))$$

Parallel Dependency Tuples PDT($\mathcal{R}$) for **independent** function calls:

$$\mathsf{d}^\sharp(\mathsf{Zero}) \to \mathsf{Com}_0$$
$$\mathsf{d}^\sharp(\mathsf{S}(x)) \to \mathsf{Com}_1(\mathsf{d}^\sharp(x))$$
$$\mathsf{doubles}^\sharp(\mathsf{Zero}) \to \mathsf{Com}_0$$
$$\mathsf{doubles}^\sharp(\mathsf{S}(x)) \to \mathsf{Com}_1(\mathsf{d}^\sharp(\mathsf{S}(x)))$$
$$\mathsf{doubles}^\sharp(\mathsf{S}(x)) \to \mathsf{Com}_1(\mathsf{doubles}^\sharp(x))$$

$\Rightarrow$ function calls for last rule **in parallel**

$\Rightarrow$ consider them **separately**, get the maximum of the costs

$$\begin{aligned} \mathrm{cost}(\mathsf{doubles}^\sharp(\mathsf{S}(x))) &= 1 + \max(\ \mathrm{cost}(\mathsf{d}^\sharp(\mathsf{S}(x))),\ \mathrm{cost}(\mathsf{doubles}^\sharp(x))\ ) \\ &= \max(\ 1 + \mathrm{cost}(\mathsf{d}^\sharp(\mathsf{S}(x))),\ 1 + \mathrm{cost}(\mathsf{doubles}^\sharp(x))\ ) \end{aligned}$$

Parallel-innermost complexity: Example 1

TRS $\mathcal{R}$:

$$\mathsf{d}(\mathsf{Zero}) \to \mathsf{Zero}$$
$$\mathsf{d}(\mathsf{S}(x)) \to \mathsf{S}(\mathsf{S}(\mathsf{d}(x)))$$
$$\mathsf{doubles}(\mathsf{Zero}) \to \mathsf{Nil}$$
$$\mathsf{doubles}(\mathsf{S}(x)) \to \mathsf{Cons}(\mathsf{d}(\mathsf{S}(x)), \mathsf{doubles}(x))$$

Parallel Dependency Tuples PDT($\mathcal{R}$) for **independent** function calls:

$$\mathsf{d}^\sharp(\mathsf{Zero}) \to \mathsf{Com}_0$$
$$\mathsf{d}^\sharp(\mathsf{S}(x)) \to \mathsf{Com}_1(\mathsf{d}^\sharp(x))$$
$$\mathsf{doubles}^\sharp(\mathsf{Zero}) \to \mathsf{Com}_0$$
$$\mathsf{doubles}^\sharp(\mathsf{S}(x)) \to \mathsf{Com}_1(\mathsf{d}^\sharp(\mathsf{S}(x)))$$
$$\mathsf{doubles}^\sharp(\mathsf{S}(x)) \to \mathsf{Com}_1(\mathsf{doubles}^\sharp(x))$$

$\Rightarrow$ function calls for last rule **in parallel**

$\Rightarrow$ consider them **separately**, get the maximum of the costs

$$\begin{aligned} \mathrm{cost}(\mathsf{doubles}^\sharp(\mathsf{S}(x))) &= 1 + \max(\ \mathrm{cost}(\mathsf{d}^\sharp(\mathsf{S}(x))),\ \mathrm{cost}(\mathsf{doubles}^\sharp(x))\ ) \\ &= \max(\ 1 + \mathrm{cost}(\mathsf{d}^\sharp(\mathsf{S}(x))),\ 1 + \mathrm{cost}(\mathsf{doubles}^\sharp(x))\ ) \end{aligned}$$

$\Rightarrow$ DT framework (for sequential complexity!) finds parallel complexity $\mathbf{pirc}_{\mathcal{R}}(n) \in \mathcal{O}(n)$

Parallel-innermost complexity: Example 1

TRS $\mathcal{R}$:

$$\mathsf{d}(\mathsf{Zero}) \to \mathsf{Zero}$$
$$\mathsf{d}(\mathsf{S}(x)) \to \mathsf{S}(\mathsf{S}(\mathsf{d}(x)))$$
$$\mathsf{doubles}(\mathsf{Zero}) \to \mathsf{Nil}$$
$$\mathsf{doubles}(\mathsf{S}(x)) \to \mathsf{Cons}(\mathsf{d}(\mathsf{S}(x)), \mathsf{doubles}(x))$$

Parallel Dependency Tuples PDT($\mathcal{R}$) for **independent** function calls:

$$\mathsf{d}^\sharp(\mathsf{Zero}) \to \mathsf{Com}_0$$
$$\mathsf{d}^\sharp(\mathsf{S}(x)) \to \mathsf{Com}_1(\mathsf{d}^\sharp(x))$$
$$\mathsf{doubles}^\sharp(\mathsf{Zero}) \to \mathsf{Com}_0$$
$$\mathsf{doubles}^\sharp(\mathsf{S}(x)) \to \mathsf{Com}_1(\mathsf{d}^\sharp(\mathsf{S}(x)))$$
$$\mathsf{doubles}^\sharp(\mathsf{S}(x)) \to \mathsf{Com}_1(\mathsf{doubles}^\sharp(x))$$

$\Rightarrow$ function calls for last rule **in parallel**

$\Rightarrow$ consider them **separately**, get the maximum of the costs

$$\begin{aligned} \mathrm{cost}(\mathsf{doubles}^\sharp(\mathsf{S}(x))) &= 1 + \max(\ \mathrm{cost}(\mathsf{d}^\sharp(\mathsf{S}(x))),\ \mathrm{cost}(\mathsf{doubles}^\sharp(x))\ ) \\ &= \max(\ 1 + \mathrm{cost}(\mathsf{d}^\sharp(\mathsf{S}(x))),\ 1 + \mathrm{cost}(\mathsf{doubles}^\sharp(x))\ ) \end{aligned}$$

$\Rightarrow$ DT framework (for sequential complexity!) finds parallel complexity $\mathbf{pirc}_\mathcal{R}(n) \in \mathcal{O}(n)$

Theorem (Baudon, Fuhs, Gonnord, LOPSTR 2022)

Parallel-innermost complexity of $\mathcal{R}$ ($\mathbf{pirc}_\mathcal{R}$) $\leq$ innermost complexity for DT problem for $\mathbf{PDT}(\mathcal{R})$.

