Automatic Analysis of Runtime Complexity for 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

TeReSe 2022, Nijmegen, The Netherlands, 8th June 2022

Overview

1. What is parallel-innermost rewriting (and why is it interesting)?
2. How do we measure its complexity?
3. How can we find **upper** bounds for its complexity?
4. How can we find **lower** bounds for its complexity?
5. Related work
6. Experiments

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

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)

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)

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)

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)

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

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 $\mathcal{R}$:

$$\begin{aligned}\mathsf{d}(\mathsf{Zero}) &\to \mathsf{Zero}\\ \mathsf{d}(\mathsf{S}(x)) &\to \mathsf{S}(\mathsf{S}(\mathsf{d}(x)))\end{aligned}$$

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

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:

$$\mathsf{doubles}(\mathsf{S}(\mathsf{Zero}))$$
$$\xrightarrow{\mathsf{i}}_{\mathcal{R}} \mathsf{Cons}(\mathsf{d}(\mathsf{S}(\mathsf{Zero}))\,, \mathsf{doubles}(\mathsf{Zero}))$$

Parallel-innermost rewriting: How it works

TRS $\mathcal{R}$:

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

Reduction with (sequential) innermost rewriting:

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

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:

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

4 steps.

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:

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

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:

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

4 steps.

Reduction with **parallel-**innermost rewriting:

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

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:

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

4 steps.

Reduction with **parallel-**innermost rewriting:

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

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:

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

4 steps.

Reduction with **parallel-**innermost rewriting:

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

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:

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

4 steps.

Reduction with **parallel**-innermost rewriting:

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

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:

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

4 steps.

Reduction with **parallel**-innermost rewriting:

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

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 $\mathcal{R}$:

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

Sequential complexity: Example 1

TRS $\mathcal{R}$:

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

$\Rightarrow$ 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 ***irc***$_{\mathcal{R}}$*) $\leq$ innermost complexity for DT problem for* $\boldsymbol{DT}(\mathcal{R})$.

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 **irc**$_\mathcal{R}$) $\leq$ innermost complexity for DT problem for **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 $\mathcal{R}$:

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

Parallel-innermost complexity: Example 1

TRS $\mathcal{R}$:

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

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

Parallel-innermost complexity: Example 1

TRS $\mathcal{R}$:

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

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

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

Parallel-innermost complexity: Example 1

TRS $\mathcal{R}$:

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

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

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

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

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

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

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

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

Parallel-innermost complexity: Example 1

TRS $\mathcal{R}$:

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

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

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

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

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

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

Parallel-innermost complexity: Example 1

TRS $\mathcal{R}$:

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

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

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

$\Rightarrow$ 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:

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

⇒ function calls for last rule **in parallel**

⇒ consider them **separately**, get the maximum of the costs

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

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

Theorem (Baudon, Fuhs, Gonnord, 2022)

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

Sequential complexity: Example 2

TRS $\mathcal{R}$:

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

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

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

Sequential complexity: Example 2

TRS $\mathcal{R}$:

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

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

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

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

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

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**

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

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

Parallel-innermost complexity: Example 2

TRS $\mathcal{R}$:

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

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

Parallel-innermost complexity: Example 2

TRS $\mathcal{R}$:

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

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

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

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

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

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

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

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

Parallel-innermost complexity: Example 2

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

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

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

Parallel-innermost complexity: Example 2

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

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

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

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

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

Parallel-innermost complexity: Example 2

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

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

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

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

$\Rightarrow$ 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, 2022)

Let $\mathcal{R}$ be a TRS with $|\mathit{PDT}(\mathcal{R})| = |\mathcal{R}|$. Then:

1. $\mathit{PDT}(\mathcal{R}) = \mathit{DT}(\mathcal{R})$
2. *from basic terms* $f(t_1, \ldots, t_n)$, $\mathrel{\rightarrow\!\!\!\!\!\!\!+\!\!\!\!+}^{\mathsf{i}}_{\mathcal{R}} \;=\; \xrightarrow{\mathsf{i}}_{\mathcal{R}}$
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, 2022)

