Automated Complexity Analysis of Term Rewrite Systems

Martin Avanzini (martin.avanzini@inria.fr)

ISR 2019, July 5–6, 2019

Today's Lecture

From Theory to Automation

1. complexity pairs and relative rewriting
2. dependency pairs for complexity analysis
3. case study: TCT, its complexity framework

Applications to Program Analysis

4. case study: higher-order functional programs

Experimental Evaluation

Input	#rules	orders	TcT
appendAll	12	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
bfs	57	?	$\mathcal{O}(n)$
bft mmult	59	?	$\mathcal{O}(n^3)$
bitonic	78	?	$\mathcal{O}(n^4)$
bitvectors	148	?	$\mathcal{O}(n^2)$
clevermmult	39	?	$\mathcal{O}(n^2)$
duplicates	37	?	$\mathcal{O}(n^2)$
dyade	31	?	$\mathcal{O}(n^2)$
eratosthenes	74	?	$\mathcal{O}(n^2)$
flatten	31	?	$\mathcal{O}(n^2)$
insertionsort	36	?	$\mathcal{O}(n^2)$
listsort	56	?	$\mathcal{O}(n^2)$
lcs	87	?	$\mathcal{O}(n^2)$
matrix	74	?	$\mathcal{O}(n^3)$
mergesort	35	?	$\mathcal{O}(n^3)$
minsort	26	?	$\mathcal{O}(n^2)$
queue	35	?	$\mathcal{O}(n^5)$
quicksort	46	?	$\mathcal{O}(n^2)$
rationalPotential	14	$\mathcal{O}(n)$	$\mathcal{O}(n)$
splitandsort	70	?	$\mathcal{O}(n^3)$
subtrees	8	?	$\mathcal{O}(n^2)$
tuples	33	?	?

Figure: Analysis of translated resource aware ML programs.

Towards a Modular Analysis

- complexity pairs and relative rewriting
- weak dependency pairs/dependency tuples
- safe reduction pairs

Complexity Analysis via Relative Rewriting

Definition (relative reduction relation)

- for to ARSs $\to$ and $\rightsquigarrow$ over carrier A, define

$$\to / \rightsquigarrow \triangleq \rightsquigarrow^* \cdot \to \cdot \rightsquigarrow^* .$$

- for two TRSs $\mathcal{R}$ and $\mathcal{S}$,

$$\to_{\mathcal{R}/\mathcal{S}} \triangleq \to_{\mathcal{R}} / \to_{\mathcal{S}} \qquad \xrightarrow{\mathsf{i}}_{\mathcal{R}/\mathcal{S}} \triangleq \xrightarrow{\mathcal{R} \cup \mathcal{S}}_{\mathcal{R}} / \xrightarrow{\mathcal{R} \cup \mathcal{S}}_{\mathcal{S}}$$

  - $C[\mathtt{f}(l_1\sigma, \dots, l_k\sigma)] \xrightarrow{\mathcal{Q}}_{\mathcal{R}} C[r\sigma]$ if $\mathtt{f}(l_1, \dots, l_k) \to r \in \mathcal{R}$ and $l_i\sigma \in \mathsf{NF}(\to_{\mathcal{Q}})$.

Complexity Analysis via Relative Rewriting

Definition (relative reduction relation)

- ⋆ for to ARSs $\to$ and $\rightsquigarrow$ over carrier A, define

$$\to / \rightsquigarrow \triangleq \rightsquigarrow^* \cdot \to \cdot \rightsquigarrow^* .$$

- ⋆ for two TRSs $\mathcal{R}$ and $\mathcal{S}$,

$$\to_{\mathcal{R}/\mathcal{S}} \triangleq \to_{\mathcal{R}} / \to_{\mathcal{S}} \qquad \xrightarrow{\mathsf{i}}_{\mathcal{R}/\mathcal{S}} \triangleq \xrightarrow{\mathcal{R} \cup \mathcal{S}}_{\mathcal{R}} / \xrightarrow{\mathcal{R} \cup \mathcal{S}}_{\mathcal{S}}$$

  - $C[\mathtt{f}(l_1\sigma, \dots, l_k\sigma)] \xrightarrow{\mathcal{Q}}_{\mathcal{R}} C[r\sigma]$ if $\mathtt{f}(l_1, \dots, l_k) \to r \in \mathcal{R}$ and $l_i\sigma \in \mathsf{NF}(\to_{\mathcal{Q}})$.

Theorem

$$\mathsf{dc}_{\to \cup \rightsquigarrow, S} \leqslant \mathsf{dc}_{\to / \rightsquigarrow, S} + \mathsf{dc}_{\rightsquigarrow / \to, S} .$$

Complexity Analysis via Relative Rewriting

Definition (relative reduction relation)

- for to ARSs $\rightarrow$ and $\rightsquigarrow$ over carrier A, define

$$\rightarrow / \rightsquigarrow \triangleq \rightsquigarrow^* \cdot \rightarrow \cdot \rightsquigarrow^* .$$

- for two TRSs $\mathcal{R}$ and $\mathcal{S}$,

$$\rightarrow_{\mathcal{R}/\mathcal{S}} \triangleq \rightarrow_{\mathcal{R}} / \rightarrow_{\mathcal{S}} \qquad \xrightarrow{\mathsf{i}}_{\mathcal{R}/\mathcal{S}} \triangleq \xrightarrow{\mathcal{R}\cup\mathcal{S}}_{\mathcal{R}} / \xrightarrow{\mathcal{R}\cup\mathcal{S}}_{\mathcal{S}}$$

  - $C[\mathtt{f}(l_1\sigma, \ldots, l_k\sigma)] \xrightarrow{\mathcal{Q}}_{\mathcal{R}} C[r\sigma]$ if $\mathtt{f}(l_1, \ldots, l_k) \rightarrow r \in \mathcal{R}$ and $l_i\sigma \in \mathsf{NF}(\rightarrow_{\mathcal{Q}})$.

Theorem

$$\mathsf{dc}_{\rightarrow\cup\rightsquigarrow, S} \leqslant \mathsf{dc}_{\rightarrow/\rightsquigarrow, S} + \mathsf{dc}_{\rightsquigarrow/\rightarrow, S} .$$

Example

For $\mathsf{a} \rightarrow \mathsf{b}$ and $\mathsf{a} \rightsquigarrow \mathsf{c}$, $\mathsf{dh}_{\rightarrow\cup\rightsquigarrow}(a) = 1 < 2 = \mathsf{dh}_{\rightarrow/\rightsquigarrow}(a) + \mathsf{dh}_{\rightsquigarrow/\rightarrow}(a)$.

Complexity Pairs

Definition (Zankl & Korp, LMCS'14)

- ★ Complexity pair (CP) is pair $(\succ, \gtrsim)$ of rewrite orders s.t. $\gtrsim \cdot \succ \cdot \gtrsim \subseteq \succ$.
- ★ Compatibility with relative TRS $\mathcal{R}/\mathcal{S}$ if $\mathcal{R} \subseteq \succ$ and $\mathcal{S} \subseteq \gtrsim$.

Complexity Pairs

Definition (Zankl & Korp, LMCS'14)

- ⋆ Complexity pair (CP) is pair $(\succ, \gtrsim)$ of rewrite orders s.t. $\gtrsim \cdot \succ \cdot \gtrsim \subseteq \succ$.
- ⋆ Compatibility with relative TRS $\mathcal{R}/\mathcal{S}$ if $\mathcal{R} \subseteq \succ$ and $\mathcal{S} \subseteq \gtrsim$.

Theorem (Soundness)

If CP $(\succ, \gtrsim)$ compatible with $\mathcal{R}/\mathcal{S}$ then

$$\mathsf{dc}_{\rightarrow_{\mathcal{R}/\mathcal{S}},T}(n) \leq \mathsf{dc}_{\succ,T}(n) \; .$$

Complexity Pairs

Definition (Zankl & Korp, LMCS'14)

- Complexity pair (CP) is pair $(\succ, \gtrsim)$ of rewrite orders s.t. $\gtrsim \cdot \succ \cdot \gtrsim \subseteq \succ$.
- Compatibility with relative TRS $\mathcal{R}/\mathcal{S}$ if $\mathcal{R} \subseteq \succ$ and $\mathcal{S} \subseteq \gtrsim$.

Theorem (Soundness)

If CP $(\succ, \gtrsim)$ compatible with $\mathcal{R}/\mathcal{S}$ then

$$\mathsf{dc}_{\to_{\mathcal{R}/\mathcal{S}},T}(n) \leq \mathsf{dc}_{\succ,T}(n)\ .$$

Theorem (Iterative Complexity Analysis)

If CP $(\succ, \gtrsim)$ compatible with $\mathcal{R}_1/\mathcal{R}_2 \cup \mathcal{S}$ then

$$\mathsf{dc}_{\to_{\mathcal{R}_1 \cup \mathcal{R}_2/\mathcal{S}},T}(n) \leq \mathsf{dc}_{\succ,T}(n) + \mathsf{dc}_{\to_{\mathcal{R}_2/\mathcal{R}_1 \cup \mathcal{S}},T}(n)\ .$$

Complexity Pairs

Definition (Zankl & Korp, LMCS'14)

- ⋆ Complexity pair (CP) is pair $(\succ, \succsim)$ of rewrite orders s.t. $\succsim \cdot \succ \cdot \succsim \subseteq \succ$.
- ⋆ Compatibility with relative TRS $\mathcal{R}/\mathcal{S}$ if $\mathcal{R} \subseteq \succ$ and $\mathcal{S} \subseteq \succsim$.

Theorem (Soundness)

If CP $(\succ, \succsim)$ compatible with $\mathcal{R}/\mathcal{S}$ then

$$\mathsf{dc}_{\rightarrow_{\mathcal{R}/\mathcal{S}}, T}(n) \leq \mathsf{dc}_{\succ, T}(n) .$$

Theorem (Iterative Complexity Analysis)

If CP $(\succ, \succsim)$ compatible with $\mathcal{R}_1/\mathcal{R}_2 \cup \mathcal{S}$ then

$$\mathsf{dc}_{\rightarrow_{\mathcal{R}_1 \cup \mathcal{R}_2/\mathcal{S}}, T}(n) \leq \mathsf{dc}_{\succ, T}(n) + \mathsf{dc}_{\rightarrow_{\mathcal{R}_2/\mathcal{R}_1 \cup \mathcal{S}}, T}(n) .$$

Note: remains valid for rewriting under strategies

Experimental Evaluation

Input	#rules	orders	iterative	TcT
appendAll	12	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
bfs	57	?	?	$\mathcal{O}(n)$
bft mmult	59	?	?	$\mathcal{O}(n^3)$
bitonic	78	?	?	$\mathcal{O}(n^4)$
bitvectors	148	?	?	$\mathcal{O}(n^2)$
clevermmult	39	?	?	$\mathcal{O}(n^2)$
duplicates	37	?	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
dyade	31	?	?	$\mathcal{O}(n^2)$
eratosthenes	74	?	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$
flatten	31	?	?	$\mathcal{O}(n^2)$
insertionsort	36	?	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$
listsort	56	?	?	$\mathcal{O}(n^2)$
lcs	87	?	?	$\mathcal{O}(n^2)$
matrix	74	?	?	$\mathcal{O}(n^3)$
mergesort	35	?	?	$\mathcal{O}(n^3)$
minsort	26	?	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$
queue	35	?	?	$\mathcal{O}(n^5)$
quicksort	46	?	?	$\mathcal{O}(n^2)$
rationalPotential	14	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
splitandsort	70	?	?	$\mathcal{O}(n^3)$
subtrees	8	?	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
tuples	33	?	?	?

Figure: Analysis of translated resource aware ML programs.

Dependency Pairs and RC

Theorem

TRS $\mathcal{R}$ is terminating iff there is no infinite and minimal chain

$$\mathtt{f}^{\#}(s_1, \ldots, s_m) \rightarrow_{\mathtt{DP}(\mathcal{R})/\mathcal{R}} \mathtt{g}^{\#}(t_1, \ldots, t_n) \rightarrow_{\mathtt{DP}(\mathcal{R})/\mathcal{R}} \cdots$$

Dependency Pairs and RC

Theorem

TRS $\mathcal{R}$ is terminating iff there is no infinite and minimal chain

$$\mathtt{f}^{\#}(s_1, \ldots, s_m) \rightarrow_{\mathtt{DP}(\mathcal{R})/\mathcal{R}} \mathtt{g}^{\#}(t_1, \ldots, t_n) \rightarrow_{\mathtt{DP}(\mathcal{R})/\mathcal{R}} \cdots$$

Corollary

TRS $\mathcal{R}$ is terminating on $\mathcal{B}$ iff

$$\forall n \in \mathbb{N}.\ \mathrm{rc}^{\#}_{\mathtt{DP}(\mathcal{R})/\mathcal{R}}(n) \triangleq \mathrm{dc}_{\rightarrow_{\mathtt{DP}(\mathcal{R})/\mathcal{R}}, \mathcal{B}^{\#}}(n) \in \mathbb{N}\,.$$

Dependency Pairs and RC

Theorem

TRS $\mathcal{R}$ is terminating iff there is no infinite and minimal chain

$$\mathtt{f}^{\#}(s_1, \ldots, s_m) \to_{\mathtt{DP}(\mathcal{R})/\mathcal{R}} \mathtt{g}^{\#}(t_1, \ldots, t_n) \to_{\mathtt{DP}(\mathcal{R})/\mathcal{R}} \cdots$$

Corollary

TRS $\mathcal{R}$ is terminating on $\mathcal{B}$ iff

$$\forall n \in \mathbb{N}.\ \mathsf{rc}^{\#}_{\mathtt{DP}(\mathcal{R})/\mathcal{R}}(n) \triangleq \mathsf{dc}_{\to_{\mathtt{DP}(\mathcal{R})/\mathcal{R}}, \mathcal{B}^{\#}}(n) \in \mathbb{N}\,.$$

Pros:

1. gets rid of nasty monotonicity requirements
2. DP framework enables true modular analysis

Dependency Pairs and RC

Theorem

TRS $\mathcal{R}$ is terminating iff there is no infinite and minimal chain

$$\mathtt{f}^\#(s_1,\dots,s_m) \rightarrow_{\mathtt{DP}(\mathcal{R})/\mathcal{R}} \mathtt{g}^\#(t_1,\dots,t_n) \rightarrow_{\mathtt{DP}(\mathcal{R})/\mathcal{R}} \cdots$$

Corollary

TRS $\mathcal{R}$ is terminating on $\mathcal{B}$ iff

$$\forall n \in \mathbb{N}.\ \mathrm{rc}^\#_{\mathtt{DP}(\mathcal{R})/\mathcal{R}}(n) \triangleq \mathrm{dc}_{\rightarrow_{\mathtt{DP}(\mathcal{R})/\mathcal{R}},\mathcal{B}^\#}(n) \in \mathbb{N}\,.$$

Pros:

1. gets rid of nasty monotonicity requirements
2. DP framework enables true modular analysis

Questions:

1. is there a "small" $f \colon \mathbb{N} \to \mathbb{N}$ s.t. $\mathrm{rc}_{\mathcal{R}}(n) \leq f(\mathrm{rc}^\#_{\mathtt{DP}(\mathcal{R})/\mathcal{R}}(n))$?
2. what about techniques from the DP framework?