Sequential complexity: Example 2

TRS $\mathcal{R}$:

$$\mathsf{plus}(\mathsf{Zero}, y) \to y$$
$$\mathsf{plus}(\mathsf{S}(x), y) \to \mathsf{S}(\mathsf{plus}(x, y))$$
$$\mathsf{size}(\mathsf{Nil}) \to \mathsf{Zero}$$

$$\mathsf{size}(\mathsf{Tree}(v, l, r)) \to \mathsf{S}(\mathsf{plus}(\mathsf{size}(l), \mathsf{size}(r)))$$

$\Rightarrow$ sum up costs of all function calls of a rule **together**

Sequential complexity: Example 2

TRS $\mathcal{R}$:

$$\mathsf{plus}(\mathsf{Zero}, y) \to y$$
$$\mathsf{plus}(\mathsf{S}(x), y) \to \mathsf{S}(\mathsf{plus}(x, y))$$
$$\mathsf{size}(\mathsf{Nil}) \to \mathsf{Zero}$$
$$\mathsf{size}(\mathsf{Tree}(v, l, r)) \to \mathsf{S}(\mathsf{plus}(\mathsf{size}(l), \mathsf{size}(r)))$$

Dependency Tuples $\mathbf{DT}(\mathcal{R})$ for function calls:

$$\mathsf{plus}^\sharp(\mathsf{Zero}) \to \mathsf{Com}_0$$
$$\mathsf{plus}^\sharp(\mathsf{S}(x), y) \to \mathsf{Com}_1(\mathsf{plus}^\sharp(x, y))$$
$$\mathsf{size}^\sharp(\mathsf{Nil}) \to \mathsf{Com}_0$$
$$\mathsf{size}^\sharp(\mathsf{Tree}(v, l, r)) \to \mathsf{Com}_3(\mathsf{size}^\sharp(l), \mathsf{size}^\sharp(r), \mathsf{plus}^\sharp(\mathsf{size}(l), \mathsf{size}(r)))$$

$\Rightarrow$ sum up costs of all function calls of a rule **together**

$$\mathrm{cost}(\mathsf{size}^\sharp(\mathsf{Tree}(v, l, r))) = 1 + \mathrm{cost}(\mathsf{size}^\sharp(l)) + \mathrm{cost}(\mathsf{size}^\sharp(r)) + \mathrm{cost}(\mathsf{plus}^\sharp(\mathsf{size}(l)\downarrow, \mathsf{size}(r)\downarrow))$$

Sequential complexity: Example 2

TRS $\mathcal{R}$:

$$\mathsf{plus}(\mathsf{Zero}, y) \to y$$
$$\mathsf{plus}(\mathsf{S}(x), y) \to \mathsf{S}(\mathsf{plus}(x, y))$$
$$\mathsf{size}(\mathsf{Nil}) \to \mathsf{Zero}$$
$$\mathsf{size}(\mathsf{Tree}(v, l, r)) \to \mathsf{S}(\mathsf{plus}(\mathsf{size}(l), \mathsf{size}(r)))$$

Dependency Tuples $\mathbf{DT}(\mathcal{R})$ for function calls:

$$\mathsf{plus}^\sharp(\mathsf{Zero}) \to \mathsf{Com}_0$$
$$\mathsf{plus}^\sharp(\mathsf{S}(x), y) \to \mathsf{Com}_1(\mathsf{plus}^\sharp(x, y))$$
$$\mathsf{size}^\sharp(\mathsf{Nil}) \to \mathsf{Com}_0$$
$$\mathsf{size}^\sharp(\mathsf{Tree}(v, l, r)) \to \mathsf{Com}_3(\mathsf{size}^\sharp(l), \mathsf{size}^\sharp(r), \mathsf{plus}^\sharp(\mathsf{size}(l), \mathsf{size}(r)))$$

$\Rightarrow$ sum up costs of all function calls of a rule **together**

$$\mathrm{cost}(\mathsf{size}^\sharp(\mathsf{Tree}(v, l, r))) = 1 + \mathrm{cost}(\mathsf{size}^\sharp(l)) + \mathrm{cost}(\mathsf{size}^\sharp(r)) + \mathrm{cost}(\mathsf{plus}^\sharp(\mathsf{size}(l)\downarrow, \mathsf{size}(r)\downarrow))$$

$\Rightarrow$ sequential complexity $\mathbf{irc}_\mathcal{R}(n) \in \mathcal{O}(n^2)$

Parallel-innermost complexity: Example 2

TRS $\mathcal{R}$:

$$\mathsf{plus}(\mathsf{Zero}, y) \to y$$
$$\mathsf{plus}(\mathsf{S}(x), y) \to \mathsf{S}(\mathsf{plus}(x, y))$$
$$\mathsf{size}(\mathsf{Nil}) \to \mathsf{Zero}$$

$$\mathsf{size}(\mathsf{Tree}(v, l, r)) \to \mathsf{S}(\mathsf{plus}(\mathsf{size}(l), \mathsf{size}(r)))$$

Parallel-innermost complexity: Example 2

TRS $\mathcal{R}$:

$$\mathsf{plus}(\mathsf{Zero}, y) \to y$$
$$\mathsf{plus}(\mathsf{S}(x), y) \to \mathsf{S}(\mathsf{plus}(x, y))$$
$$\mathsf{size}(\mathsf{Nil}) \to \mathsf{Zero}$$

$$\mathsf{size}(\mathsf{Tree}(v, l, r)) \to \mathsf{S}(\mathsf{plus}(\mathsf{size}(l), \mathsf{size}(r)))$$