Let $\mathcal{R}$ be a TRS with $|\mathit{PDT}(\mathcal{R})| = |\mathcal{R}|$. Then:

1. $\mathit{PDT}(\mathcal{R}) = \mathit{DT}(\mathcal{R})$
2. *from basic terms* $f(t_1, \ldots, t_n)$, $\mathrel{\stackrel{\mathsf{i}}{+\!\!\!\!\!\!\parallel\!\!\!\rightarrow}}_{\mathcal{R}} \; = \; \stackrel{\mathsf{i}}{\rightarrow}_{\mathcal{R}}$
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, 2022)

Let $\mathcal{R}$ be a TRS with $|\mathit{PDT}(\mathcal{R})| = |\mathcal{R}|$. Then:

1. $\mathit{PDT}(\mathcal{R}) = \mathit{DT}(\mathcal{R})$
2. *from basic terms* $f(t_1, \ldots, t_n)$, $\mathrel{+\!\!\!\!|\!|\!\xrightarrow{\mathrm{i}}}_{\mathcal{R}} \;=\; \xrightarrow{\mathrm{i}}_{\mathcal{R}}$
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! $\square$

$\Rightarrow$ Check $|\mathbf{PDT}(\mathcal{R})| = |\mathcal{R}|$ to refute that parallel evaluation is feasible.

Lower bounds on worst-case parallel complexity? (1/3)

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? (1/3)

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)

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

Lower bounds on worst-case parallel complexity? (1/3)

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)

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

Lower bounds on worst-case parallel complexity? (1/3)

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)

$\mathbf{\mathit{irc}}_{\mathcal{R}}$ = *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 $\mathbf{pirc}_{\mathcal{R}}$?

Lower bounds on worst-case parallel complexity? (1/3)

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, 2022)

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

Lower bounds on worst-case parallel complexity? (2/3)

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

Lower bounds on worst-case parallel complexity? (2/3)

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

Need:

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

Lower bounds on worst-case parallel complexity? (2/3)

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

Need:

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

Ad 1: Finding lower bounds for DT problem

Lower bounds on worst-case parallel complexity? (2/3)

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

Need:

1. Find lower bounds for DT problem
2. Prove confluence of $\xrightarrow{\mathsf{i}}\!\!\!\!\!\!\!\Vert\;\;_{\mathcal{R}}$

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? (2/3)

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

Need:

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

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, 2022)

pirc$_{\mathcal{R}}$ $=$ *(sequential) innermost complexity of relative TRS* $\mathbf{PDT}(\mathcal{R})/\mathcal{R}$ ***if*** $\overset{\mathsf{i}}{\twoheadrightarrow\!\!\!\!\!\!\;\!\!\!\!\shortparallel}_{\mathcal{R}}$ ***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? (3/3)

Ad 2: Confluence of $\xrightarrow{\parallel \mathrm{i}}_{\mathcal{R}}$:

Lower bounds on worst-case parallel complexity? (3/3)

Ad 2: Confluence of $\mathrel{\overset{\mathsf{i}}{+\!\!\!\|\!\!\!\to}}_{\mathcal{R}}$:

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

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

Lower bounds on worst-case parallel complexity? (3/3)

Ad 2: Confluence of $\mathrel{+\!\!\!\!\Vert\!\!\xrightarrow{\mathsf{i}}}_{\mathcal{R}}$:

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 $\to_{\mathcal{R}}$ But. . .

a

Lower bounds on worst-case parallel complexity? (3/3)

Ad 2: Confluence of $\mathrel{+\!\!\!\Vert\!\!\xrightarrow{\mathsf{i}}}_{\mathcal{R}}$:

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 $\to_{\mathcal{R}}$ But...

$$\mathsf{a} \mathrel{+\!\!\!\Vert\!\!\xrightarrow{\mathsf{i}}}_{\mathcal{R}} \mathsf{f}(\mathsf{b}, \mathsf{c})$$