Dependency Pairs and RC (II)

Example

Consider $\mathcal{R}$

$$\mathtt{f}(\mathtt{s}(x)) \to \mathtt{s}(\mathtt{f}(\mathtt{f}(x))) \qquad \mathtt{f}(x) \to \mathtt{dup}(x, x)\ ,$$

with $\mathsf{DP}(\mathcal{R})$

$$\mathtt{f}^\#(\mathtt{s}(x)) \to \mathtt{f}^\#(\mathtt{f}(x)) \qquad \mathtt{f}^\#(\mathtt{s}(x)) \to \mathtt{f}^\#(x)\ .$$

Then $\mathsf{rc}^{\#}_{\mathsf{DP}(\mathcal{R})/\mathcal{R}}$ is linear whereas $\mathsf{rc}_{\mathcal{R}}(n)$ grows double-exponential.

Dependency Pairs and RC (II)

Example

Consider $\mathcal{R}$

$$\mathtt{f}(\mathtt{s}(x)) \to \mathtt{s}(\mathtt{f}(\mathtt{f}(x))) \qquad \mathtt{f}(x) \to \mathtt{dup}(x, x)\ ,$$

with $\mathsf{DP}(\mathcal{R})$

$$\mathtt{f}^{\#}(\mathtt{s}(x)) \to \mathtt{f}^{\#}(\mathtt{f}(x)) \qquad \mathtt{f}^{\#}(\mathtt{s}(x)) \to \mathtt{f}^{\#}(x)\ .$$

Then $\mathsf{rc}^{\#}_{\mathsf{DP}(\mathcal{R})/\mathcal{R}}$ is linear whereas $\mathsf{rc}_{\mathcal{R}}(n)$ grows double-exponential.

Question: Reasons that cause this blow-up?

Dependency Pairs and RC (II)

Example

Consider $\mathcal{R}$

$$\mathtt{f}(\mathtt{s}(x)) \to \mathtt{s}(\mathtt{f}(\mathtt{f}(x))) \qquad \mathtt{f}(x) \to \mathtt{dup}(x, x)\ ,$$

with $\mathsf{DP}(\mathcal{R})$

$$\mathtt{f}^{\#}(\mathtt{s}(x)) \to \mathtt{f}^{\#}(\mathtt{f}(x)) \qquad \mathtt{f}^{\#}(\mathtt{s}(x)) \to \mathtt{f}^{\#}(x)\ .$$

Then $\mathsf{rc}^{\#}_{\mathsf{DP}(\mathcal{R})/\mathcal{R}}$ is linear whereas $\mathsf{rc}_{\mathcal{R}}(n)$ grows double-exponential.

Question: Reasons that cause this blow-up?

1. DPs track single path in “calls graph”
2. DPs do not account for duplication

Weak Dependency Pairs and Dependency Tuples

★ Weak Dependency Pairs WDP($\mathcal{R}$) [Hirokawa & Moser, IJCAR'08]

1. bundle outermost function calls in weak dependency pair

$$\mathtt{f}^{\#}(l_1, \ldots, l_k) \to \mathsf{c}_n(r_1^{\#}, \ldots, r_n^{\#}) \quad \text{for each } \mathtt{f}(l_1, \ldots, l_k) \to C[r_1, \ldots, r_n] \in \mathcal{R}$$

   where C maximal constructor-context

2. impose non-duplication & weight-gap condition

N. Hirokawa and G. Moser. *"Automated Complexity Analysis Based on the Dependency Pair Method"*. In *Proc. of 4th IJCAR*, pp. 364–380, 2008.

Weak Dependency Pairs and Dependency Tuples

⋆ Weak Dependency Pairs WDP($\mathcal{R}$) [Hirokawa & Moser, IJCAR'08]

1. bundle outermost function calls in weak dependency pair

$$\mathtt{f}^{\#}(l_1, \dots, l_k) \to \mathsf{c}_n(r_1^{\#}, \dots, r_n^{\#}) \quad \text{for each } \mathtt{f}(l_1, \dots, l_k) \to C[r_1, \dots, r_n] \in \mathcal{R}$$

   where C maximal constructor-context

2. impose non-duplication & weight-gap condition

⋆ Dependency Pair Tuples DT($\mathcal{R}$) [Noschinksi et al., CADE'11]

1. bundle all function calls in dependency tuple
2. restricted to innermost rewriting

N. Hirokawa and G. Moser. *"Automated Complexity Analysis Based on the Dependency Pair Method"*. In *Proc. of 4th IJCAR*, pp. 364–380, 2008.

L. Noschinski, F. Emmes, and J. Giesl. *"A Dependency Pair Framework for Innermost Complexity Analysis of Term Rewrite Systems"*. In *Proc. of 23rd CADE*, pp. 422–438, 2011.

Dependency Tuples

Definition (dependency tuples, Noschinski et. al, CADE'11)

- ⋆ dependency tuple of $\mathtt{f}(l_1, \ldots, l_m) \to r$ is

$$\mathtt{f}^\#(l_1, \ldots, l_m) \to \mathtt{c_k}(\mathtt{g}_1^\#(\vec{t}_1), \ldots, \mathtt{g}_\mathtt{k}^\#(\vec{t}_k)) \ ,$$

  where $\mathtt{g}_1(\vec{t}_1), \ldots, \mathtt{g}_k(\vec{t}_k)$ are all subterms of r with defined root;

- ⋆ $\mathsf{DT}(\mathcal{R})$ collects DTs of rules in $\mathcal{R}$

L. Noschinski, F. Emmes, and J. Giesl. *"A Dependency Pair Framework for Innermost Complexity Analysis of Term Rewrite Systems"*. In *Proc. of 23rd CADE*, pp. 422–438, 2011.

Dependency Tuples

Definition (dependency tuples, Noschinski et. al, CADE'11)

- ⋆ dependency tuple of $\mathtt{f}(l_1, \ldots, l_m) \to r$ is

$$\mathtt{f}^{\#}(l_1, \ldots, l_m) \to \mathtt{c}_{\mathtt{k}}(\mathtt{g}_{\mathtt{1}}^{\#}(\vec{t}_1), \ldots, \mathtt{g}_{\mathtt{k}}^{\#}(\vec{t}_k)) \; ,$$

  where $\mathtt{g}_1(\vec{t}_1), \ldots, \mathtt{g}_k(\vec{t}_k)$ are all subterms of r with defined root;
- ⋆ $\mathtt{DT}(\mathcal{R})$ collects DTs of rules in $\mathcal{R}$

Example

$\mathcal{R}$	$\mathtt{DT}(\mathcal{R})$
$[\,] +\!\!+ ys \to ys$	$[\,] +\!\!+^{\#} \to \mathtt{c}_0$
$(x :: xs) +\!\!+ ys \to x :: (xs +\!\!+ ys)$	$(x :: xs) +\!\!+^{\#} ys \to \mathtt{c}_1(xs +\!\!+^{\#} ys)$
$\mathtt{rev}([\,]) \to [\,]$	$\mathtt{rev}^{\#}([\,]) \to \mathtt{c}_0$
$\mathtt{rev}(x :: xs) \to \mathtt{rev}(xs) +\!\!+ [x]$	$\mathtt{rev}^{\#}(x :: xs) \to \mathtt{c}_2(\mathtt{rev}(xs) +\!\!+^{\#} [x], \mathtt{rev}^{\#}(xs))$

Dependency Tuples (II)

Lemma

Reduction sequence

$$\mathtt{f}(v_1, \ldots, v_k) \xrightarrow{i}_{\mathcal{R}} t_1 \xrightarrow{i}_{\mathcal{R}} t_2 \xrightarrow{i}_{\mathcal{R}} \ldots,$$

simulated step-wise by reduction

$$\mathtt{f}^{\#}(v_1, \ldots, v_k) \xrightarrow{i}_{\mathtt{DT}(\mathcal{R})/\mathcal{R}} C_1[\vec{s}_1] \xrightarrow{i}_{\mathtt{DT}(\mathcal{R})/\mathcal{R}} C_2[\vec{s}_2] \xrightarrow{i}_{\mathtt{DT}(\mathcal{R})/\mathcal{R}} \ldots,$$

with $\vec{s}_i$ marked innermost redexes in t_i.

Dependency Tuples (III)

Example

Sequence

$$\texttt{rev}([1,2]) \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{rev}}} \underline{\texttt{rev}([3])} +\!\!+ [1] \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{rev}}} (\texttt{rev}([\,]) +\!\!+ [2]) +\!\!+ [1] \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{rev}}} \dots ,$$

translates to

$$\begin{aligned} \texttt{rev}^{\#}([1,2]) &\xrightarrow{\mathsf{i}}_{\mathsf{DT}(\mathcal{R}_{\mathsf{rev}})/\mathcal{R}_{\mathsf{rev}}} C_1[\underline{\texttt{rev}([3])} +\!\!+^{\#} [1], \underline{\texttt{rev}^{\#}([3])}] \\ &\xrightarrow{\mathsf{i}}_{\mathsf{DT}(\mathcal{R}_{\mathsf{rev}})/\mathcal{R}_{\mathsf{rev}}} C_2[(\texttt{rev}([\,]) +\!\!+ [2]) +\!\!+^{\#} [1], \texttt{rev}([\,]) +\!\!+^{\#} [2], \texttt{rev}^{\#}([\,])] \\ &\xrightarrow{\mathsf{i}}_{\mathsf{DT}(\mathcal{R}_{\mathsf{rev}})/\mathcal{R}_{\mathsf{rev}}} \dots . \end{aligned}$$

Dependency Tuples (III)

Example

Sequence

$$\mathtt{rev}([1,2]) \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{rev}}} \underline{\mathtt{rev}([3])} + [1] \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{rev}}} (\mathtt{rev}([\,]) + [2]) + [1] \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{rev}}} \dots ,$$

translates to

$$\begin{aligned} \mathtt{rev}^{\#}([1,2]) &\xrightarrow{\mathsf{i}}_{\mathsf{DT}(\mathcal{R}_{\mathsf{rev}})/\mathcal{R}_{\mathsf{rev}}} C_1[\underline{\mathtt{rev}([3])} +^{\#} [1], \underline{\mathtt{rev}^{\#}([3])}] \\ &\xrightarrow{\mathsf{i}}_{\mathsf{DT}(\mathcal{R}_{\mathsf{rev}})/\mathcal{R}_{\mathsf{rev}}} C_2[(\mathtt{rev}([\,]) + [2]) +^{\#} [1], \mathtt{rev}([\,]) +^{\#} [2], \mathtt{rev}^{\#}([\,])] \\ &\xrightarrow{\mathsf{i}}_{\mathsf{DT}(\mathcal{R}_{\mathsf{rev}})/\mathcal{R}_{\mathsf{rev}}} \cdots . \end{aligned}$$

Theorem (Soundness of DTs (Noschinski et. al, CADE'11))

$$\mathsf{rc}_{\mathcal{R}}(n) \leq \mathsf{rc}^{\#}_{\mathsf{DT}(\mathcal{R})/\mathcal{R}}(n) .$$

Dependency Tuples (III)

Example

Sequence

$$\mathtt{rev}([1,2]) \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{rev}}} \underline{\mathtt{rev}([3])} \mathbin{+\!\!+} [1] \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{rev}}} (\mathtt{rev}([\,]) \mathbin{+\!\!+} [2]) \mathbin{+\!\!+} [1] \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{rev}}} \dots ,$$

translates to

$$\begin{aligned}\mathtt{rev}^{\#}([1,2]) &\xrightarrow{\mathsf{i}}_{\mathtt{DT}(\mathcal{R}_{\mathsf{rev}})/\mathcal{R}_{\mathsf{rev}}} C_1[\underline{\mathtt{rev}([3])} \mathbin{+\!\!+}^{\#} [1], \underline{\mathtt{rev}^{\#}([3])}] \\ &\xrightarrow{\mathsf{i}}_{\mathtt{DT}(\mathcal{R}_{\mathsf{rev}})/\mathcal{R}_{\mathsf{rev}}} C_2[(\mathtt{rev}([\,]) \mathbin{+\!\!+} [2]) \mathbin{+\!\!+}^{\#} [1], \mathtt{rev}([\,]) \mathbin{+\!\!+}^{\#} [2], \mathtt{rev}^{\#}([\,])] \\ &\xrightarrow{\mathsf{i}}_{\mathtt{DT}(\mathcal{R}_{\mathsf{rev}})/\mathcal{R}_{\mathsf{rev}}} \dots .\end{aligned}$$

Theorem (Soundness of DTs (Noschinski et. al, CADE'11))

$$\mathsf{rc}_{\mathcal{R}}(n) \leq \mathsf{rc}^{\#}_{\mathtt{DT}(\mathcal{R})/\mathcal{R}}(n) .$$

Question: What about inverse, i.e., completeness?

Dependency Tuples (III)

Example

Sequence

$$\mathtt{rev}([1,2]) \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{rev}}} \underline{\mathtt{rev}([3])} \mathbin{+\!\!+} [1] \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{rev}}} (\mathtt{rev}([\,]) \mathbin{+\!\!+} [2]) \mathbin{+\!\!+} [1] \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\mathsf{rev}}} \dots ,$$

translates to

$$\begin{aligned}\mathtt{rev}^{\#}([1,2]) &\xrightarrow{\mathsf{i}}_{\mathtt{DT}(\mathcal{R}_{\mathsf{rev}})/\mathcal{R}_{\mathsf{rev}}} C_1[\underline{\mathtt{rev}([3])} \mathbin{+\!\!+}^{\#} [1], \underline{\mathtt{rev}^{\#}([3])}] \\ &\xrightarrow{\mathsf{i}}_{\mathtt{DT}(\mathcal{R}_{\mathsf{rev}})/\mathcal{R}_{\mathsf{rev}}} C_2[(\mathtt{rev}([\,]) \mathbin{+\!\!+} [2]) \mathbin{+\!\!+}^{\#} [1], \mathtt{rev}([\,]) \mathbin{+\!\!+}^{\#} [2], \mathtt{rev}^{\#}([\,])] \\ &\xrightarrow{\mathsf{i}}_{\mathtt{DT}(\mathcal{R}_{\mathsf{rev}})/\mathcal{R}_{\mathsf{rev}}} \dots .\end{aligned}$$

Theorem (Soundness of DTs (Noschinski et. al, CADE'11))

$$\mathsf{rc}_{\mathcal{R}}(n) \leq \mathsf{rc}^{\#}_{\mathtt{DT}(\mathcal{R})/\mathcal{R}}(n) .$$

Question: What about inverse, i.e., completeness?

★ $\mathsf{rc}_{\mathcal{R}}(n) = \mathsf{rc}^{\#}_{\mathtt{DT}(\mathcal{R})/\mathcal{R}}(n)$ if $\mathcal{R}$ is confluent

Safe Reduction Pairs

Definition (Hirokawa & Moser, IJCAR'08)

- Safe reduction pair is pair $(\succ, \gtrsim)$ of orders on terms s.t.
  - $\succ$ is closed under substitutions and monotone on compound symbols c_i introduced by WDPs/DTs
  - $\gtrsim$ is a rewrite order
  - $\gtrsim \cdot \succ \cdot \gtrsim \subseteq \succ$
- compatible with $\mathcal{P}/\mathcal{R}$ if $\mathcal{P} \subseteq \succ$ and $\mathcal{R} \subseteq \gtrsim$.