$\Rightarrow$ consider **structural dependencies** of nested function calls

Parallel-innermost complexity: Example 2

Parallel Dependency Tuples PDT($\mathcal{R}$) for **chains** of nested function calls:

TRS $\mathcal{R}$:

$$\mathsf{plus}(\mathsf{Zero}, y) \to y$$
$$\mathsf{plus}(\mathsf{S}(x), y) \to \mathsf{S}(\mathsf{plus}(x, y))$$
$$\mathsf{size}(\mathsf{Nil}) \to \mathsf{Zero}$$
$$\mathsf{size}(\mathsf{Tree}(v, l, r)) \to \mathsf{S}(\mathsf{plus}(\mathsf{size}(l), \mathsf{size}(r)))$$

$$\mathsf{plus}^\sharp(\mathsf{Zero}) \to \mathsf{Com}_0$$
$$\mathsf{plus}^\sharp(\mathsf{S}(x), y) \to \mathsf{Com}_1(\mathsf{plus}^\sharp(x, y))$$
$$\mathsf{size}^\sharp(\mathsf{Nil}) \to \mathsf{Com}_0$$
$$\mathsf{size}^\sharp(\mathsf{Tree}(v, l, r)) \to \mathsf{Com}_2(\mathsf{size}^\sharp(l), \mathsf{plus}^\sharp(\mathsf{size}(l), \mathsf{size}(r)))$$
$$\mathsf{size}^\sharp(\mathsf{Tree}(v, l, r)) \to \mathsf{Com}_2(\mathsf{size}^\sharp(r), \mathsf{plus}^\sharp(\mathsf{size}(l), \mathsf{size}(r)))$$

$\Rightarrow$ consider **structural dependencies** of nested function calls

Parallel-innermost complexity: Example 2

Parallel Dependency Tuples PDT($\mathcal{R}$) for **chains** of nested function calls:

TRS $\mathcal{R}$:

$$\mathsf{plus}(\mathsf{Zero}, y) \to y$$
$$\mathsf{plus}(\mathsf{S}(x), y) \to \mathsf{S}(\mathsf{plus}(x, y))$$
$$\mathsf{size}(\mathsf{Nil}) \to \mathsf{Zero}$$
$$\mathsf{size}(\mathsf{Tree}(v, l, r)) \to \mathsf{S}(\mathsf{plus}(\mathsf{size}(l), \mathsf{size}(r)))$$

$$\mathsf{plus}^\sharp(\mathsf{Zero}) \to \mathsf{Com}_0$$
$$\mathsf{plus}^\sharp(\mathsf{S}(x), y) \to \mathsf{Com}_1(\mathsf{plus}^\sharp(x, y))$$
$$\mathsf{size}^\sharp(\mathsf{Nil}) \to \mathsf{Com}_0$$
$$\mathsf{size}^\sharp(\mathsf{Tree}(v, l, r)) \to \mathsf{Com}_2(\mathsf{size}^\sharp(l), \mathsf{plus}^\sharp(\mathsf{size}(l), \mathsf{size}(r)))$$
$$\mathsf{size}^\sharp(\mathsf{Tree}(v, l, r)) \to \mathsf{Com}_2(\mathsf{size}^\sharp(r), \mathsf{plus}^\sharp(\mathsf{size}(l), \mathsf{size}(r)))$$

$\Rightarrow$ consider **structural dependencies** of nested function calls

$$\mathrm{cost}(\mathsf{size}^\sharp(\mathsf{S}(x))) \quad = \quad 1 + \max(\ \mathrm{cost}(\mathsf{size}^\sharp(l)),\ \mathrm{cost}(\mathsf{size}^\sharp(r))\ ) + \mathrm{cost}(\mathsf{plus}^\sharp(\mathsf{size}(l)\downarrow, \mathsf{size}(r)\downarrow)\ )$$

Parallel-innermost complexity: Example 2

Parallel Dependency Tuples PDT($\mathcal{R}$) for **chains** of nested function calls:

TRS $\mathcal{R}$:

$$\mathsf{plus}(\mathsf{Zero}, y) \to y$$
$$\mathsf{plus}(\mathsf{S}(x), y) \to \mathsf{S}(\mathsf{plus}(x, y))$$
$$\mathsf{size}(\mathsf{Nil}) \to \mathsf{Zero}$$
$$\mathsf{size}(\mathsf{Tree}(v, l, r)) \to \mathsf{S}(\mathsf{plus}(\mathsf{size}(l), \mathsf{size}(r)))$$

$$\mathsf{plus}^\sharp(\mathsf{Zero}) \to \mathsf{Com}_0$$
$$\mathsf{plus}^\sharp(\mathsf{S}(x), y) \to \mathsf{Com}_1(\mathsf{plus}^\sharp(x, y))$$
$$\mathsf{size}^\sharp(\mathsf{Nil}) \to \mathsf{Com}_0$$
$$\mathsf{size}^\sharp(\mathsf{Tree}(v, l, r)) \to \mathsf{Com}_2(\mathsf{size}^\sharp(l), \mathsf{plus}^\sharp(\mathsf{size}(l), \mathsf{size}(r)))$$
$$\mathsf{size}^\sharp(\mathsf{Tree}(v, l, r)) \to \mathsf{Com}_2(\mathsf{size}^\sharp(r), \mathsf{plus}^\sharp(\mathsf{size}(l), \mathsf{size}(r)))$$

$\Rightarrow$ consider **structural dependencies** of nested function calls

$$\begin{aligned}\mathrm{cost}(\mathsf{size}^\sharp(\mathsf{S}(x))) &= 1 + \max(\ \mathrm{cost}(\mathsf{size}^\sharp(l)),\ \mathrm{cost}(\mathsf{size}^\sharp(r))\ ) + \mathrm{cost}(\mathsf{plus}^\sharp(\mathsf{size}(l)\downarrow, \mathsf{size}(r)\downarrow)\ ) \\ &= \max(\ 1 + \mathrm{cost}(\mathsf{size}^\sharp(l)) + \mathrm{cost}(\mathsf{plus}^\sharp(\mathsf{size}(l)\downarrow, \mathsf{size}(r)\downarrow), \\ &\qquad 1 + \mathrm{cost}(\mathsf{size}^\sharp(r)) + \mathrm{cost}(\mathsf{plus}^\sharp(\mathsf{size}(l)\downarrow, \mathsf{size}(r)\downarrow)\ )\end{aligned}$$