Lower bounds on worst-case parallel complexity? (3/3)

Ad 2: Confluence of $\xrightarrow{\parallel\, \mathsf{i}}_{\mathcal{R}}$:

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 $\to_{\mathcal{R}}$ But...

$$\mathsf{f}(\mathsf{b}, \mathsf{b}) \;{}_{\mathcal{R}}\!\xleftarrow{\mathsf{i}\,\parallel}\; \mathsf{a} \;\xrightarrow{\parallel\,\mathsf{i}}_{\mathcal{R}}\; \mathsf{f}(\mathsf{b}, \mathsf{c})$$

Lower bounds on worst-case parallel complexity? (3/3)

Ad 2: Confluence of $\overset{\mathsf{i}}{\twoheadrightarrow\!\!\!\!\shortmid\!\shortmid}_{\mathcal{R}}$:

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

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

$\mathsf{f}(\mathsf{b}, \mathsf{b})\ {}_{\mathcal{R}}\overset{\mathsf{i}}{\leftarrow\!\shortmid\!\shortmid}\ \mathsf{a}\ \overset{\mathsf{i}}{\shortmid\!\shortmid\!\!\rightarrow}_{\mathcal{R}} \mathsf{f}(\mathsf{b}, \mathsf{c})\ \overset{\mathsf{i}}{\shortmid\!\shortmid\!\!\rightarrow}_{\mathcal{R}} \mathsf{f}(\mathsf{c}, \mathsf{b})\ \overset{\mathsf{i}}{\shortmid\!\shortmid\!\!\rightarrow}_{\mathcal{R}} \mathsf{f}(\mathsf{b}, \mathsf{c})\ \overset{\mathsf{i}}{\shortmid\!\shortmid\!\!\rightarrow}_{\mathcal{R}} \dots$

Lower bounds on worst-case parallel complexity? (3/3)

Ad 2: Confluence of $\mathrel{\not\mathrel{\|}\!\xrightarrow{\mathsf{i}}}_{\mathcal{R}}$:

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 $\to_{\mathcal{R}}$ But...

$$\ldots \; {}_{\mathcal{R}}\!\xleftarrow{\mathsf{i}}\!\!\!\!\|\; \mathsf{f}(\mathsf{b},\mathsf{b}) \; {}_{\mathcal{R}}\!\xleftarrow{\mathsf{i}}\!\!\!\!\|\; \mathsf{f}(\mathsf{c},\mathsf{c}) \; {}_{\mathcal{R}}\!\xleftarrow{\mathsf{i}}\!\!\!\!\|\; \mathsf{f}(\mathsf{b},\mathsf{b}) \; {}_{\mathcal{R}}\!\xleftarrow{\mathsf{i}}\!\!\!\!\|\; \mathsf{a} \; \|\!\!\!\!\xrightarrow{\mathsf{i}}_{\mathcal{R}} \mathsf{f}(\mathsf{b},\mathsf{c}) \; \|\!\!\!\!\xrightarrow{\mathsf{i}}_{\mathcal{R}} \mathsf{f}(\mathsf{c},\mathsf{b}) \; \|\!\!\!\!\xrightarrow{\mathsf{i}}_{\mathcal{R}} \mathsf{f}(\mathsf{b},\mathsf{c}) \; \|\!\!\!\!\xrightarrow{\mathsf{i}}_{\mathcal{R}} \ldots$$

Lower bounds on worst-case parallel complexity? (3/3)

Ad 2: Confluence of $\mathrel{\|\!\!\xrightarrow{\mathsf{i}}}_{\mathcal{R}}$:

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 $\to_{\mathcal{R}}$ But...