Safe Reduction Pairs

Definition (Hirokawa & Moser, IJCAR'08)

- ⋆ Safe reduction pair is pair $(\succ, \gtrsim)$ of orders on terms s.t.
  - $\succ$ is closed under substitutions and monotone on compound symbols c_i introduced by WDPs/DTs
  - $\gtrsim$ is a rewrite order
  - $\gtrsim \cdot \succ \cdot \gtrsim \subseteq \succ$
- ⋆ compatible with $\mathcal{P}/\mathcal{R}$ if $\mathcal{P} \subseteq \succ$ and $\mathcal{R} \subseteq \gtrsim$.

Theorem

If $(\succ, \gtrsim)$ compatible with $\mathcal{P}/\mathcal{R}$ then

$$\mathsf{rc}^{\#}_{\mathcal{P}/\mathcal{R}}(n) \leq \mathsf{dc}_{\succ, \mathcal{B}^{\#}} \ .$$

Note: As for complexity pairs, can be applied in iterative way

Experimental Evaluation

Input	#rules	orders	iterative	DT+iterative+simps	TCT
appendAll	12	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
bfs	57	?	?	$\mathcal{O}(n^1)$	$\mathcal{O}(n)$
bft mmult	59	?	?	?	$\mathcal{O}(n^3)$
bitonic	78	?	?	?	$\mathcal{O}(n^4)$
bitvectors	148	?	?	?	$\mathcal{O}(n^2)$
clevermmult	39	?	?	?	$\mathcal{O}(n^2)$
duplicates	37	?	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
dyade	31	?	?	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
eratosthenes	74	?	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
flatten	31	?	?	?	$\mathcal{O}(n^2)$
insertionsort	36	?	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
listsort	56	?	?	?	$\mathcal{O}(n^2)$
lcs	87	?	?	?	$\mathcal{O}(n^2)$
matrix	74	?	?	?	$\mathcal{O}(n^3)$
mergesort	35	?	?	?	$\mathcal{O}(n^3)$
minsort	26	?	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
queue	35	?	?	?	$\mathcal{O}(n^5)$
quicksort	46	?	?	?	$\mathcal{O}(n^2)$
rationalPotential	14	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n^1)$	$\mathcal{O}(n)$
splitandsort	70	?	?	?	$\mathcal{O}(n^3)$
subtrees	8	?	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
tuples	33	?	?	?	?

Figure: Analysis of translated resource aware ML programs.

Case Study: TCT

- ⋆ complexity problems and processors
- ⋆ complexity processors
  - dependency graph decomposition
  - usable rules
  - complexity pairs & relative rewriting

Tyrolean Complexity Tool
History

2008 **version 1.0** *extension to termination prover $\mathrm{T_TT_2}$*

- ⋆ 4 dedicated complexity techniques (POP*, WDPs, safe reduction pairs, usable rules)

2009 **version 1.5** *first dedicated implementation*

- ⋆ 9 methods implemented

2013 **version 2.0** *Gödel award at FLOC Olympic Games*

- ⋆ 23 methods implemented
- ⋆ modular complexity framework

2015 **version 3.3** *current version*

- ⋆ certification support through CeTA
- ⋆ frontends for functional and imperative programs

Complexity Framework Underlying TCT

1. complexity problem is tuple $\mathcal{P} = \langle \mathcal{S}, \mathcal{W}, \mathcal{Q}, \mathcal{T} \rangle$
   - $\mathcal{S}$, $\mathcal{W}$ and $\mathcal{Q}$ define rewrite relation $\xrightarrow{\mathcal{Q}}_{\mathcal{S} \cup \mathcal{W}}$ of $\mathcal{P}$
   - $\mathcal{T}$ is set of starting terms

Complexity Framework Underlying TCT

1. complexity problem is tuple $\mathcal{P} = \langle \mathcal{S}, \mathcal{W}, \mathcal{Q}, \mathcal{T} \rangle$
   - $\mathcal{S}$, $\mathcal{W}$ and $\mathcal{Q}$ define rewrite relation $\xrightarrow{\mathcal{Q}}_{\mathcal{S} \cup \mathcal{W}}$ of $\mathcal{P}$
   - $\mathcal{T}$ is set of starting terms

2. complexity function of $\mathcal{P}$ is

$$\mathsf{cp}_{\mathcal{P}}(n) \triangleq \mathsf{dc}_{\xrightarrow{\mathcal{Q}}_{\mathcal{S}/\mathcal{W}}, \mathcal{T}}(n) ,$$

Complexity Framework Underlying TCT

1. complexity problem is tuple $\mathcal{P} = \langle \mathcal{S}, \mathcal{W}, \mathcal{Q}, \mathcal{T} \rangle$
   - $\mathcal{S}$, $\mathcal{W}$ and $\mathcal{Q}$ define rewrite relation $\xrightarrow{\mathcal{Q}}_{\mathcal{S} \cup \mathcal{W}}$ of $\mathcal{P}$
   - $\mathcal{T}$ is set of starting terms
2. complexity function of $\mathcal{P}$ is

$$\mathsf{cp}_{\mathcal{P}}(n) \triangleq \mathsf{dc}_{\xrightarrow{\mathcal{Q}}_{\mathcal{S}/\mathcal{W}}, \mathcal{T}}(n) ,$$

3. complexity processor is inference rule

$$\frac{\vdash \mathcal{P}_1 : f_1 \qquad \cdots \qquad \vdash \mathcal{P}_n : f_n}{\vdash \mathcal{P} : f}$$

   - judgement $\vdash \mathcal{P} : f$ valid if $\mathsf{cp}_{\mathcal{P}}(n) \in \mathcal{O}(f(n))$
   - processor sound if validity of judgements preserved

Complexity Framework Underlying TCT

1. complexity problem is tuple $\mathcal{P} = \langle \mathcal{S}, \mathcal{W}, \mathcal{Q}, \mathcal{T} \rangle$
   - $\mathcal{S}$, $\mathcal{W}$ and $\mathcal{Q}$ define rewrite relation $\xrightarrow{\mathcal{Q}}_{\mathcal{S} \cup \mathcal{W}}$ of $\mathcal{P}$
   - $\mathcal{T}$ is set of starting terms
2. complexity function of $\mathcal{P}$ is

$$\mathsf{cp}_{\mathcal{P}}(n) \triangleq \mathsf{dc}_{\xrightarrow{\mathcal{Q}}_{\mathcal{S}/\mathcal{W}}, \mathcal{T}}(n) ,$$

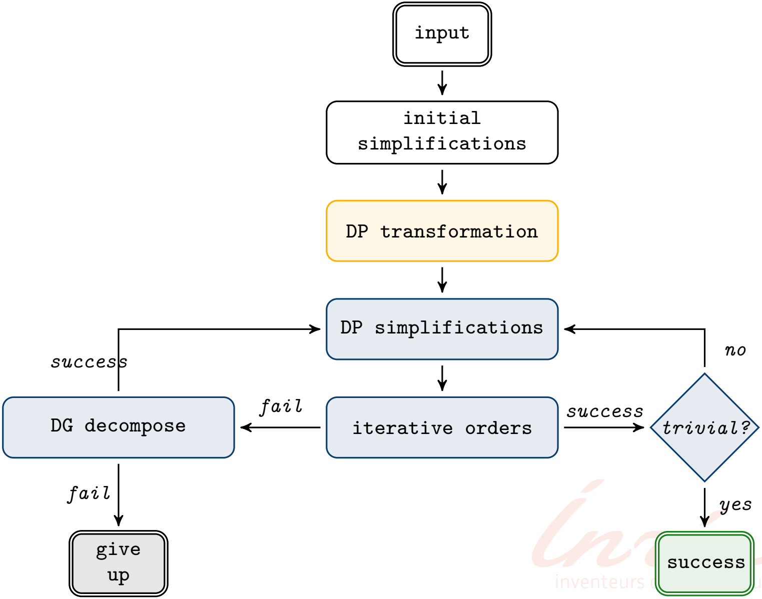
3. complexity processor is inference rule

$$\frac{\vdash \mathcal{P}_1 : f_1 \qquad \cdots \qquad \vdash \mathcal{P}_n : f_n}{\vdash \mathcal{P} : f}$$

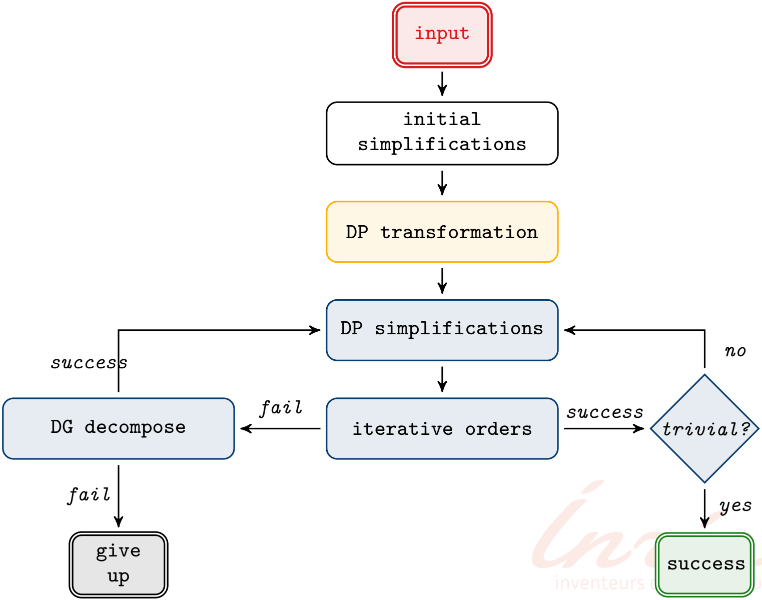
   - judgement $\vdash \mathcal{P} : f$ valid if $\mathsf{cp}_{\mathcal{P}}(n) \in \mathcal{O}(f(n))$
   - processor sound if validity of judgements preserved
4. complexity proof is deduction using sound processors and axiom

$$\vdash \langle \varnothing, \mathcal{W}, \mathcal{Q}, \mathcal{T} \rangle : f$$

Runtime Complexity Proof Search in TCT



Runtime Complexity Proof Search in TCT



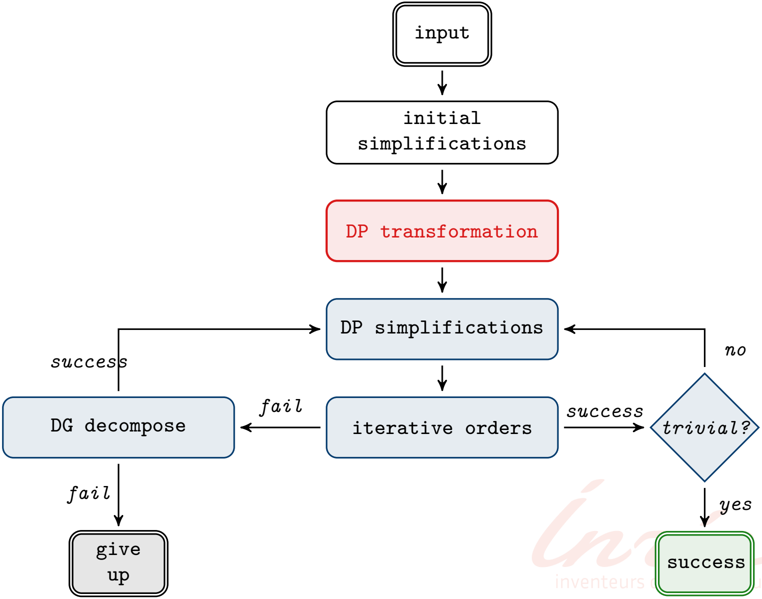
Canonical Complexity Problems

Definition (canonical complexity problem)

Let $\mathcal{R}$ be a TRS over terms $\mathcal{T}$ and basic terms $\mathcal{B}$

	full	innermost
derivational	$\langle \mathcal{R}, \varnothing, \varnothing, \mathcal{T} \rangle$	$\langle \mathcal{R}, \varnothing, \mathcal{R}, \mathcal{T} \rangle$
runtime	$\langle \mathcal{R}, \varnothing, \varnothing, \mathcal{B} \rangle$	$\langle \mathcal{R}, \varnothing, \mathcal{R}, \mathcal{B} \rangle$

Runtime Complexity Proof Search in TCT



Dependency Tuples in TCT

Theorem (Dependency Tuple Transformation)

The following processor is sound

$$\frac{\vdash \langle \mathsf{DT}(\mathcal{S}), \mathsf{DT}(\mathcal{W}) \cup \mathcal{S} \cup \mathcal{W}, \mathcal{Q}, \mathcal{B}^{\#} \rangle : f \quad \mathsf{NF}(\mathcal{Q}) \subseteq \mathsf{NF}(\mathcal{S} \cup \mathcal{W})}{\vdash \langle \mathcal{S}, \mathcal{W}, \mathcal{Q}, \mathcal{B} \rangle : f} \; \mathsf{DT}$$

Example: Initial IRC Problem

current: $\langle \mathcal{S}, \mathcal{W}, \mathcal{Q}, \mathcal{B} \rangle$

$\mathcal{S}$

$$[\,] +\!\!+\ ys \to ys \qquad \mathtt{rev}([\,]) \to [\,]$$
$$(x :: xs) +\!\!+\ ys \to x :: (xs +\!\!+\ ys) \qquad \mathtt{rev}(x :: xs) \to \mathtt{rev}(xs) +\!\!+\ [x]$$

$\mathcal{W}$

$$\varnothing$$

$\mathcal{Q}$

$$[\,] +\!\!+\ ys \to ys \qquad \mathtt{rev}([\,]) \to [\,]$$
$$(x :: xs) +\!\!+\ ys \to x :: (xs +\!\!+\ ys) \qquad \mathtt{rev}(x :: xs) \to \mathtt{rev}(xs) +\!\!+\ [x]$$

Example: DT Transformation

current: $\langle \mathcal{S}, \mathcal{W}, \mathcal{Q}, \mathcal{B}^{\#} \rangle$

$\mathcal{S}$

$$[\,] +\!\!+^{\#}\ ys \to \mathtt{c}_0 \qquad \mathtt{rev}^{\#}([\,]) \to \mathtt{c}_0$$

$$(x :: xs) +\!\!+^{\#}\ ys \to \mathtt{c}_1(xs +\!\!+^{\#}\ ys) \qquad \mathtt{rev}^{\#}(x :: xs) \to \mathtt{c}_2(\mathtt{rev}(xs) +\!\!+^{\#}\ [x], \mathtt{rev}^{\#}(xs))$$