Parallel-innermost complexity: Example 2

Parallel Dependency Tuples PDT($\mathcal{R}$) for **chains** of nested function calls:

TRS $\mathcal{R}$:

$$\mathsf{plus}(\mathsf{Zero}, y) \to y$$
$$\mathsf{plus}(\mathsf{S}(x), y) \to \mathsf{S}(\mathsf{plus}(x, y))$$
$$\mathsf{size}(\mathsf{Nil}) \to \mathsf{Zero}$$
$$\mathsf{size}(\mathsf{Tree}(v, l, r)) \to \mathsf{S}(\mathsf{plus}(\mathsf{size}(l), \mathsf{size}(r)))$$

$$\mathsf{plus}^\sharp(\mathsf{Zero}) \to \mathsf{Com}_0$$
$$\mathsf{plus}^\sharp(\mathsf{S}(x), y) \to \mathsf{Com}_1(\mathsf{plus}^\sharp(x, y))$$
$$\mathsf{size}^\sharp(\mathsf{Nil}) \to \mathsf{Com}_0$$
$$\mathsf{size}^\sharp(\mathsf{Tree}(v, l, r)) \to \mathsf{Com}_2(\mathsf{size}^\sharp(l), \mathsf{plus}^\sharp(\mathsf{size}(l), \mathsf{size}(r)))$$
$$\mathsf{size}^\sharp(\mathsf{Tree}(v, l, r)) \to \mathsf{Com}_2(\mathsf{size}^\sharp(r), \mathsf{plus}^\sharp(\mathsf{size}(l), \mathsf{size}(r)))$$

$\Rightarrow$ consider **structural dependencies** of nested function calls

$$\begin{aligned}\mathrm{cost}(\mathsf{size}^\sharp(\mathsf{S}(x))) &= 1 + \max(\ \mathrm{cost}(\mathsf{size}^\sharp(l)),\ \mathrm{cost}(\mathsf{size}^\sharp(r))\ ) + \mathrm{cost}(\mathsf{plus}^\sharp(\mathsf{size}(l)\downarrow, \mathsf{size}(r)\downarrow)\ ) \\ &= \max(\ 1 + \mathrm{cost}(\mathsf{size}^\sharp(l)) + \mathrm{cost}(\mathsf{plus}^\sharp(\mathsf{size}(l)\downarrow, \mathsf{size}(r)\downarrow), \\ &\qquad 1 + \mathrm{cost}(\mathsf{size}^\sharp(r)) + \mathrm{cost}(\mathsf{plus}^\sharp(\mathsf{size}(l)\downarrow, \mathsf{size}(r)\downarrow)\ )\end{aligned}$$

$\Rightarrow$ parallel complexity $\mathbf{pirc}_{\mathcal{R}}(n) \in \mathcal{O}(n^2)$

Parallel-innermost complexity: Example 2

Parallel Dependency Tuples PDT($\mathcal{R}$) for **chains** of nested function calls:

TRS $\mathcal{R}$:

$$\mathsf{plus}(\mathsf{Zero}, y) \to y$$
$$\mathsf{plus}(\mathsf{S}(x), y) \to \mathsf{S}(\mathsf{plus}(x, y))$$
$$\mathsf{size}(\mathsf{Nil}) \to \mathsf{Zero}$$
$$\mathsf{size}(\mathsf{Tree}(v, l, r)) \to \mathsf{S}(\mathsf{plus}(\mathsf{size}(l), \mathsf{size}(r)))$$

$$\mathsf{plus}^\sharp(\mathsf{Zero}) \to \mathsf{Com}_0$$
$$\mathsf{plus}^\sharp(\mathsf{S}(x), y) \to \mathsf{Com}_1(\mathsf{plus}^\sharp(x, y))$$
$$\mathsf{size}^\sharp(\mathsf{Nil}) \to \mathsf{Com}_0$$
$$\mathsf{size}^\sharp(\mathsf{Tree}(v, l, r)) \to \mathsf{Com}_2(\mathsf{size}^\sharp(l), \mathsf{plus}^\sharp(\mathsf{size}(l), \mathsf{size}(r)))$$
$$\mathsf{size}^\sharp(\mathsf{Tree}(v, l, r)) \to \mathsf{Com}_2(\mathsf{size}^\sharp(r), \mathsf{plus}^\sharp(\mathsf{size}(l), \mathsf{size}(r)))$$

$\Rightarrow$ consider **structural dependencies** of nested function calls

$$\begin{aligned}\mathrm{cost}(\mathsf{size}^\sharp(\mathsf{S}(x))) &= 1 + \max(\ \mathrm{cost}(\mathsf{size}^\sharp(l)),\ \mathrm{cost}(\mathsf{size}^\sharp(r))\ ) + \mathrm{cost}(\mathsf{plus}^\sharp(\mathsf{size}(l)\downarrow, \mathsf{size}(r)\downarrow)\ ) \\ &= \max(\ 1 + \mathrm{cost}(\mathsf{size}^\sharp(l)) + \mathrm{cost}(\mathsf{plus}^\sharp(\mathsf{size}(l)\downarrow, \mathsf{size}(r)\downarrow), \\ &\qquad 1 + \mathrm{cost}(\mathsf{size}^\sharp(r)) + \mathrm{cost}(\mathsf{plus}^\sharp(\mathsf{size}(l)\downarrow, \mathsf{size}(r)\downarrow)\ )\end{aligned}$$

$\Rightarrow$ parallel complexity $\mathbf{pirc}_{\mathcal{R}}(n) \in \mathcal{O}(n^2)$