$$\ldots \;{}_{\mathcal{R}}\mathrel{\xleftarrow{\mathsf{i}}\!\!\|}\; \mathsf{f}(\mathsf{b}, \mathsf{b}) \;{}_{\mathcal{R}}\mathrel{\xleftarrow{\mathsf{i}}\!\!\|}\; \mathsf{f}(\mathsf{c}, \mathsf{c}) \;{}_{\mathcal{R}}\mathrel{\xleftarrow{\mathsf{i}}\!\!\|}\; \mathsf{f}(\mathsf{b}, \mathsf{b}) \;{}_{\mathcal{R}}\mathrel{\xleftarrow{\mathsf{i}}\!\!\|}\; \mathsf{a} \mathrel{\|\!\!\xrightarrow{\mathsf{i}}}_{\mathcal{R}} \mathsf{f}(\mathsf{b}, \mathsf{c}) \mathrel{\|\!\!\xrightarrow{\mathsf{i}}}_{\mathcal{R}} \mathsf{f}(\mathsf{c}, \mathsf{b}) \mathrel{\|\!\!\xrightarrow{\mathsf{i}}}_{\mathcal{R}} \mathsf{f}(\mathsf{b}, \mathsf{c}) \mathrel{\|\!\!\xrightarrow{\mathsf{i}}}_{\mathcal{R}} \ldots$$

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.

Lower bounds on worst-case parallel complexity? (3/3)

Ad 2: Confluence of $\mathrel{+\!\!\!\!\!\Vert\!\!\!\xrightarrow{\mathsf{i}}}_{\mathcal{R}}$:

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

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

$\ldots\ {}_{\mathcal{R}}\!\xleftarrow{\mathsf{i}}\!\!\!\!\Vert\!\!\!+\ \mathsf{f}(\mathsf{b}, \mathsf{b})\ {}_{\mathcal{R}}\!\xleftarrow{\mathsf{i}}\!\!\!\!\Vert\!\!\!+\ \mathsf{f}(\mathsf{c}, \mathsf{c})\ {}_{\mathcal{R}}\!\xleftarrow{\mathsf{i}}\!\!\!\!\Vert\!\!\!+\ \mathsf{f}(\mathsf{b}, \mathsf{b})\ {}_{\mathcal{R}}\!\xleftarrow{\mathsf{i}}\!\!\!\!\Vert\!\!\!+\ \mathsf{a}\ \mathrel{+\!\!\!\!\!\Vert\!\!\!\xrightarrow{\mathsf{i}}}_{\mathcal{R}} \mathsf{f}(\mathsf{b}, \mathsf{c})\ \mathrel{+\!\!\!\!\!\Vert\!\!\!\xrightarrow{\mathsf{i}}}_{\mathcal{R}} \mathsf{f}(\mathsf{c}, \mathsf{b})\ \mathrel{+\!\!\!\!\!\Vert\!\!\!\xrightarrow{\mathsf{i}}}_{\mathcal{R}} \mathsf{f}(\mathsf{b}, \mathsf{c})\ \mathrel{+\!\!\!\!\!\Vert\!\!\!\xrightarrow{\mathsf{i}}}_{\mathcal{R}} \ldots$

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.

$\mathrel{+\!\!\!\!\!\Vert\!\!\!\xrightarrow{\mathsf{i}}}_{\mathcal{R}}$ fixes the used redexes: all innermost redexes at the same time

Lower bounds on worst-case parallel complexity? (3/3)

Ad 2: Confluence of $\mathrel{+\!\!\!\|\!\!\xrightarrow{\mathsf{i}}}_{\mathcal{R}}$:

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 $\to_{\mathcal{R}}$ But...