$\mathcal{W}$

$$[\,] +\!\!+\ ys \to ys \qquad \mathtt{rev}([\,]) \to [\,]$$

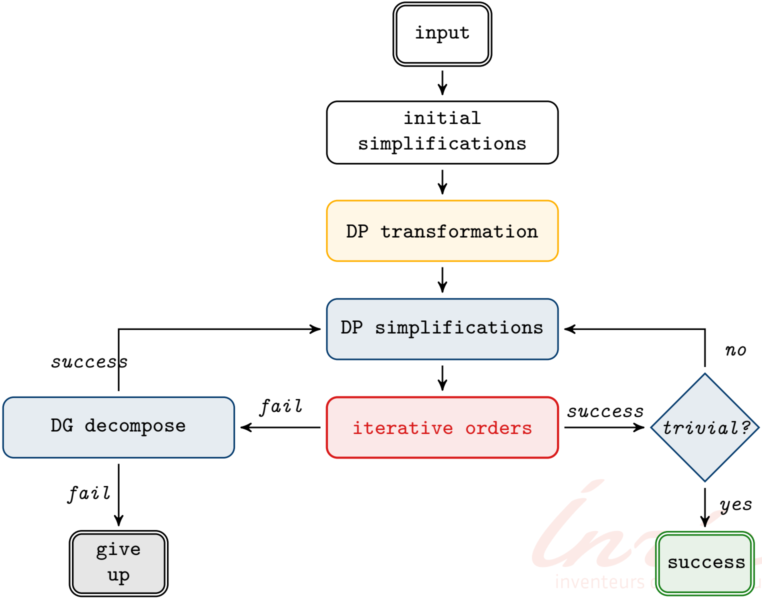
$$(x :: xs) +\!\!+\ ys \to x :: (xs +\!\!+\ ys) \qquad \mathtt{rev}(x :: xs) \to \mathtt{rev}(xs) +\!\!+\ [x]$$

$\mathcal{Q}$

$$[\,] +\!\!+\ ys \to ys \qquad \mathtt{rev}([\,]) \to [\,]$$

$$(x :: xs) +\!\!+\ ys \to x :: (xs +\!\!+\ ys) \qquad \mathtt{rev}(x :: xs) \to \mathtt{rev}(xs) +\!\!+\ [x]$$

Runtime Complexity Proof Search in TCT



Complexity Pairs & Relative Rewriting

Theorem (Relative Decomposition Processor)

The following processor is sound:

$$\frac{\vdash \langle \mathcal{S}_1, \mathcal{S}_2 \cup \mathcal{W}, \mathcal{Q}, \mathcal{T} \rangle \colon f \qquad \vdash \langle \mathcal{S}_2, \mathcal{S}_1 \cup \mathcal{W}, \mathcal{Q}, \mathcal{T} \rangle \colon g}{\vdash \langle \mathcal{S}_1 \cup \mathcal{S}_2, \mathcal{W}, \mathcal{Q}, \mathcal{T} \rangle \colon f + g} \ \mathsf{RD}$$

Theorem (Complexity Pair Processor)

The following processor is sound:

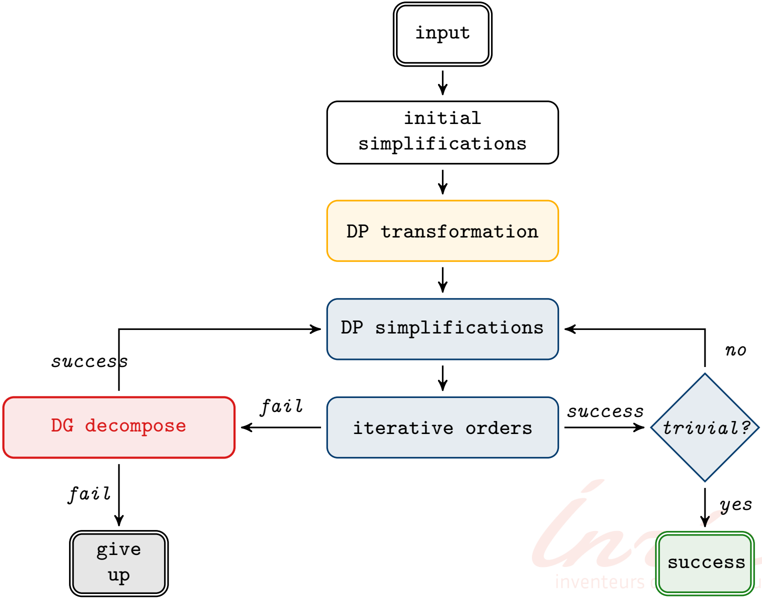
$$\frac{\mathcal{W} \subseteq \gtrsim \qquad \mathcal{S} \subseteq \succ}{\vdash \langle \mathcal{S}, \mathcal{W}, \mathcal{Q}, \mathcal{T} \rangle \colon \mathsf{dc}_{\succ, \mathcal{T}}} \ \mathsf{CP}$$

where $(\gtrsim, \succ)$ *is* (ν, μ)*-monotone complexity pair with*

$$\xrightarrow{\mathcal{Q}}{}^{*}_{\mathcal{S} \cup \mathcal{W}}(\mathcal{T}) \subseteq \mathcal{T}_{\nu}(\xrightarrow{\mathcal{Q}}_{\mathcal{W}}) \qquad\qquad \xrightarrow{\mathcal{Q}}{}^{*}_{\mathcal{S} \cup \mathcal{W}}(\mathcal{T}) \subseteq \mathcal{T}_{\mu}(\xrightarrow{\mathcal{Q}}_{\mathcal{S}}) .$$

★ CP-processor encompasses safe reduction pairs Question: why?

Runtime Complexity Proof Search in TCT



Dependency Graphs

Definition (dependency graph (DG))

dependency graph of (DP) problem $\mathcal{P} = \langle \mathcal{S}, \mathcal{W}, \mathcal{Q}, \mathcal{T} \rangle$ is graph where

- nodes are dependency pairs of $\mathcal{P}$
- there is an edge labeled i from $s \to \mathsf{c}_k(t_1, \ldots, t_k)$ to $u \to \mathsf{c}_l(v_1, \ldots, v_l)$ if $t_i\sigma \xrightarrow{\mathcal{Q}}{}^{*}_{\mathcal{S} \cup \mathcal{W}} u\tau$ holds for some substitutions σ, τ

Dependency Graphs

Definition (dependency graph (DG))

dependency graph of (DP) problem $\mathcal{P} = \langle \mathcal{S}, \mathcal{W}, \mathcal{Q}, \mathcal{T} \rangle$ is graph where

- ⋆ nodes are dependency pairs of $\mathcal{P}$
- ⋆ there is an edge labeled i from $s \to \mathsf{c}_k(t_1, \ldots, t_k)$ to $u \to \mathsf{c}_l(v_1, \ldots, v_l)$ if $t_i\sigma \xrightarrow{\mathcal{Q}}{}^*_{\mathcal{S} \cup \mathcal{W}} u\tau$ holds for some substitutions σ, τ

- ⋆ DG reflects order of dependency pair application
- ⋆ not computable in general $\Rightarrow$ over-approximations exist

R. Thiemann. *"The DP Framework for Proving Termination of Term Rewriting"*. "The DP Framework for Proving Termination of Term Rewriting", 2007.

Example: Dependency Graph

current: $\langle \mathcal{S}, \mathcal{W}, \mathcal{Q}, \mathcal{B}^{\#} \rangle$

$\mathcal{S}$

(1) $[\,] +\!\!+^{\#}\ ys \to \mathtt{c}_0$

(2) $(x :: xs) +\!\!+^{\#}\ ys \to \mathtt{c}_1(xs +\!\!+^{\#}\ ys)$

(3) $\mathtt{rev}^{\#}([\,]) \to \mathtt{c}_0$

(4) $\mathtt{rev}^{\#}(x :: xs) \to \mathtt{c}_2(\mathtt{rev}(xs) +\!\!+^{\#}\ [x], \mathtt{rev}^{\#}(xs))$

$\mathcal{W}$

$[\,] +\!\!+\ ys \to ys$

$(x :: xs) +\!\!+\ ys \to x :: (xs +\!\!+\ ys)$

$\mathtt{rev}([\,]) \to [\,]$

$\mathtt{rev}(x :: xs) \to \mathtt{rev}(xs) +\!\!+\ [x]$

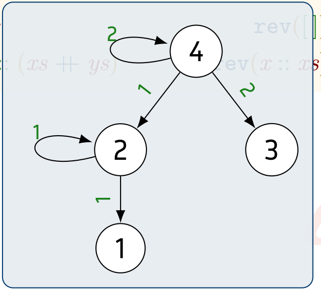
$\mathcal{Q}$

$[\,] +\!\!+\ ys \to ys$

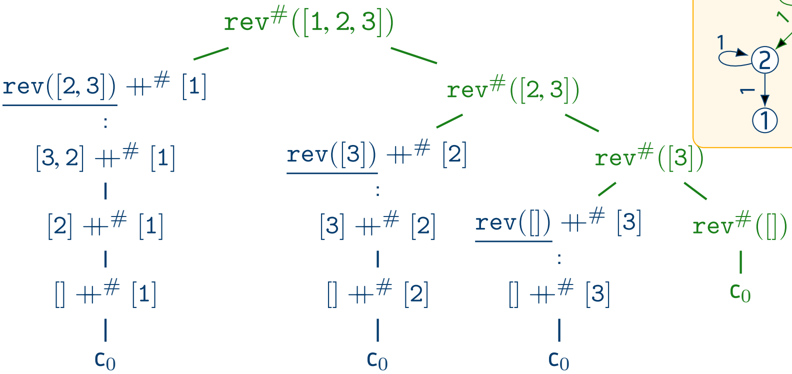
$(x :: xs) +\!\!+\ ys \to x :: (xs +\!\!+\ ys)$

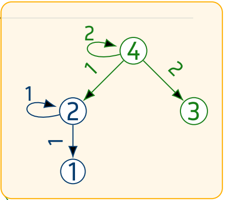
$\mathtt{rev}([\,]) \to [\,]$

$\mathtt{rev}(x :: xs) \to \mathtt{rev}(xs) +\!\!+\ [x]$



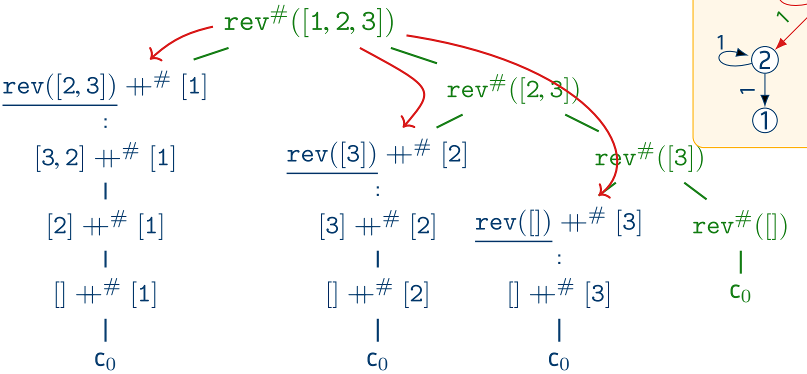
DG Decomposition: Intuitions

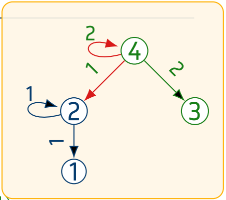




$$\frac{\vdash \langle\{\textcircled{1},\textcircled{2}\},\{\textcircled{3},\textcircled{4}\}\cup \mathcal{W},\mathcal{Q},\mathcal{B}^{\#}\rangle : f \qquad \vdash \langle\{\textcircled{3},\textcircled{4}\},\{\textcircled{1},\textcircled{2}\}\cup\mathcal{W},\mathcal{Q},\mathcal{B}^{\#}\rangle : g}{\vdash \langle \{\textcircled{3},\textcircled{4}\}\cup\{\textcircled{1},\textcircled{2}\},\mathcal{W},\mathcal{Q},\mathcal{B}^{\#}\rangle : f+g}\ \text{RD}$$

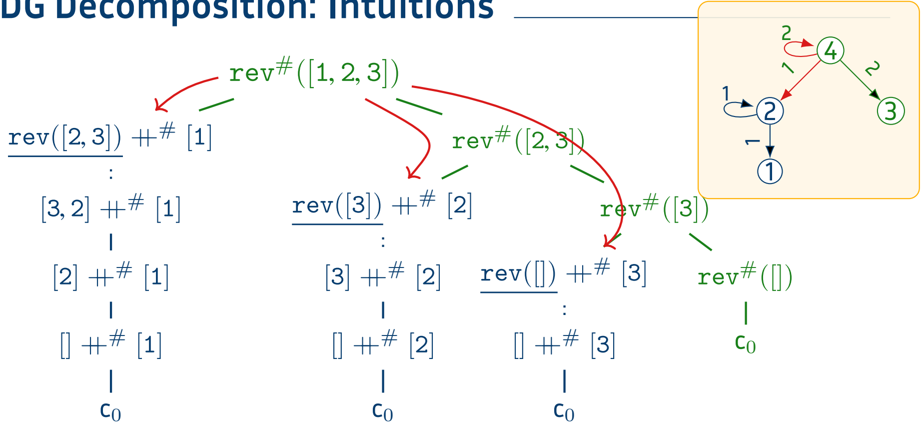
DG Decomposition: Intuitions





$$\frac{\vdash \langle \{\textcircled{1},\textcircled{2}\}, \mathcal{C}\cup\mathcal{W},\mathcal{Q},\mathcal{B}^{\#}\rangle : f \qquad \vdash \langle\{\textcircled{3},\textcircled{4}\},\mathcal{W},\mathcal{Q},\mathcal{B}^{\#}\rangle : g}{\vdash \langle \{\textcircled{3},\textcircled{4}\}\cup\{\textcircled{1},\textcircled{2}\},\mathcal{W},\mathcal{Q},\mathcal{B}^{\#}\rangle : f\times g}\ \mathsf{DGD}$$

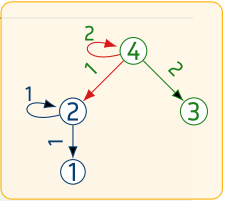
DG Decomposition: Intuitions



$$\frac{\vdash \langle \{①, ②\}, \mathcal{C} \cup \mathcal{W}, \mathcal{Q}, \mathcal{B}^{\#} \rangle : f \qquad \vdash \langle \{③, ④\}, \mathcal{W}, \mathcal{Q}, \mathcal{B}^{\#} \rangle : g}{\vdash \langle \{③, ④\} \cup \{①, ②\}, \mathcal{W}, \mathcal{Q}, \mathcal{B}^{\#} \rangle : f \times g} \text{ DGD}$$

$\mathcal{C}$ $(4 \xrightarrow{1} 2)$ $\texttt{rev}^{\#}(x :: xs) \to \texttt{rev}(xs) +\!\!+^{\#} [x]$ $\quad$ $(4 \xrightarrow{2} 4)$ $\texttt{rev}^{\#}(x :: xs) \to \texttt{rev}^{\#}(xs)$

Example: DG Decomposition



current: $\langle \mathcal{S}_{\Downarrow}, \mathcal{C} \cup \mathcal{W}, \mathcal{Q}, \mathcal{B}^{\#} \rangle$ and $\langle \mathcal{S}_{\Uparrow}, \mathcal{W}, \mathcal{Q}, \mathcal{B}^{\#} \rangle$

$\mathcal{S}_{\Downarrow}$

(1) $[\,] +\!\!+^{\#} ys \to \mathtt{c}_0$

(2) $(x :: xs) +\!\!+^{\#} ys \to \mathtt{c}_1(xs +\!\!+^{\#} ys)$

$\mathcal{S}_{\Uparrow}$

(3) $\mathtt{rev}^{\#}([\,]) \to \mathtt{c}_0$

(4) $\mathtt{rev}^{\#}(x :: xs) \to \mathtt{c}_2(\mathtt{rev}(xs) +\!\!+^{\#} [x], \mathtt{rev}^{\#}(xs))$