... alas, a tight bound: consider $\mathsf{size}(\mathsf{Tree}(\mathsf{Zero}, \mathsf{Tree}(\mathsf{Zero}, \ldots \mathsf{Tree}(\mathsf{Zero}, \mathsf{Nil}, \mathsf{Nil}))), \ldots \mathsf{Nil}, \mathsf{Nil})$

Proving Absence of Parallelism

Parallel Dependency Tuples can detect TRSs without potential for parallelism:

Proving Absence of Parallelism

Parallel Dependency Tuples can detect TRSs without potential for parallelism:

Theorem (Baudon, Fuhs, Gonnord, LOPSTR 2022)

*Let $\mathcal{R}$ be a TRS where no rule $\ell \to r$ has defined symbols at **parallel positions** of r. Then:*

1. $\mathit{PDT}(\mathcal{R}) = \mathit{DT}(\mathcal{R})$
2. *from basic terms* $f(t_1, \ldots, t_n)$, $\parallel\!\!\to_{\mathcal{R}} \;=\; \to_{\mathcal{R}}$
3. $\mathit{pirc}_{\mathcal{R}}(n) \;=\; \mathit{irc}_{\mathcal{R}}(n)$

Proving Absence of Parallelism

Parallel Dependency Tuples can detect TRSs without potential for parallelism:

Theorem (Baudon, Fuhs, Gonnord, LOPSTR 2022)

*Let $\mathcal{R}$ be a TRS where no rule $\ell \to r$ has defined symbols at **parallel positions** of r. Then:*

1. $\mathit{PDT}(\mathcal{R}) = \mathit{DT}(\mathcal{R})$
2. *from basic terms* $f(t_1, \ldots, t_n)$, $\fatsemi\!\!\fatsemi\!\!\to_{\mathcal{R}} \;=\; \to_{\mathcal{R}}$
3. $\mathit{pirc}_{\mathcal{R}}(n) \;=\; \mathit{irc}_{\mathcal{R}}(n)$

Proof Idea.

To get parallelism from basic term (1 function call), some rule $\ell \to r$ must make 2+ function calls at **parallel positions** in r. Each rule has only 1 **PDT** $\Rightarrow$ no parallel calls! □

Proving Absence of Parallelism

Parallel Dependency Tuples can detect TRSs without potential for parallelism:

Theorem (Baudon, Fuhs, Gonnord, LOPSTR 2022)

*Let $\mathcal{R}$ be a TRS where no rule $\ell \to r$ has defined symbols at **parallel positions** of r. Then:*

1. $\mathit{PDT}(\mathcal{R}) = \mathit{DT}(\mathcal{R})$
2. *from basic terms* $f(t_1, \ldots, t_n)$, $\parallel\!\!\to^{\mathrm{i}}_{\mathcal{R}} \;=\; \to^{\mathrm{i}}_{\mathcal{R}}$
3. $\mathit{pirc}_{\mathcal{R}}(n) \;=\; \mathit{irc}_{\mathcal{R}}(n)$

Proof Idea.

To get parallelism from basic term (1 function call), some rule $\ell \to r$ must make 2+ function calls at **parallel positions** in r. Each rule has only 1 **PDT** $\Rightarrow$ no parallel calls! □

$\Rightarrow$ Quick check to refute that parallel evaluation is feasible.

Lower bounds on worst-case parallel complexity? (1/2)

Back to the motivation – which function calls are worth parallelising?

Want: **lower bounds** on $\mathbf{pirc}_{\mathcal{R}}$

Lower bounds on worst-case parallel complexity? (1/2)

Back to the motivation – which function calls are worth parallelising?

Want: **lower bounds** on $\mathbf{pirc}_{\mathcal{R}}$

Recall for sequential complexity:

Theorem (Noschinski, Emmes, Giesl, J. Autom. Reasoning 2013)

$\mathbf{irc}_{\mathcal{R}} \leq$ innermost complexity for DT problem for $\mathbf{DT}(\mathcal{R})$

Lower bounds on worst-case parallel complexity? (1/2)

Back to the motivation – which function calls are worth parallelising?

Want: **lower bounds** on $\mathbf{pirc}_{\mathcal{R}}$

Recall for sequential complexity:

Theorem (Noschinski, Emmes, Giesl, J. Autom. Reasoning 2013)

$\mathbf{irc}_{\mathcal{R}}$ = innermost complexity for DT problem for $\mathbf{DT}(\mathcal{R})$ ***if $\xrightarrow{\text{i}}_{\mathcal{R}}$ is confluent.***

Lower bounds on worst-case parallel complexity? (1/2)

Back to the motivation – which function calls are worth parallelising?

Want: **lower bounds** on $\mathbf{pirc}_{\mathcal{R}}$

Recall for sequential complexity:

Theorem (Noschinski, Emmes, Giesl, J. Autom. Reasoning 2013)

$\mathbf{irc}_{\mathcal{R}}$ = innermost complexity for DT problem for $\mathbf{DT}(\mathcal{R})$ if $\xrightarrow{\text{i}}_{\mathcal{R}}$ is confluent.

So: lower bounds of DT problem for $\mathbf{DT}(\mathcal{R})$ carry over to $\mathcal{R}$ itself!

Also for $\mathbf{pirc}_{\mathcal{R}}$?

Lower bounds on worst-case parallel complexity? (1/2)

Back to the motivation – which function calls are worth parallelising?

Want: **lower bounds** on $\mathbf{pirc}_{\mathcal{R}}$

Recall for sequential complexity:

Theorem (Noschinski, Emmes, Giesl, J. Autom. Reasoning 2013)

*$\mathbf{irc}_{\mathcal{R}}$ = innermost complexity for DT problem for $\mathbf{DT}(\mathcal{R})$ **if** $\xrightarrow{\mathrm{i}}_{\mathcal{R}}$ **is confluent**.*

So: lower bounds of DT problem for $\mathbf{DT}(\mathcal{R})$ carry over to $\mathcal{R}$ itself!