$$\ldots\ {}_{\mathcal{R}}\!\xleftarrow{\mathsf{i}}\!\!\!\|\!\!+\ \mathsf{f}(\mathsf{b},\mathsf{b})\ {}_{\mathcal{R}}\!\xleftarrow{\mathsf{i}}\!\!\!\|\!\!+\ \mathsf{f}(\mathsf{c},\mathsf{c})\ {}_{\mathcal{R}}\!\xleftarrow{\mathsf{i}}\!\!\!\|\!\!+\ \mathsf{f}(\mathsf{b},\mathsf{b})\ {}_{\mathcal{R}}\!\xleftarrow{\mathsf{i}}\!\!\!\|\!\!+\ \mathsf{a}\ \mathrel{+\!\!\!\|\!\!\xrightarrow{\mathsf{i}}}_{\mathcal{R}} \mathsf{f}(\mathsf{b},\mathsf{c}) \mathrel{+\!\!\!\|\!\!\xrightarrow{\mathsf{i}}}_{\mathcal{R}} \mathsf{f}(\mathsf{c},\mathsf{b}) \mathrel{+\!\!\!\|\!\!\xrightarrow{\mathsf{i}}}_{\mathcal{R}} \mathsf{f}(\mathsf{b},\mathsf{c}) \mathrel{+\!\!\!\|\!\!\xrightarrow{\mathsf{i}}}_{\mathcal{R}} \ldots$$

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.

$\mathrel{+\!\!\!\|\!\!\xrightarrow{\mathsf{i}}}_{\mathcal{R}}$ fixes the used redexes: all innermost redexes at the same time

Corollary (Baudon, Fuhs, Gonnord, 2022)

If $\mathcal{R}$ is non-overlapping, $\mathrel{+\!\!\!\|\!\!\xrightarrow{\mathsf{i}}}_{\mathcal{R}}$ is confluent.

Related Work

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

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

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

Implementation and Experiments (1/3)

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

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

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 $\mathcal{R}$ where $|\mathbf{PDT}(\mathcal{R})| = |\mathcal{R}|$: no parallelism!

$\Rightarrow$ **294** remaining TRSs

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 $\mathcal{R}$ where $|\mathbf{PDT}(\mathcal{R})| = |\mathcal{R}|$: no parallelism!

$\Rightarrow$ **294** remaining TRSs

Timeout per TRS: 300 seconds

Implementation and Experiments (2/3)

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

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

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

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 $\mathcal{O}(n^1)$

Implementation and Experiments (3/3)

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

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

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

Together:

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

Conclusion

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

Conclusion

- First approach to finding bounds on **parallel-**innermost runtime complexity of TRSs
- Works for upper **and** lower bounds

Conclusion

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

Conclusion

- First approach to finding bounds on parallel-innermost runtime complexity of TRSs
- Works for upper **and** lower bounds
- Builds on Dependency Tuples available in AProVE and TcT
- T. Baudon, C. Fuhs, L. Gonnord
  Parallel Complexity of Term Rewriting Systems
  In *Proc. 17th Workshop on Termination*, pages 39 – 44, 2021.

Conclusion

- First approach to finding bounds on parallel-innermost runtime complexity of TRSs
- Works for upper **and** lower bounds
- Builds on Dependency Tuples available in AProVE and TcT
- T. Baudon, C. Fuhs, L. Gonnord
  Parallel Complexity of Term Rewriting Systems
  In *Proc. 17th Workshop on Termination*, pages 39 – 44, 2021.
- Evaluation page with details of experiments:

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

Conclusion

- First approach to finding bounds on parallel-innermost runtime complexity of TRSs
- Works for upper **and** lower bounds
- Builds on Dependency Tuples available in AProVE and TcT
- T. Baudon, C. Fuhs, L. Gonnord
  Parallel Complexity of Term Rewriting Systems
  In *Proc. 17th Workshop on Termination*, pages 39 – 44, 2021.
- Evaluation page with details of experiments:

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

- Conference paper: under submission

Conclusion

- First approach to finding bounds on parallel-innermost runtime complexity of TRSs
- Works for upper **and** lower bounds
- Builds on Dependency Tuples available in AProVE and TcT
- T. Baudon, C. Fuhs, L. Gonnord
  Parallel Complexity of Term Rewriting Systems
  In *Proc. 17th Workshop on Termination*, pages 39 – 44, 2021.
- Evaluation page with details of experiments:

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

- Conference paper: under submission

Thanks a lot for your attention!