$\mathcal{C}$

$\mathtt{rev}^{\#}(x :: xs) \to \mathtt{rev}(xs) +\!\!+^{\#} [x]$

$\mathtt{rev}^{\#}(x :: xs) \to \mathtt{rev}^{\#}(xs)$

$\mathcal{W}$

$[\,] +\!\!+ ys \to ys$ $\qquad$ $\mathtt{rev}([\,]) \to [\,]$

$(x :: xs) +\!\!+ ys \to x :: (xs +\!\!+ ys)$ $\qquad$ $\mathtt{rev}(x :: xs) \to \mathtt{rev}(xs) +\!\!+ [x]$

$\mathcal{Q}$

$[\,] +\!\!+ ys \to ys$ $\qquad$ $\mathtt{rev}([\,]) \to [\,]$

$(x :: xs) +\!\!+ ys \to x :: (xs +\!\!+ ys)$ $\qquad$ $\mathtt{rev}(x :: xs) \to \mathtt{rev}(xs) +\!\!+ [x]$

DG Decomposition

Theorem (DG Decomposition)

The following processor is sound:

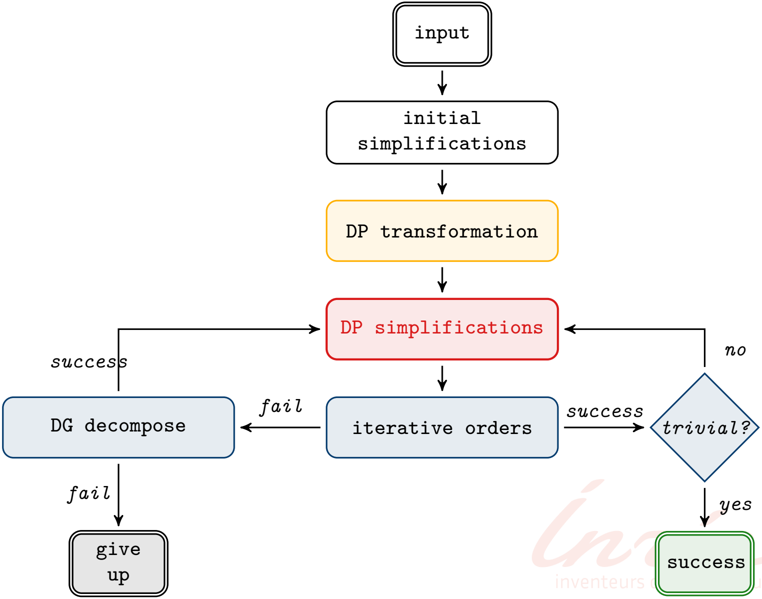
$$\frac{\vdash \langle \mathcal{S}_{\Downarrow}, \mathsf{sep}(\mathcal{S}_{\Uparrow} \cup \mathcal{W}_{\Uparrow}) \cup \mathcal{W}_{\Downarrow} \cup \mathcal{W}, \mathcal{Q}, \mathcal{B}^{\#} \rangle : f \qquad \vdash \langle \mathcal{S}_{\Uparrow}, \mathcal{W}_{\Uparrow} \cup \mathcal{W}, \mathcal{Q}, \mathcal{B}^{\#} \rangle : g}{\vdash \langle \mathcal{S}_{\Downarrow} \cup \mathcal{S}_{\Uparrow}, \mathcal{W}_{\Downarrow} \uplus \mathcal{W}_{\Uparrow} \cup \mathcal{W}, \mathcal{Q}, \mathcal{B}^{\#} \rangle : f \times g}$$

where

- ★ $\mathcal{S}_{\Downarrow}, \mathcal{S}_{\Uparrow}, \mathcal{W}_{\Downarrow}, \mathcal{W}_{\Uparrow}$ *are DPs:*
  1. $\mathcal{S}_{\Downarrow} \cup \mathcal{W}_{\Downarrow}$ *is forward closed set of DPs in the DG*
  2. *DG-predecessors of* $\mathcal{S}_{\Downarrow} \cup \mathcal{W}_{\Downarrow}$ *are in* $\mathcal{S}_{\Uparrow}$
- ★ $\mathsf{sep}(\mathcal{R}) \triangleq \{l \to r_i \mid l \to \mathsf{c}_k(r_1, \ldots, r_k) \in \mathcal{R}\}$

M. Avanzini and G. Moser. *"A Combination Framework for Complexity"*. *Information and Computation, Vol. 248*, pp. 22–55, 2016.

Runtime Complexity Proof Search in TCT



Simplifications: Guided by DG

Theorem (simplify RHSs, remove weak suffix, predecessor estimation)

The following processors are sound:

- *Simplify RHSs:*

$$\frac{\vdash \langle \mathsf{simp}(\mathcal{S}), \mathsf{simp}(\mathcal{W}), \mathcal{Q}, \mathcal{B}^{\#} \rangle \colon f}{\vdash \langle \mathcal{S}, \mathcal{W}, \mathcal{Q}, \mathcal{B}^{\#} \rangle \colon f}\ \text{SIMP−RHS}$$

  where simp *drops* r_i *if DP* $l \to \mathsf{c}_k(r_1, \ldots, r_i, \ldots, r_k)$ *has no outgoing edge labeled by i*

- *Remove weak suffix:*

$$\frac{\mathcal{W}_{\Downarrow}\ \textit{forward-closed DPs} \quad \vdash \langle \mathcal{S}, \mathcal{W}, \mathcal{Q}, \mathcal{B}^{\#} \rangle \colon f}{\vdash \langle \mathcal{S}, \mathcal{W} \uplus \mathcal{W}_{\Downarrow}, \mathcal{Q}, \mathcal{B}^{\#} \rangle \colon f}\ \text{RWS}$$

- *Predecessor estimation:*

$$\frac{\textit{DG-predecessors of } \mathcal{S}_1 \subseteq \mathcal{S}_2 \quad \vdash \langle \mathcal{S}_2, \mathcal{W}, \mathcal{Q}, \mathcal{B}^{\#} \rangle \colon f}{\vdash \langle \mathcal{S}_1 \cup \mathcal{S}_2, \mathcal{W}, \mathcal{Q}, \mathcal{B}^{\#} \rangle \colon f}\ \text{PE}$$

Simplifications: Usable Rules

Theorem (Usable Rules Processor, Semantic Version)

Usable rules $\mathcal{U}_{\mathcal{P}}(\mathcal{R}) \subseteq \mathcal{R}$ *of TRS* $\mathcal{R}$ *wrt.* $\mathcal{P} = \langle \mathcal{S}, \mathcal{W}, \mathcal{Q}, \mathcal{T} \rangle$ *are those that can be applied in* $\mathcal{P}$*-derivation from* $\mathcal{T}$.
The following processor is sound:

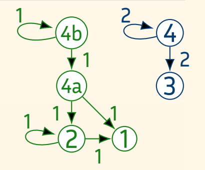
$$\frac{\vdash \langle \mathcal{U}_{\mathcal{P}}(\mathcal{S}), \mathcal{U}_{\mathcal{P}}(\mathcal{W}), \mathcal{Q}, \mathcal{T} \rangle : f}{\vdash \langle \mathcal{S}, \mathcal{W}, \mathcal{Q}, \mathcal{T} \rangle : f} \text{ UR}$$

Notes:

- ⋆ non-usable rules ≈ dead code
- ⋆ usable rules not computable in general
- ⋆ over-approximated, e.g. using tree automata or via usable symbols
  - $\mathtt{f} \triangleright \mathtt{g}$ iff $\mathtt{f}(\vec{l}) \to r \in \mathcal{P}$ and $\mathtt{g} \in \mathcal{D}(r)$
  - usable symbols of terms $\mathcal{T}$ are $\mathcal{US}_{\mathcal{P}}(\mathcal{T}) \triangleq \{\mathtt{g} \mid \exists \mathtt{f} \in \mathcal{D}(\mathcal{T}).\ \mathtt{f} \triangleright^* \mathtt{g}\}$
  - approximated usable rules are $\mathcal{U}_{\mathcal{P}}(\mathcal{R}) \triangleq \{\mathtt{f}(\vec{l}) \to r \in \mathcal{R} \mid \mathtt{f} \in \mathcal{US}_{\mathcal{P}}(\mathcal{T})\}$

Example: Simplifications

current: $\langle \mathcal{S}_{\Downarrow}, \mathcal{C} \cup \mathcal{W}, \mathcal{Q}, \mathcal{B}^{\#} \rangle$ and $\langle \mathcal{S}_{\Uparrow}, \mathcal{W}, \mathcal{Q}, \mathcal{B}^{\#} \rangle$



$\mathcal{S}_{\Downarrow}$

(1) $[\,] +\!\!+^{\#} ys \to \mathsf{c}_0$

(2) $(x :: xs) +\!\!+^{\#} ys \to \mathsf{c}_1(xs +\!\!+^{\#} ys)$

$\mathcal{S}_{\Uparrow}$

(3) $\mathtt{rev}^{\#}([\,]) \to \mathsf{c}_0$

(4) $\mathtt{rev}^{\#}(x :: xs) \to \mathsf{c}_1(\mathtt{rev}(xs) +\!\!+^{\#} [x], \mathtt{rev}^{\#}(xs))$

$\mathcal{C}$

(4a) $\mathtt{rev}^{\#}(x :: xs) \to \mathtt{rev}(xs) +\!\!+^{\#} [x]$

(4b) $\mathtt{rev}^{\#}(x :: xs) \to \mathtt{rev}^{\#}(xs)$

$\mathcal{W}$

$[\,] +\!\!+ ys \to ys$

$(x :: xs) +\!\!+ ys \to x :: (xs +\!\!+ ys)$

$\mathtt{rev}([\,]) \to [\,]$

$\mathtt{rev}(x :: xs) \to \mathtt{rev}(xs) +\!\!+ [x]$

$\mathcal{Q}$

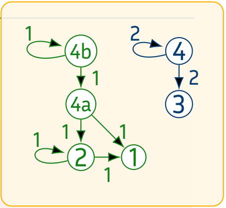
$[\,] +\!\!+ ys \to ys$

$(x :: xs) +\!\!+ ys \to x :: (xs +\!\!+ ys)$

$\mathtt{rev}([\,]) \to [\,]$

$\mathtt{rev}(x :: xs) \to \mathtt{rev}(xs) +\!\!+ [x]$

Example: Simplifications



current: $\langle \mathcal{S}_{\Downarrow}, \mathcal{C} \cup \mathcal{W}, \mathcal{Q}, \mathcal{B}^{\#} \rangle$ and $\langle \mathcal{S}_{\Uparrow}, \mathcal{W}, \mathcal{Q}, \mathcal{B}^{\#} \rangle$

$\mathcal{S}_{\Downarrow}$

(1) $[\,] +\!\!+^{\#} ys \to \mathtt{c}_0$

(2) $(x :: xs) +\!\!+^{\#} ys \to \mathtt{c}_1(xs +\!\!+^{\#} ys)$

$\mathcal{S}_{\Uparrow}$

(3) $\mathtt{rev}^{\#}([\,]) \to \mathtt{c}_0$

(4) $\mathtt{rev}^{\#}(x :: xs) \to \mathtt{c}_1(\mathtt{rev}(xs) +\!\!+^{\#} [x], \mathtt{rev}^{\#}(xs))$

$\mathcal{C}$

(4a) $\mathtt{rev}^{\#}(x :: xs) \to \mathtt{rev}(xs) +\!\!+^{\#} [x]$

(4b) $\mathtt{rev}^{\#}(x :: xs) \to \mathtt{rev}^{\#}(xs)$

$\mathcal{W}$

$[\,] +\!\!+ ys \to ys$ $\qquad$ $\mathtt{rev}([\,]) \to [\,]$

$(x :: xs) +\!\!+ ys \to x :: (xs +\!\!+ ys)$ $\qquad$ $\mathtt{rev}(x :: xs) \to \mathtt{rev}(xs) +\!\!+ [x]$

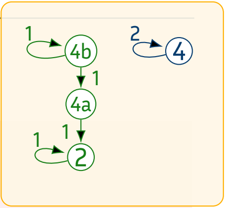
$\mathcal{Q}$

$[\,] +\!\!+ ys \to ys$ $\qquad$ $\mathtt{rev}([\,]) \to [\,]$

$(x :: xs) +\!\!+ y$ … $\mathtt{ev}(xs) +\!\!+ [x]$

predecessor estimation

Example: Simplifications



current: $\langle \mathcal{S}_{\Downarrow}, \mathcal{C} \cup \mathcal{W}, \mathcal{Q}, \mathcal{B}^{\#} \rangle$ and $\langle \mathcal{S}_{\Uparrow}, \mathcal{W}, \mathcal{Q}, \mathcal{B}^{\#} \rangle$

$\mathcal{S}_{\Downarrow}$

(2) $(x :: xs) +\!\!+^{\#} ys \to \mathtt{c}_1(xs +\!\!+^{\#} ys)$

$\mathcal{S}_{\Uparrow}$

(4) $\mathtt{rev}^{\#}(x :: xs) \to \mathtt{c}_2(\mathtt{rev}(xs) +\!\!+^{\#} [x], \mathtt{rev}^{\#}(xs))$

$\mathcal{C}$

(4a) $\mathtt{rev}^{\#}(x :: xs) \to \mathtt{rev}(xs) +\!\!+^{\#} [x]$

(4b) $\mathtt{rev}^{\#}(x :: xs) \to \mathtt{rev}^{\#}(xs)$

$\mathcal{W}$

$[\,] +\!\!+ ys \to ys$ $\quad$ $\mathtt{rev}([\,]) \to [\,]$

$(x :: xs) +\!\!+ ys \to x :: (xs +\!\!+ ys)$ $\quad$ $\mathtt{rev}(x :: xs) \to \mathtt{rev}(xs) +\!\!+ [x]$

$\mathcal{Q}$

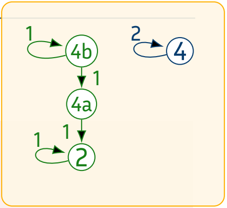
$[\,] +\!\!+ ys \to ys$ $\quad$ $\mathtt{rev}([\,]) \to [\,]$

$(x :: xs) +\!\!+ y$ … $\mathtt{ev}(xs) +\!\!+ [x]$

simplify RHSs

Example: Simplifications

current: $\langle \mathcal{S}_{\Downarrow}, \mathcal{C} \cup \mathcal{W}, \mathcal{Q}, \mathcal{B}^{\#} \rangle$ and $\langle \mathcal{S}_{\Uparrow}, \mathcal{W}, \mathcal{Q}, \mathcal{B}^{\#} \rangle$



$\mathcal{S}_{\Downarrow}$

(2) $(x :: xs) +\!\!+^{\#}\, ys \to \mathtt{c}_1(xs +\!\!+^{\#}\, ys)$

$\mathcal{S}_{\Uparrow}$

(4) $\mathtt{rev}^{\#}(x :: xs) \to \mathtt{c}_1(\mathtt{rev}^{\#}(xs))$

$\mathcal{C}$

(4a) $\mathtt{rev}^{\#}(x :: xs) \to \mathtt{rev}(xs) +\!\!+^{\#} [x]$

(4b) $\mathtt{rev}^{\#}(x :: xs) \to \mathtt{rev}^{\#}(xs)$

$\mathcal{W}$

$[\,] +\!\!+ ys \to ys$ $\qquad$ $\mathtt{rev}([\,]) \to [\,]$

$(x :: xs) +\!\!+ ys \to x :: (xs +\!\!+ ys)$ $\qquad$ $\mathtt{rev}(x :: xs) \to \mathtt{rev}(xs) +\!\!+ [x]$