Also for $\mathbf{pirc}_{\mathcal{R}}$? – Yes!

Theorem (Baudon, Fuhs, Gonnord, LOPSTR 2022)

*$\mathbf{pirc}_{\mathcal{R}}$ = innermost complexity for DT problem for $\mathbf{PDT}(\mathcal{R})$ **if** $\parallel\!\!\xrightarrow{\mathrm{i}}_{\mathcal{R}}$ **is confluent**.*

Lower bounds on worst-case parallel complexity? (2/2)

Have: Input TRS $\mathcal{R}$, Parallel Dependency Tuples $\mathbf{PDT}(\mathcal{R})$.

Lower bounds on worst-case parallel complexity? (2/2)

Have: Input TRS $\mathcal{R}$, Parallel Dependency Tuples $\mathbf{PDT}(\mathcal{R})$.

Need:

1. Find lower bounds for DT problem
2. Prove confluence of $\overset{i}{\Longrightarrow}_{\mathcal{R}}$

Lower bounds on worst-case parallel complexity? (2/2)

Have: Input TRS $\mathcal{R}$, Parallel Dependency Tuples $\mathbf{PDT}(\mathcal{R})$.

Need:

1. Find lower bounds for DT problem
2. Prove confluence of $\pitchfork\!\!\!\rightarrow_{\mathcal{R}}$

Ad 1: Finding lower bounds for DT problem

Lower bounds on worst-case parallel complexity? (2/2)

Have: Input TRS $\mathcal{R}$, Parallel Dependency Tuples $\mathbf{PDT}(\mathcal{R})$.

Need:

1. Find lower bounds for DT problem
2. Prove confluence of $\pitchfork\!\!\!\rightarrow^{i}_{\mathcal{R}}$

Ad 1: Finding lower bounds for DT problem

- Transform DT problem back to TRS
- Use existing lower bound inference for (sequential) innermost rewriting on TRS level[1]

[1] F. Frohn, J. Giesl, J. Hensel, C. Aschermann, T. Ströder: *Lower bounds for runtime complexity of term rewriting*, J. Autom. Reasoning, 2017

Lower bounds on worst-case parallel complexity? (2/2)

Have: Input TRS $\mathcal{R}$, Parallel Dependency Tuples $\mathbf{PDT}(\mathcal{R})$.

Need:

1. Find lower bounds for DT problem
2. Prove confluence of $\overset{\mathsf{i}}{\Longrightarrow}_{\mathcal{R}}$

Ad 1: Finding lower bounds for DT problem

- Transform DT problem back to TRS
- Use existing lower bound inference for (sequential) innermost rewriting on TRS level[1]

Theorem (Baudon, Fuhs, Gonnord, LOPSTR 2022)

$\mathbf{pirc}_{\mathcal{R}}$ = (sequential) innermost complexity of relative TRS $\mathbf{PDT}(\mathcal{R})/\mathcal{R}$ ***if*** $\overset{\mathsf{i}}{\Longrightarrow}_{\mathcal{R}}$ ***is confluent.***

[1] F. Frohn, J. Giesl, J. Hensel, C. Aschermann, T. Ströder: *Lower bounds for runtime complexity of term rewriting*, J. Autom. Reasoning, 2017

Lower bounds on worst-case parallel complexity? (2/2)

Have: Input TRS $\mathcal{R}$, Parallel Dependency Tuples $\mathbf{PDT}(\mathcal{R})$.

Need:

1. Find lower bounds for DT problem
2. Prove confluence of $\overset{\mathsf{i}}{\Longrightarrow}_{\mathcal{R}}$

Ad 1: Finding lower bounds for DT problem

- Transform DT problem back to TRS
- Use existing lower bound inference for (sequential) innermost rewriting on TRS level[1]

Theorem (Baudon, Fuhs, Gonnord, LOPSTR 2022)

$\mathit{pirc}_{\mathcal{R}}$ = *(sequential) innermost complexity of relative TRS* $\mathbf{PDT}(\mathcal{R})/\mathcal{R}$ ***if*** $\overset{\mathsf{i}}{\Longrightarrow}_{\mathcal{R}}$ ***is confluent.***

Ad 2: Done! (...with a lot of room for improvement)

[1] F. Frohn, J. Giesl, J. Hensel, C. Aschermann, T. Ströder: *Lower bounds for runtime complexity of term rewriting*, J. Autom. Reasoning, 2017

Conclusion for "Proving confluence of parallel-innermost rewriting: why?"

- First approach to finding bounds on **parallel**-innermost runtime complexity of TRSs

Conclusion for "Proving confluence of parallel-innermost rewriting: why?"

- First approach to finding bounds on **parallel**-innermost runtime complexity of TRSs
- Works for upper **and** lower bounds (confluence required for lower bounds!)

Conclusion for "Proving confluence of parallel-innermost rewriting: why?"

- First approach to finding bounds on **parallel**-innermost runtime complexity of TRSs
- Works for upper **and** lower bounds (confluence required for lower bounds!)
- Builds on Dependency Tuples available in AProVE and TcT

Conclusion for "Proving confluence of parallel-innermost rewriting: why?"

- First approach to finding bounds on **parallel**-innermost runtime complexity of TRSs
- Works for upper **and** lower bounds (confluence required for lower bounds!)
- Builds on Dependency Tuples available in AProVE and TcT
- Thaïs Baudon, Carsten Fuhs, Laure Gonnord
  Analysing Parallel Complexity of Term Rewriting
  In *Proc. 32nd International Symposium on Logic-based Program Synthesis and Transformation (LOPSTR)*, 2022. To appear.

Conclusion for "Proving confluence of parallel-innermost rewriting: why?"

- First approach to finding bounds on **parallel**-innermost runtime complexity of TRSs
- Works for upper **and** lower bounds (confluence required for lower bounds!)
- Builds on Dependency Tuples available in AProVE and TcT
- Thaïs Baudon, Carsten Fuhs, Laure Gonnord
  Analysing Parallel Complexity of Term Rewriting
  In *Proc. 32nd International Symposium on Logic-based Program Synthesis and Transformation (LOPSTR)*, 2022. To appear.
- Evaluation page with details of experiments:

https://www.dcs.bbk.ac.uk/~carsten/eval/parallel_complexity/

Conclusion for "Proving confluence of parallel-innermost rewriting: why?"

- First approach to finding bounds on **parallel**-innermost runtime complexity of TRSs
- Works for upper **and** lower bounds (confluence required for lower bounds!)
- Builds on Dependency Tuples available in AProVE and TcT
- Thaïs Baudon, Carsten Fuhs, Laure Gonnord
  Analysing Parallel Complexity of Term Rewriting
  In *Proc. 32nd International Symposium on Logic-based Program Synthesis and Transformation (LOPSTR)*, 2022. To appear.
- Evaluation page with details of experiments:

https://www.dcs.bbk.ac.uk/~carsten/eval/parallel_complexity/

Thanks a lot for your attention!

Bonus slides:

More on complexity analysis for parallel-innermost term rewriting

Complexity: Related Work

- Literature on parallel computing / program analysis:
  - sequential complexity, $\mathbf{irc}_{\mathcal{R}}$: **work**
  - parallel complexity, $\mathbf{pirc}_{\mathcal{R}}$: **depth**, **span**

Complexity: Related Work

- Literature on parallel computing / program analysis:
  - sequential complexity, $\mathbf{irc}_{\mathcal{R}}$: **work**
  - parallel complexity, $\mathbf{pirc}_{\mathcal{R}}$: **depth**, **span**
- Analysis of **parallel complexity** in many settings
  - async/finish programs[2][3]
  - RaML: functional programs with list and pair constructors[4]
  - logic programs[5]
  - pi calculus[6][7]

---

[2] E. Albert, P. Arenas, S. Genaim, D. Zanardini: *Task-level analysis for a language with async/finish parallelism*, LCTES 2011

[3] E. Albert, J. Correas, E.B. Johnsen, V.K.I. Pun, G. Román-Díez: *Parallel Cost Analysis*, TOCL 2018

[4] J. Hoffmann, Z. Shao: *Automatic Static Cost Analysis for Parallel Programs*, ESOP 2015

[5] M. Klemen, P. López-García, J.P. Gallagher, J.F. Morales, M.V. Hermenegildo: *A General Framework for Static Cost Analysis of Parallel Logic Programs*, LOPSTR 2019

[6] P. Baillot, A. Ghyselen: *Types for complexity of parallel computation in pi-calculus*, ESOP 2021

[7] P. Baillot, A. Ghyselen, N. Kobayashi: *Sized Types with Usages for Parallel Complexity of Pi-Calculus Processes*, CONCUR 2021

Complexity: Related Work

- Literature on parallel computing / program analysis:
  - sequential complexity, $\mathbf{irc}_{\mathcal{R}}$: **work**
  - parallel complexity, $\mathbf{pirc}_{\mathcal{R}}$: **depth**, **span**
- Analysis of **parallel complexity** in many settings
  - async/finish programs[2][3]
  - RaML: functional programs with list and pair constructors[4]
  - logic programs[5]
  - pi calculus[6][7]
- **Massively parallel** implementation of innermost rewriting on GPUs[8]

[2] E. Albert, P. Arenas, S. Genaim, D. Zanardini: *Task-level analysis for a language with async/finish parallelism*, LCTES 2011

[3] E. Albert, J. Correas, E.B. Johnsen, V.K.I. Pun, G. Román-Díez: *Parallel Cost Analysis*, TOCL 2018

[4] J. Hoffmann, Z. Shao: *Automatic Static Cost Analysis for Parallel Programs*, ESOP 2015

[5] M. Klemen, P. López-García, J.P. Gallagher, J.F. Morales, M.V. Hermenegildo: *A General Framework for Static Cost Analysis of Parallel Logic Programs*, LOPSTR 2019

[6] P. Baillot, A. Ghyselen: *Types for complexity of parallel computation in pi-calculus*, ESOP 2021

[7] P. Baillot, A. Ghyselen, N. Kobayashi: *Sized Types with Usages for Parallel Complexity of Pi-Calculus Processes*, CONCUR 2021

[8] J. van Eerd, J.F. Groote, P. Hijma, J. Martens, A. Wijs: *Term Rewriting on GPUs*, FSEN 2021

Complexity: Implementation and Experiments (1/3)

- Implementation in program analysis tool AProVE
- Building on existing framework for sequential innermost runtime complexity

Complexity: Implementation and Experiments (1/3)

- Implementation in program analysis tool AProVE
- Building on existing framework for sequential innermost runtime complexity

Question for experiments:

Does (unbounded) parallelism lead to asymptotically more efficient innermost rewriting?

Complexity: Implementation and Experiments (1/3)

- Implementation in program analysis tool AProVE
- Building on existing framework for sequential innermost runtime complexity

Question for experiments:

> Does (unbounded) parallelism lead to asymptotically more efficient innermost rewriting?

Benchmark set:

Termination Problem DataBase (TPDB), v11.2, category `Innermost_Runtime_Complexity`
$\Rightarrow$ 663 TRSs

Complexity: Implementation and Experiments (1/3)

- Implementation in program analysis tool AProVE
- Building on existing framework for sequential innermost runtime complexity

Question for experiments:

Does (unbounded) parallelism lead to asymptotically more efficient innermost rewriting?

Benchmark set:

Termination Problem DataBase (TPDB), v11.2, category `Innermost_Runtime_Complexity`
$\Rightarrow$ 663 TRSs

Preprocessing:

Remove TRSs $\mathcal{R}$ without parallel function calls!
$\Rightarrow$ **294** remaining TRSs

Complexity: Implementation and Experiments (1/3)

- Implementation in program analysis tool AProVE
- Building on existing framework for sequential innermost runtime complexity

Question for experiments:

Does (unbounded) parallelism lead to asymptotically more efficient innermost rewriting?