$\mathcal{Q}$

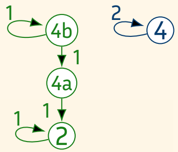
$[\,] +\!\!+ ys \to ys$ $\qquad$ $\mathtt{rev}([\,]) \to [\,]$

$(x :: xs) +\!\!+ y$ … $\mathtt{rev}(xs) +\!\!+ [x]$

usable rules

Example: Simplifications

current: $\langle \mathcal{S}_{\Downarrow}, \mathcal{C} \cup \mathcal{W}, \mathcal{Q}, \mathcal{B}^{\#} \rangle$ and $\langle \mathcal{S}_{\Uparrow}, \varnothing, \mathcal{Q}, \mathcal{B}^{\#} \rangle$



$\mathcal{S}_{\Downarrow}$

(2) $(x :: xs) +\!\!+^{\#} ys \to \mathtt{c}_1(xs +\!\!+^{\#} ys)$

$\mathcal{S}_{\Uparrow}$

(4) $\mathtt{rev}^{\#}(x :: xs) \to \mathtt{c}_1(\mathtt{rev}^{\#}(xs))$

$\mathcal{C}$

(4a) $\mathtt{rev}^{\#}(x :: xs) \to \mathtt{rev}(xs) +\!\!+^{\#} [x]$

(4b) $\mathtt{rev}^{\#}(x :: xs) \to \mathtt{rev}^{\#}(xs)$

$\mathcal{W}$

$[\,] +\!\!+ ys \to ys$ $\qquad$ $\mathtt{rev}([\,]) \to [\,]$

$(x :: xs) +\!\!+ ys \to x :: (xs +\!\!+ ys)$ $\qquad$ $\mathtt{rev}(x :: xs) \to \mathtt{rev}(xs) +\!\!+ [x]$

$\mathcal{Q}$

$[\,] +\!\!+ ys \to ys$ $\qquad$ $\mathtt{rev}([\,]) \to [\,]$

$(x :: xs) +\!\!+ ys \to x :: (xs +\!\!+ ys)$ $\qquad$ $\mathtt{rev}(x :: xs) \to \mathtt{rev}(xs) +\!\!+ [x]$

Example: Finishing the Proof

$$\dfrac{\dfrac{\vdash \langle (2), \mathcal{C} \cup \mathcal{R}_{\mathsf{rev}}, \mathcal{R}_{\mathsf{rev}}, \mathcal{B}^{\#}\rangle : ?}{\vdash \langle \mathcal{S}_{\Downarrow}, \mathcal{C} \cup \mathcal{R}_{\mathsf{rev}}, \mathcal{R}_{\mathsf{rev}}, \mathcal{B}^{\#}\rangle : ?}\ \mathsf{SIMPS} \qquad \dfrac{\vdash \langle (4), \varnothing, \mathcal{R}_{\mathsf{rev}}, \mathcal{B}^{\#}\rangle : ?}{\vdash \langle \mathcal{S}_{\Uparrow}, \mathcal{R}_{\mathsf{rev}}, \mathcal{R}_{\mathsf{rev}}, \mathcal{B}^{\#}\rangle : ?}\ \mathsf{SIMPS}}{\dfrac{\vdash \langle \mathsf{DT}(\mathcal{R}_{\mathsf{rev}}), \mathcal{R}_{\mathsf{rev}}, \mathcal{R}_{\mathsf{rev}}, \mathcal{B}^{\#}\rangle : ?}{\vdash \langle \mathcal{R}_{\mathsf{rev}}, \varnothing, \mathcal{R}_{\mathsf{rev}}, \mathcal{B}\rangle : ?}\ \mathsf{DT}}\ \mathsf{DGD}$$

(2) $(x :: xs) +\!\!+^{\#}\, ys \to \mathtt{c}_1(xs +\!\!+^{\#}\, ys)$

(4) $\mathtt{rev}^{\#}(x :: xs) \to \mathtt{c}_1(\mathtt{rev}^{\#}(xs))$

$\mathcal{C}$

(4a) $\mathtt{rev}^{\#}(x :: xs) \to \mathtt{rev}(xs) +\!\!+^{\#} [x]$

(4b) $\mathtt{rev}^{\#}(x :: xs) \to \mathtt{rev}^{\#}(xs)$

$\mathcal{R}_{\mathsf{rev}}$

$[\,] +\!\!+\, ys \to ys$

$(x :: xs) +\!\!+\, ys \to x :: (xs +\!\!+\, ys)$

$\mathtt{rev}([\,]) \to [\,]$

$\mathtt{rev}(x :: xs) \to \mathtt{rev}(xs) +\!\!+ [x]$

Example: Finishing the Proof

$$\dfrac{\dfrac{\dfrac{(2) \subseteq >_{\mathcal{A}} \quad \mathcal{R}_{\mathsf{rev}} \subseteq \geq_{\mathcal{A}}}{\vdash \langle (2), \mathcal{C} \cup \mathcal{R}_{\mathsf{rev}}, \mathcal{R}_{\mathsf{rev}}, \mathcal{B}^{\#} \rangle : n}\ \text{PI}}{\vdash \langle \mathcal{S}_{\Downarrow}, \mathcal{C} \cup \mathcal{R}_{\mathsf{rev}}, \mathcal{R}_{\mathsf{rev}}, \mathcal{B}^{\#} \rangle : n}\ \text{SIMPS} \qquad \dfrac{\vdash \ \cdots}{\vdash \ \cdots}\ \cdots\text{S}}{\dfrac{\vdash \langle \mathsf{DT}(\mathcal{R}_{\mathsf{rev}}), \mathcal{R}_{\mathsf{rev}}, \mathcal{R}_{\mathsf{rev}}, \mathcal{B} \cdots}{\vdash \langle \mathcal{R}_{\mathsf{rev}}, \varnothing, \mathcal{R}_{\mathsf{rev}}, \mathcal{B} \rangle : \cdots}}$$

$$[\,]_{\mathcal{A}} \triangleq 0$$
$$x ::_{\mathcal{A}} xs \triangleq 1 + xs$$
$$\mathtt{rev}_{\mathcal{A}}(xs) \triangleq xs$$
$$xs \mathbin{+\!\!+}_{\mathcal{A}} ys \triangleq xs + ys$$
$$\mathtt{rev}^{\#}_{\mathcal{A}}(xs) \triangleq xs$$
$$xs \mathbin{+\!\!+}^{\#}_{\mathcal{A}} ys \triangleq xs$$
$$\mathsf{c}_1(t) \triangleq t$$

(2) $(x :: xs) \mathbin{+\!\!+}^{\#} ys \to \mathsf{c}_1(xs \mathbin{+\!\!+}^{\#} ys)$

(4) $\mathtt{rev}^{\#}(x :: xs) \to \mathsf{c}_1(\mathtt{rev}^{\#}(xs))$

$\mathcal{C}$

(4a) $\mathtt{rev}^{\#}(x :: xs) \to \mathtt{rev}(xs) \mathbin{+\!\!+}^{\#} [x]$

(4b) $\mathtt{rev}^{\#}(x :: xs) \to \mathtt{rev}^{\#}(xs)$

$\mathcal{R}_{\mathsf{rev}}$

$[\,] \mathbin{+\!\!+} ys \to ys$	$\mathtt{rev}([\,]) \to [\,]$
$(x :: xs) \mathbin{+\!\!+} ys \to x :: (xs \mathbin{+\!\!+} ys)$	$\mathtt{rev}(x :: xs) \to \mathtt{rev}(xs) \mathbin{+\!\!+} [x]$

Example: Finishing the Proof

- $\star$ $\mathsf{normal}(\mathtt{rev}^{\#}) \triangleq \{1\}$
- $\star$ $\mathtt{rev}^{\#} > \mathtt{c}_1$
- $\star$ recursion depth 1

$$\dfrac{\dfrac{\dfrac{(4) \subseteq >_{\mathsf{spop}*}}{\vdash \langle (4), \varnothing, \mathcal{R}_{\mathsf{rev}}, \mathcal{B}^{\#} \rangle : n}\ \mathsf{SPOP}^*}{\vdash \langle \mathcal{S}_{\Uparrow}, \mathcal{R}_{\mathsf{rev}}, \mathcal{R}_{\mathsf{rev}}, \mathcal{B}^{\#} \rangle : n}\ \mathsf{SIMPS} \qquad \cdots\ \mathsf{SIMPS}}{\dfrac{\vdash \langle \mathsf{DT}(\mathcal{R}_{\mathsf{rev}}), \mathcal{R}_{\mathsf{rev}}, \mathcal{R}_{\mathsf{rev}}, \mathcal{B}^{\#} \rangle : ?}{\vdash \langle \mathcal{R}_{\mathsf{rev}}, \varnothing, \mathcal{R}_{\mathsf{rev}}, \mathcal{B} \rangle : ?}\ \mathsf{DT}}\ \mathsf{DGD}$$

(2) $(x :: xs) +\!\!+^{\#} ys \to \mathtt{c}_1(xs +\!\!+^{\#} ys)$

(4) $\mathtt{rev}^{\#}(x :: xs) \to \mathtt{c}_1(\mathtt{rev}^{\#}(xs))$

$\mathcal{C}$

(4a) $\mathtt{rev}^{\#}(x :: xs) \to \mathtt{rev}(xs) +\!\!+^{\#} [x]$

(4b) $\mathtt{rev}^{\#}(x :: xs) \to \mathtt{rev}^{\#}(xs)$

$\mathcal{R}_{\mathsf{rev}}$

$[\,] +\!\!+ ys \to ys$ $\qquad$ $\mathtt{rev}([\,]) \to [\,]$

$(x :: xs) +\!\!+ ys \to x :: (xs +\!\!+ ys)$ $\qquad$ $\mathtt{rev}(x :: xs) \to \mathtt{rev}(xs) +\!\!+ [x]$

Example: Finishing the Proof

$$\dfrac{\dfrac{\dfrac{(2) \subseteq >_{\mathcal{A}} \quad \mathcal{R}_{\mathsf{rev}} \subseteq \geq_{\mathcal{A}}}{\vdash \langle (2), \mathcal{C} \cup \mathcal{R}_{\mathsf{rev}}, \mathcal{R}_{\mathsf{rev}}, \mathcal{B}^{\#} \rangle : n}\ \mathsf{PI}}{\vdash \langle \mathcal{S}_{\Downarrow}, \mathcal{C} \cup \mathcal{R}_{\mathsf{rev}}, \mathcal{R}_{\mathsf{rev}}, \mathcal{B}^{\#} \rangle : n}\ \mathsf{SIMPS} \qquad \dfrac{\dfrac{(4) \subseteq >_{\mathsf{spop}*}}{\vdash \langle (4), \varnothing, \mathcal{R}_{\mathsf{rev}}, \mathcal{B}^{\#} \rangle : n}\ \mathsf{SPOP}^*}{\vdash \langle \mathcal{S}_{\Uparrow}, \mathcal{R}_{\mathsf{rev}}, \mathcal{R}_{\mathsf{rev}}, \mathcal{B}^{\#} \rangle : n}\ \mathsf{SIMPS}}{\dfrac{\vdash \langle \mathsf{DT}(\mathcal{R}_{\mathsf{rev}}), \mathcal{R}_{\mathsf{rev}}, \mathcal{R}_{\mathsf{rev}}, \mathcal{B}^{\#} \rangle : n^2}{\vdash \langle \mathcal{R}_{\mathsf{rev}}, \varnothing, \mathcal{R}_{\mathsf{rev}}, \mathcal{B} \rangle : n^2}\ \mathsf{DT}}\ \mathsf{DGD}$$

(2) $(x :: xs) +\!\!+^{\#} ys \to \mathtt{c}_1(xs +\!\!+^{\#} ys)$

(4) $\mathtt{rev}^{\#}(x :: xs) \to \mathtt{c}_1(\mathtt{rev}^{\#}(xs))$

$\mathcal{C}$

(4a) $\mathtt{rev}^{\#}(x :: xs) \to \mathtt{rev}(xs) +\!\!+^{\#} [x]$

(4b) $\mathtt{rev}^{\#}(x :: xs) \to \mathtt{rev}^{\#}(xs)$

$\mathcal{R}_{\mathsf{rev}}$

$$[\,] +\!\!+ ys \to ys \qquad \mathtt{rev}([\,]) \to [\,]$$

$$(x :: xs) +\!\!+ ys \to x :: (xs +\!\!+ ys) \qquad \mathtt{rev}(x :: xs) \to \mathtt{rev}(xs) +\!\!+ [x]$$

Implementation notes

- implementing complexity pairs

Complexity Pairs in TCT

- ⋆ polynomial, matrix, arctic interpretations and (small) polynomial path orders (modulo argument filtering) implemented in TCT

Complexity Pairs in TCT

- ⋆ polynomial, matrix, arctic interpretations and (small) polynomial path orders (modulo argument filtering) implemented in TCT
- ⋆ RD-processor, CP-processor and UR-processor combined in one

$$\frac{\mathcal{U}_{\mathcal{P},\succ}(\mathcal{S}_1) \subseteq \succ \quad \mathcal{U}_{\mathcal{P},\succ}(\mathcal{S}_2 \cup \mathcal{W}) \subseteq \gtrsim \quad \vdash \langle \mathcal{S}_2, \mathcal{S}_1 \cup \mathcal{W}, \mathcal{Q}, \mathcal{T} \rangle \colon g}{\vdash \langle \mathcal{S}_1 \cup \mathcal{S}_2, \mathcal{W}, \mathcal{Q}, \mathcal{T} \rangle \colon \mathsf{dc}_{\succ,\mathcal{T}} + g}$$

  - usable rules $\mathcal{U}_{\mathcal{P},\succ}$ take problem $\mathcal{P}$ and order $\succ$ into account
  - "function usable" only if occurs in right-hand-side "inspected by" $(\succ, \gtrsim)$
  - specific definition depends on kind of order

Complexity Pairs in TCT

- polynomial, matrix, arctic interpretations and (small) polynomial path orders (modulo argument filtering) implemented in TCT
- RD-processor, CP-processor and UR-processor combined in one

$$\frac{\mathcal{U}_{\mathcal{P},\succ}(\mathcal{S}_1) \subseteq \succ \quad \mathcal{U}_{\mathcal{P},\succ}(\mathcal{S}_2 \cup \mathcal{W}) \subseteq \gtrsim \quad \vdash \langle \mathcal{S}_2, \mathcal{S}_1 \cup \mathcal{W}, \mathcal{Q}, \mathcal{T} \rangle \colon g}{\vdash \langle \mathcal{S}_1 \cup \mathcal{S}_2, \mathcal{W}, \mathcal{Q}, \mathcal{T} \rangle \colon \mathsf{dc}_{\succ,\mathcal{T}} + g}$$

  - usable rules $\mathcal{U}_{\mathcal{P},\succ}$ take problem $\mathcal{P}$ and order $\succ$ into account
  - "function usable" only if occurs in right-hand-side "inspected by" $(\succ, \gtrsim)$
  - specific definition depends on kind of order
- search via encoding to SAT modulo theories (SMT)