Benchmark set:

Termination Problem DataBase (TPDB), v11.2, category `Innermost_Runtime_Complexity`
$\Rightarrow$ 663 TRSs

Preprocessing:

Remove TRSs $\mathcal{R}$ without parallel function calls!
$\Rightarrow$ **294** remaining TRSs

Timeout per TRS: 300 seconds

Complexity: Implementation and Experiments (2/3)

No other tools for **parallel**-innermost complexity so far.

But: $\mathbf{pirc}_{\mathcal{R}}(n) \leq \mathbf{irc}_{\mathcal{R}}(n)$

Complexity: Implementation and Experiments (2/3)

No other tools for **parallel**-innermost complexity so far.

But: $\mathbf{pirc}_{\mathcal{R}}(n) \leq \mathbf{irc}_{\mathcal{R}}(n)$

$\Rightarrow$ Compare with upper bounds by TermComp 2021 tools for innermost complexity AProVE, TcT

Complexity: Implementation and Experiments (2/3)

No other tools for **parallel**-innermost complexity so far.

But: $\mathbf{pirc}_{\mathcal{R}}(n) \leq \mathbf{irc}_{\mathcal{R}}(n)$

$\Rightarrow$ Compare with upper bounds by TermComp 2021 tools for innermost complexity AProVE, TcT

Upper bounds	$\mathcal{O}(1)$	$\leq \mathcal{O}(n)$	$\leq \mathcal{O}(n^2)$	$\leq \mathcal{O}(n^3)$	$\leq \mathcal{O}(n^{\geq 4})$
TcT $\mathbf{irc}_{\mathcal{R}}$	4	28	39	44	44
AProVE $\mathbf{irc}_{\mathcal{R}}$	**5**	50	110	123	127
AProVE $\mathbf{pirc}_{\mathcal{R}}$: DT problem only	**5**	65	**125**	**140**	**142**
AProVE $\mathbf{pirc}_{\mathcal{R}}$: DT problem + rel. TRS	**5**	**69**	**125**	139	141

- improved linear complexity by 38%
- `TCT_12/recursion_10` improves from $\mathcal{O}(n^{10})$ to $\mathcal{O}(n^1)$

Complexity: Implementation and Experiments (3/3)

Lower bounds via relative TRS $\mathbf{PDT}(\mathcal{R})/\mathcal{R}$:

Lower bounds	|benchmark set|	confluent	$\geq \Omega(n)$	$\geq \Omega(n^2)$	$\geq \Omega(n^3)$	$\geq \Omega(n^{\geq 4})$
AProVE $\mathbf{pirc}_{\mathcal{R}}$	294	186	133	23	5	1

- **Challenge:** better confluence analysis for **parallel**-innermost rewriting!
- Non-trivial lower bounds for 133 of 186 provably confluent TRSs

Complexity: Implementation and Experiments (3/3)

Lower bounds via relative TRS $\mathbf{PDT}(\mathcal{R})/\mathcal{R}$:

Lower bounds	|benchmark set|	confluent	$\geq \Omega(n)$	$\geq \Omega(n^2)$	$\geq \Omega(n^3)$	$\geq \Omega(n^{\geq 4})$
AProVE **pirc**$_{\mathcal{R}}$	294	186	133	23	5	1

- **Challenge:** better confluence analysis for **parallel**-innermost rewriting!
- Non-trivial lower bounds for 133 of 186 provably confluent TRSs

Together:

Tight bounds	$\Theta(1)$	$\Theta(n)$	$\Theta(n^2)$	$\Theta(n^3)$	Total
AProVE **pirc**$_{\mathcal{R}}$	5	32	1	3	**41**

Complexity: Conclusion

- First approach to finding bounds on **parallel**-innermost runtime complexity of TRSs

Complexity: Conclusion

- First approach to finding bounds on **parallel**-innermost runtime complexity of TRSs
- Works for upper **and** lower bounds (confluence required for lower bounds!)

Complexity: Conclusion

- First approach to finding bounds on **parallel**-innermost runtime complexity of TRSs
- Works for upper **and** lower bounds (confluence required for lower bounds!)
- Builds on Dependency Tuples available in AProVE and TcT

Complexity: Conclusion

- First approach to finding bounds on **parallel**-innermost runtime complexity of TRSs
- Works for upper **and** lower bounds (confluence required for lower bounds!)
- Builds on Dependency Tuples available in AProVE and TcT
- T. Baudon, C. Fuhs, L. Gonnord
  Analysing Parallel Complexity of Term Rewriting
  In *Proc. 32nd International Symposium on Logic-based Program Synthesis and Transformation (LOPSTR)*, 2022. To appear.

Complexity: Conclusion

- First approach to finding bounds on **parallel**-innermost runtime complexity of TRSs
- Works for upper **and** lower bounds (confluence required for lower bounds!)
- Builds on Dependency Tuples available in AProVE and TcT
- T. Baudon, C. Fuhs, L. Gonnord
  Analysing Parallel Complexity of Term Rewriting
  In *Proc. 32nd International Symposium on Logic-based Program Synthesis and Transformation (LOPSTR)*, 2022. To appear.
- Evaluation page with details of experiments:

https://www.dcs.bbk.ac.uk/~carsten/eval/parallel_complexity/

Complexity: Conclusion

- First approach to finding bounds on **parallel**-innermost runtime complexity of TRSs
- Works for upper **and** lower bounds (confluence required for lower bounds!)
- Builds on Dependency Tuples available in AProVE and TcT
- T. Baudon, C. Fuhs, L. Gonnord
  Analysing Parallel Complexity of Term Rewriting
  In *Proc. 32nd International Symposium on Logic-based Program Synthesis and Transformation (LOPSTR)*, 2022. To appear.
- Evaluation page with details of experiments:

https://www.dcs.bbk.ac.uk/~carsten/eval/parallel_complexity/

Thanks a lot for your attention! (take 2)