Example: Synthesis PI

⋆ fix abstract shape of interpretations...

$$\mathtt{f}_{\mathcal{A}}(x) = \mathtt{c}_{\mathtt{f}}^{x} \cdot x + \mathtt{c}_{\mathtt{f}} \qquad \mathtt{g}_{\mathcal{A}}(x,y) = \mathtt{c}_{\mathtt{g}}^{xy} \cdot x \cdot y + \mathtt{c}_{\mathtt{g}}^{x} \cdot x + \mathtt{c}_{\mathtt{g}}^{y} \cdot y + \mathtt{c}_{\mathtt{g}}$$

...and lift algebraic operations and interpretation of terms:

$$[\![\mathtt{f}(\mathtt{g}(x,y))]\!] = \mathtt{c}_{\mathtt{f}}^{x} \cdot \mathtt{c}_{\mathtt{g}}^{xy} \cdot x \cdot y + \mathtt{c}_{\mathtt{f}}^{x} \cdot \mathtt{c}_{\mathtt{g}}^{x} \cdot x + \mathtt{c}_{\mathtt{f}}^{x} \cdot \mathtt{c}_{\mathtt{g}}^{y} \cdot y + \mathtt{c}_{\mathtt{f}}^{x} \cdot \mathtt{c}_{\mathtt{g}} + \mathtt{c}_{\mathtt{f}}$$

$$[\![\mathtt{f}(\mathtt{g}(x,y))]\!] - [\![\mathtt{f}(x)]\!] = \mathtt{c}_{\mathtt{f}}^{x} \cdot \mathtt{c}_{\mathtt{g}}^{xy} \cdot x \cdot y + \mathtt{c}_{\mathtt{f}}^{x} \cdot (\mathtt{c}_{\mathtt{g}}^{x} - 1) \cdot x + \mathtt{c}_{\mathtt{f}}^{x} \cdot \mathtt{c}_{\mathtt{g}}^{y} \cdot y + \mathtt{c}_{\mathtt{f}}^{x} \cdot \mathtt{c}_{\mathtt{g}}$$

Example: Synthesis PI

- fix abstract shape of interpretations...

$$\mathtt{f}_{\mathcal{A}}(x) = \mathsf{c}_{\mathtt{f}}^{x} \cdot x + \mathsf{c}_{\mathtt{f}} \qquad \mathtt{g}_{\mathcal{A}}(x, y) = \mathsf{c}_{\mathtt{g}}^{xy} \cdot x \cdot y + \mathsf{c}_{\mathtt{g}}^{x} \cdot x + \mathsf{c}_{\mathtt{g}}^{y} \cdot y + \mathsf{c}_{\mathtt{g}}$$

...and lift algebraic operations and interpretation of terms:

$$[\![\mathtt{f}(\mathtt{g}(x, y))]\!] = \mathsf{c}_{\mathtt{f}}^{x} \cdot \mathsf{c}_{\mathtt{g}}^{xy} \cdot x \cdot y + \mathsf{c}_{\mathtt{f}}^{x} \cdot \mathsf{c}_{\mathtt{g}}^{x} \cdot x + \mathsf{c}_{\mathtt{f}}^{x} \cdot \mathsf{c}_{\mathtt{g}}^{y} \cdot y + \mathsf{c}_{\mathtt{f}}^{x} \cdot \mathsf{c}_{\mathtt{g}} + \mathsf{c}_{\mathtt{f}}$$

$$[\![\mathtt{f}(\mathtt{g}(x, y))]\!] - [\![\mathtt{f}(x)]\!] = \mathsf{c}_{\mathtt{f}}^{x} \cdot \mathsf{c}_{\mathtt{g}}^{xy} \cdot x \cdot y + \mathsf{c}_{\mathtt{f}}^{x} \cdot (\mathsf{c}_{\mathtt{g}}^{x} - 1) \cdot x + \mathsf{c}_{\mathtt{f}}^{x} \cdot \mathsf{c}_{\mathtt{g}}^{y} \cdot y + \mathsf{c}_{\mathtt{f}}^{x} \cdot \mathsf{c}_{\mathtt{g}}$$

- (weak) orientation of rule $\mathtt{f}(l_1, \ldots, l_k) \to r$ expressible as

$$\mathtt{f}(l_1, \ldots, l_k) \bowtie_{\mathcal{A}} r \triangleq [\![\mathtt{f}(l_1, \ldots, l_k)]\!]_{\mathcal{A}} - [\![r]\!]_{\mathcal{A}} \bowtie 0$$

where $(\bowtie \in \{>_{\mathbb{N}}, \geq_{\mathbb{N}}\})$
  - approximated via absolute positiveness condition on coefficients

$$[\![\mathtt{f}(\mathtt{g}(x, y))]\!] >_{\mathcal{A}} [\![\mathtt{f}(x)]\!] = \mathsf{c}_{\mathtt{f}}^{x} \cdot \mathsf{c}_{\mathtt{g}}^{xy} \geq_{\mathbb{N}} 0 \wedge \mathsf{c}_{\mathtt{f}}^{x} \cdot (\mathsf{c}_{\mathtt{g}}^{x} - 1) \geq_{\mathbb{N}} 0 \wedge \mathsf{c}_{\mathtt{f}}^{x} \cdot \mathsf{c}_{\mathtt{g}}^{y} \geq_{\mathbb{N}} 0$$
$$\wedge\ \mathsf{c}_{\mathtt{f}}^{x} \cdot \mathsf{c}_{\mathtt{g}} \geq_{\mathbb{N}} 1$$

Example: Synthesis PI (II)

⋆ μ-monotonicity of $\mathtt{f}_{\mathcal{A}}$ encoded via

$$\mathsf{mono}(\mathtt{f}_{\mathcal{A}}, \mu) \triangleq \bigwedge_{\mathsf{c}_{\mathtt{f}}^{\overline{x}} \in \mathsf{coeff}(\mathtt{f})} \mathsf{c}_{\mathtt{f}}^{\overline{x}} \geq_{\mathbb{N}} 0 \wedge \bigwedge_{i \in \mu(\mathtt{f})} \mathsf{c}_{\mathtt{f}}^{x_i} \geq_{\mathbb{N}} 1$$

where $\mathtt{f}_{\mathcal{A}}(x_1, \ldots, x_k) = \sum_{\overline{x} \subseteq \{x_1, \ldots, x_k\}} \mathsf{c}_{\mathtt{f}}^{\overline{x}} \cdot \overline{x}$

Example: Synthesis PI (II)

★ μ-monotonicity of $\mathtt{f}_{\mathcal{A}}$ encoded via

$$\mathsf{mono}(\mathtt{f}_{\mathcal{A}}, \mu) \triangleq \bigwedge_{\mathsf{c}_{\mathtt{f}}^{\overline{x}} \in \mathsf{coeff}(\mathtt{f})} \mathsf{c}_{\mathtt{f}}^{\overline{x}} \geq_{\mathbb{N}} 0 \wedge \bigwedge_{i \in \mu(\mathtt{f})} \mathsf{c}_{\mathtt{f}}^{x_i} \geq_{\mathbb{N}} 1$$

where $\mathtt{f}_{\mathcal{A}}(x_1, \ldots, x_k) = \sum_{\overline{x} \subseteq \{x_1, \ldots, x_k\}} \mathsf{c}_{\mathtt{f}}^{\overline{x}} \cdot \overline{x}$

★ usable rules of $\mathcal{R}$ wrt. start terms $\mathcal{T}$ encoded with atoms $\mathsf{u}_{l \to r}$ via

$$\mathsf{URs}(\mathcal{R}, \mathcal{T}) \triangleq \bigwedge_{\substack{l \to r \in \mathcal{R} \\ \mathsf{rt}(l) \in \mathsf{Fun}(\mathcal{T})}} \mathsf{u}_{l \to r} \wedge \bigwedge_{l \to r \in \mathcal{R}} (\mathsf{u}_{l \to r} \to \phi(r))$$

where

$$\phi(x) \triangleq \top$$

$$\phi(\mathtt{f}(t_1, \ldots, t_k)) \triangleq \bigwedge_{l \to r \in \mathcal{R}, \mathsf{rt}(l) = \mathtt{f}} \mathsf{u}_{l \to r} \wedge \bigwedge_{1 \leq i \leq k} (\pi(\mathtt{f}, i) \to \phi(t_i)) \qquad \pi(\mathtt{f}, i) \triangleq \bigvee_{\mathsf{c}_{\mathtt{f}}^{\overline{x}} \in \mathsf{coeff}(\mathtt{f}), x_i \in \overline{x}} \mathsf{c}_{\mathtt{f}}^{\overline{x}} \geq_{\mathbb{N}} 1$$

Example: Synthesis PI (III)

★ weak orientation of TRS $\mathcal{R}$ via

$$\mathsf{orient}(\mathcal{R}) \triangleq \bigwedge_{l \to r \in \mathcal{R}} \mathsf{u}_{l \to r} \to [\![l]\!]_{\mathcal{A}} - [\![r]\!]_{\mathcal{A}} \geq_{\mathbb{N}} m_{l \to r}$$

with fresh integer variables $m_{l \to r} \geq 0$ for each $l \to r \in \mathcal{R}$

Example: Synthesis PI (III)

- weak orientation of TRS $\mathcal{R}$ via

$$\mathsf{orient}(\mathcal{R}) \triangleq \bigwedge_{l \to r \in \mathcal{R}} \mathsf{u}_{l \to r} \to [\![l]\!]_{\mathcal{A}} - [\![r]\!]_{\mathcal{A}} \geq_{\mathbb{N}} m_{l \to r}$$

  with fresh integer variables $m_{l \to r} \geq 0$ for each $l \to r \in \mathcal{R}$

- extended RP processor for $\langle \mathcal{S}, \mathcal{W}, \mathcal{Q}, \mathcal{T} \rangle$ implementable as

$$\bigwedge_{\mathtt{f} \in \mathcal{F}} \mathsf{mono}(\mathtt{f}_{\mathcal{A}}, \mu \cup \nu) \wedge \mathsf{URs}(\mathcal{S} \cup \mathcal{W}, \mathcal{T}) \wedge \mathsf{orient}(\mathcal{S} \cup \mathcal{W}) \wedge \Phi$$

  - formula Φ enforces which rules in $\mathcal{R} \subseteq \mathcal{S}$ should be oriented strictly, e.g.,

$$\Phi \triangleq \bigwedge_{l \to r \in \mathcal{S}} m_{l \to r} \geq_{\mathbb{N}} 1 \quad \text{or} \quad \Phi \triangleq \bigvee_{l \to r \in \mathcal{S}} m_{l \to r} \geq_{\mathbb{N}} 1$$

  - open sub-problem: $\langle \mathcal{S} \setminus \mathcal{R}, \mathcal{W} \cup \mathcal{R}, \mathcal{Q}, \mathcal{T} \rangle$ where $\mathcal{R}$ determined from assignment of variables $m_{l \to r}$

Summary

- ⋆ TCT build on top of a modular framework for complexity analysis

Summary

- ⋆ TCT build on top of a modular framework for complexity analysis
- ⋆ decomposition techniques such as DG decomposition key to strength of analysis
- ⋆ ultimately, analysis boils down to synthesising a "ranking function" (reduction orders) via SMT

Summary

- ⋆ TCT build on top of a modular framework for complexity analysis
- ⋆ decomposition techniques such as DG decomposition key to strength of analysis
- ⋆ ultimately, analysis boils down to synthesising a “ranking function” (reduction orders) via SMT
- ⋆ currently, tools give asymptotic bounds, but more precise bounds could be extracted

Summary

- TCT build on top of a modular framework for complexity analysis
- decomposition techniques such as DG decomposition key to strength of analysis
- ultimately, analysis boils down to synthesising a “ranking function” (reduction orders) via SMT
- currently, tools give asymptotic bounds, but more precise bounds could be extracted
- automated tools can treat non-trivial examples, fully automatically
- proofs requiring semantic arguments are beyond reach for fully automated analysis

Applications to Program Analysis

⋆ Case study: higher-order functional programs

Motivation

```
let (∘) f g = fun z → f (g z) ;;
let rec walk = function
  | [] → id
  | x::xs → walk xs ∘ (fun ys → x::ys) ;;
let rev l = walk l [] ;;
```

Goal: Runtime Complexity Analysis of Higher-Order Programs

Main Challenge: applied functions not statically known

Direct Approaches: Rewriting Techniques

★ Higher-Order Polynomial Interpretations

$$[\![\texttt{map}]\!] = \lambda\phi.\lambda n.n \times (\phi\, n) : (\mathbb{N} \to \mathbb{N}) \to \mathbb{N} \to \mathbb{N}$$

P. Baillot and U. Dal Lago. *"Higher-Order Interpretations and Program Complexity"*. In *Proc. of 26th CSL*, pp. 62–76, 2012.

Direct Approaches: Type Systems

- ⋆ Amortized Resource Analysis

$$\Gamma \vdash^{k} \texttt{map} : (\mathbb{N}^{p} \xrightarrow{1} \mathbb{N}^{q}) \xrightarrow{0} \mathbb{L}^{s} \xrightarrow{c} \mathbb{L}^{t}$$

S. Jost et al. *"Static Determination of Quantitative Resource Usage for Higher-order Programs"*. In *Proc. of 37th POPL*, pp. 223–236, 2010.

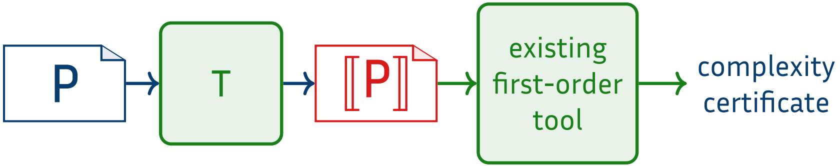
J. Hoffmann, A. Das, and S-C. Weng. *"Towards Automatic Resource Bound Analysis for OCaml"*. In *Proc. of 44th POPL*, pp. 359–373, 2017.

- ⋆ Sized types and instrumentation with clock

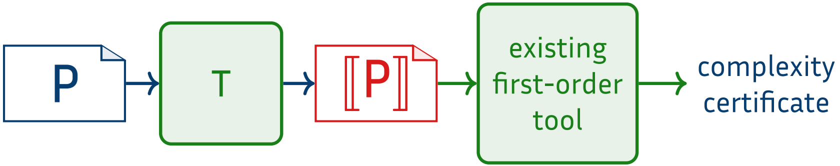
$$\Gamma \vdash \texttt{map} : \forall lk.\ (\forall i.\mathbb{N}_{i} \xrightarrow{f(i)} \mathbb{N}_{g(i)}) \xrightarrow{0} \mathbb{L}_{l}(\mathbb{N}_{k}) \xrightarrow{(f(k)+1)\cdot l} \mathbb{L}_{g}(\mathbb{N}_{f(k)})$$

M. Avanzini and U. Dal Lago. *"Automating Sized-Type Inference for Complexity Analysis"*. In *Proc. of 22nd ICFP*, 2017.

Program Transformations for Complexity Analysis



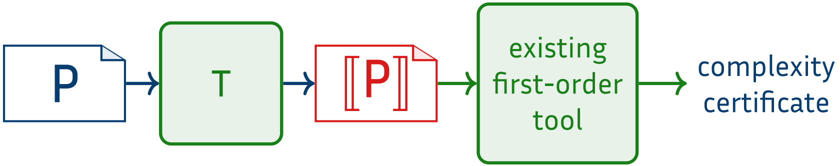
Program Transformations for Complexity Analysis



Constraints on Transformation T:

1. certificate can be relayed back to input program P
   - complexity reflecting: runtime of P $\leq$ runtime of $T(\mathsf{P})$
   - ideally, complexity preserving: runtime of $T(\mathsf{P})$ $\leq$ runtime of P

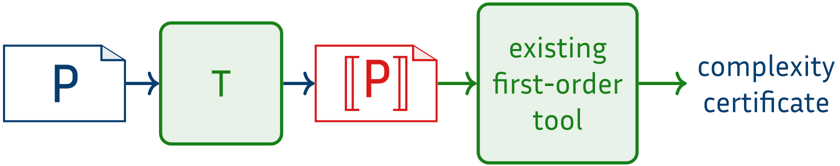
Program Transformations for Complexity Analysis



Constraints on Transformation T:

1. certificate can be relayed back to input program P
   - complexity reflecting: runtime of P $\leq$ runtime of $T(\mathsf{P})$
   - ideally, complexity preserving: runtime of $T(\mathsf{P})$ $\leq$ runtime of P
2. transformed program $[\![\mathsf{P}]\!]$ suitable for analysis by existing tools

Program Transformations for Complexity Analysis



Constraints on Transformation *T*:

1. certificate can be relayed back to input program P
   - complexity reflecting: runtime of P $\leq$ runtime of $T(\mathsf{P})$
   - ideally, complexity preserving: runtime of $T(\mathsf{P})$ $\leq$ runtime of P
2. transformed program ⟦P⟧ suitable for analysis by existing tools

Natural Candidate: Reynold's defunctionalization

From Programs to Rewrite Systems

Input:

- "*PCF* + constructors"

$$M, N ::= x \mid M\ N \mid \lambda x.M \mid \mathsf{fix}(x.M) \mid \mathtt{C}(M_1, \dots, M_k)$$
$$\mid \mathsf{match}\ M\ \mathsf{with}\ \{\mathtt{C}_1(\vec{x_1}) \mapsto M_1 \mid \cdots \mid \mathtt{C}_\mathtt{n}(\vec{x_n}) \mapsto M_n\}$$

- usual call-by-value reduction semantics

Output: applicative term rewrite system (ATRS)

From Programs to Rewrite Systems

Definition (defunctionalization to ATRS)

- ★ $\langle x \rangle \triangleq x$
- ★ $\langle M\ N \rangle \triangleq \langle M \rangle \mathbin{\texttt{@}} \langle N \rangle$
- ★ $\langle \mathtt{C}(M_1, \ldots, M_k) \rangle \triangleq \mathtt{C}(\langle M_1 \rangle, \ldots, \langle M_k \rangle)$
- ★ $\langle \lambda x.M \rangle \triangleq \mathtt{Lam}_{x.M}(\vec{y})$ where $\vec{y} = \mathsf{FVar}(\lambda x.M)$

  $\mathtt{Lam}_{x.M}(\vec{y}) \mathbin{\texttt{@}} x \to \langle M \rangle$

U. Dal Lago and S. Martini. *"On Constructor Rewrite Systems and the Lambda-Calculus"*. In *Proc. of 36th ICALP*, pp. 163–174, 2009.

From Programs to Rewrite Systems

Definition (defunctionalization to ATRS)

- ★ $\langle x \rangle \triangleq x$
- ★ $\langle M\ N \rangle \triangleq \langle M \rangle \mathbin{@} \langle N \rangle$
- ★ $\langle \mathtt{C}(M_1, \ldots, M_k) \rangle \triangleq \mathtt{C}(\langle M_1 \rangle, \ldots, \langle M_k \rangle)$
- ★ $\langle \lambda x.M \rangle \triangleq \mathtt{Lam}_{x.M}(\vec{y})$ where $\vec{y} = \mathsf{FVar}(\lambda x.M)$

  $\mathtt{Lam}_{x.M}(\vec{y}) \mathbin{@} x \to \langle M \rangle$
- ★ $\langle \mathsf{fix}(x.M) \rangle \triangleq \mathtt{Fix}_{x.M}(\vec{y})$ where $\vec{y} = \mathsf{FVar}(\mathsf{fix}(x.M))$

  $\mathtt{Fix}_{x.M}(\vec{y}) \mathbin{@} z \to \langle M \rangle\{\mathtt{Fix}_{x.M}(\vec{y})/x\} \mathbin{@} z$

U. Dal Lago and S. Martini. *"On Constructor Rewrite Systems and the Lambda-Calculus"*. In *Proc. of 36th ICALP*, pp. 163–174, 2009.

From Programs to Rewrite Systems

Definition (defunctionalization to ATRS)

- ⋆ $\langle x \rangle \triangleq x$
- ⋆ $\langle M\ N \rangle \triangleq \langle M \rangle \mathbin{@} \langle N \rangle$
- ⋆ $\langle \mathtt{C}(M_1, \dots, M_k) \rangle \triangleq \mathtt{C}(\langle M_1 \rangle, \dots, \langle M_k \rangle)$
- ⋆ $\langle \lambda x.M \rangle \triangleq \mathtt{Lam}_{x.M}(\vec{y})$ where $\vec{y} = \mathsf{FVar}(\lambda x.M)$

  $\mathtt{Lam}_{x.M}(\vec{y}) \mathbin{@} x \to \langle M \rangle$
- ⋆ $\langle \mathsf{fix}(x.M) \rangle \triangleq \mathtt{Fix}_{x.M}(\vec{y})$ where $\vec{y} = \mathsf{FVar}(\mathsf{fix}(x.M))$

  $\mathtt{Fix}_{x.M}(\vec{y}) \mathbin{@} z \to \langle M \rangle \{\mathtt{Fix}_{x.M}(\vec{y})/x\} \mathbin{@} z$
- ⋆ $\langle \mathsf{match}\ M\ \mathsf{with}\ cs \rangle = \mathtt{Match}_{cs}(\vec{y}) \mathbin{@} \langle M \rangle$ where $\vec{y} = \mathsf{FVar}(cs)$

  $\mathtt{Match}_{cs}(\vec{y}) \mathbin{@} \mathtt{C_i}(\vec{x_i}) \to \langle M_i \rangle \quad (1 \leq i \leq n, cs = \{\cdots \mid \mathtt{C_i}(\vec{x_i}) \mapsto M_i \mid \dots\})$

U. Dal Lago and S. Martini. *"On Constructor Rewrite Systems and the Lambda-Calculus"*. In *Proc. of 36th ICALP*, pp. 163–174, 2009.

From Programs to Rewrite Systems (II)

Theorem

Let $\mathcal{A}_{PCF}$ collect all rules defined synchronous to $\langle\cdot\rangle$.

1. *$\mathcal{A}_{PCF}$ implements PCF in a step-by-step manner (call-by-value)*
2. *on first-order inputs, finite restriction $\mathcal{A}_{P} \subsetneq \mathcal{A}_{PCF}$ sufficient to implement $P = \lambda\vec{x}.M$.*

ATRS $\mathcal{A}_{\mathsf{rev}}$

```
let (∘) f g = fun z → f (g z) ;;
let rec walk = function
  | [] → id
  | x::xs → walk xs ∘ (fun ys → x::ys) ;;
let rev l = walk l [] ;;
```

⇓ desugar + defunctionalize

(1)	$\mathtt{Rev} \mathbin{@} l \to \mathtt{Fix}_{\mathtt{w}} \mathbin{@} l \mathbin{@} []$	(6)	$(\circ) \mathbin{@} f \to (\circ)_1(f)$
(2)	$\mathtt{Fix}_{\mathtt{w}} \mathbin{@} l \to \mathtt{Lam}_1 \mathbin{@} l$	(7)	$(\circ)_1(f) \mathbin{@} g \to \mathtt{Lam}_3(f, g)$
(3)	$\mathtt{Lam}_1 \mathbin{@} l \to \mathtt{Match}_{\mathtt{w}} \mathbin{@} l$	(8)	$\mathtt{Lam}_3(f, g) \mathbin{@} z \to f \mathbin{@} (g \mathbin{@} z)$
(4)	$\mathtt{Match}_{\mathtt{w}} \mathbin{@} [] \to \mathtt{Id}$	(9)	$\mathtt{Id} \mathbin{@} ys \to ys$
(5)	$\mathtt{Match}_{\mathtt{w}} \mathbin{@} (x::xs) \to (\circ) \mathbin{@} (\mathtt{Fix}_{\mathtt{w}} \mathbin{@} xs) \mathbin{@} \mathtt{Lam}_2(x)$	(10)	$\mathtt{Lam}_2(x) \mathbin{@} ys \to x::ys$

ATRS $\mathcal{A}_{\mathsf{rev}}$

```
let (∘) f g = fun z → f (g z) ;;
let rec walk = function
   | [] → id
   | x::xs → walk xs ∘ (fun ys → x::ys) ;;
let rev l = walk l [] ;;
```

⇓ desugar + defunctionalize

(1)	$\mathtt{Rev} \mathbin{@} l \to \mathtt{Fix_w} \mathbin{@} l \mathbin{@} \mathtt{[]}$	(6)	$(\circ) \mathbin{@} f \to (\circ)_1(f)$
(2)	$\mathtt{Fix_w} \mathbin{@} l \to \mathtt{Lam_1} \mathbin{@} l$	(7)	$(\circ)_1(f) \mathbin{@} g \to \mathtt{Lam_3}(f, g)$
(3)	$\mathtt{Lam_1} \mathbin{@} l \to \mathtt{Match_w} \mathbin{@} l$	(8)	$\mathtt{Lam_3}(f, g) \mathbin{@} z \to f \mathbin{@} (g \mathbin{@} z)$
(4)	$\mathtt{Match_w} \mathbin{@} \mathtt{[]} \to \mathtt{Id}$	(9)	$\mathtt{Id} \mathbin{@} ys \to ys$
(5)	$\mathtt{Match_w} \mathbin{@} (x :: xs) \to (\circ) \mathbin{@} (\mathtt{Fix_w} \mathbin{@} xs) \mathbin{@} \mathtt{Lam_2}(x)$	(10)	$\mathtt{Lam_2}(x) \mathbin{@} ys \to x :: ys$

in suitable format for analysis by first-order tools

Experimental Evaluation

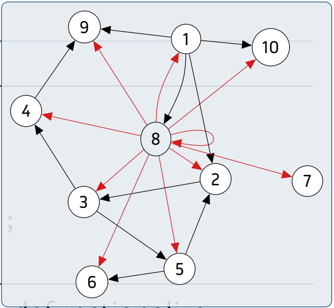
- ⋆ Implementation: `http://cbr.uibk.ac.at/tools/hoca/`
- ⋆ FOP: TCTv2 for complexity, TTT2 for termination (SN)
- ⋆ Testbed: 25 higher-order functions from literature on FP
  - higher-order sorting functions, list & tree traversals (maps, folds, ...), Okasaki's parser combinators, ...

		constant	linear	quadratic	poly	SN
RaML	# systems	2	4	8	—	—
	avg. ET (secs)	2.79	0.32	1.55	—	—
Defunctionalize	# systems	2	5	5	5	8
	FOP avg. ET (secs)	1.71	4.82	4.82	4.82	1.38

Table: Experimental evaluation on 25 higher-order examples. Defunctionalize: Amortized, type-based analysis with RaML prototoype (`http://raml.co/`). Simplify: FOP on defunctionalized ATRS.

ATRS $\mathcal{A}_{\text{rev}}$

```
let (∘) f g = fun z → f (g z) ;;
let rec walk = function
  | [] → id
  | x::xs → walk xs ∘ (fun ys → x::ys) ;;
let rev l = walk l [] ;;
```



⇓ desugar + defunctionalize

(1)	$\texttt{Rev} \;@\; l \to \texttt{Fix}_{\texttt{w}} \;@\; l \;@\; []$	(6)	$(\circ) \;@\; f \to (\circ)_1(f)$
(2)	$\texttt{Fix}_{\texttt{w}} \;@\; l \to \texttt{Lam}_1 \;@\; l$	(7)	$(\circ)_1(f) \;@\; g \to \texttt{Lam}_3(f, g)$
(3)	$\texttt{Lam}_1 \;@\; l \to \texttt{Match}_{\texttt{w}} \;@\; l$	(8)	$\texttt{Lam}_3(f, g) \;@\; z \to f \;@\; (g \;@\; z)$
(4)	$\texttt{Match}_{\texttt{w}} \;@\; [] \to \texttt{Id}$	(9)	$\texttt{Id} \;@\; ys \to ys$
(5)	$\texttt{Match}_{\texttt{w}} \;@\; (x::xs) \to (\circ) \;@\; (\texttt{Fix}_{\texttt{w}} \;@\; xs) \;@\; \texttt{Lam}_2(x)$	(10)	$\texttt{Lam}_2(x) \;@\; ys \to x::ys$

★ recursive structure of translated ATRSs apparently too complicated

1. defines one global function @
2. computation entirely driving by data

Governing the Chaos

program transformations can remedy the situations

1. inlining
   - remove unnecessary indirections introduced by rigid transformation
2. dead code elimination
   - eliminate inlined functions
3. **instantiation**
   - specialize “higher-order variables” via control/data flow analysis
4. uncurrying
   - effectively replaces global apply function with specialized ones

Inlining & Dead Code Elimination

- ★ inlining is optimization that replaces function calls by bodies
- ★ dead code elimination removes non-reachable code

(2)	$\mathtt{Fix}_{\mathtt{w}} \mathbin{@} l \to \boxed{\mathtt{Lam}_1 \mathbin{@} l}$
(3)	$\mathtt{Lam}_1 \mathbin{@} l \to \mathtt{Match}_{\mathtt{w}} \mathbin{@} l$
(4)	$\mathtt{Match}_{\mathtt{w}} \mathbin{@} [] \to \mathtt{Id}$
(5)	$\mathtt{Match}_{\mathtt{w}} \mathbin{@} (x :: xs) \to (\circ) \mathbin{@} (\mathtt{Fix}_{\mathtt{w}} \mathbin{@} xs) \mathbin{@} \mathtt{Lam}_2(x)$

⇓ inline $\mathtt{Lam}_1$

(2)	$\mathtt{Fix}_{\mathtt{w}} \mathbin{@} l \to \boxed{\mathtt{Match}_{\mathtt{w}} \mathbin{@} l}$
(4)	$\mathtt{Match}_{\mathtt{w}} \mathbin{@} [] \to \mathtt{Id}$
(5)	$\mathtt{Match}_{\mathtt{w}} \mathbin{@} (x :: xs) \to (\circ) \mathbin{@} (\mathtt{Fix}_{\mathtt{w}} \mathbin{@} xs) \mathbin{@} \mathtt{Lam}_2(x)$

⇓ inline $\mathtt{Match}_{\mathtt{w}}$

(2a)	$\mathtt{Fix}_{\mathtt{w}} \mathbin{@} [] \to \mathtt{Id}$
(2b)	$\mathtt{Fix}_{\mathtt{w}} \mathbin{@} (x :: xs) \to (\circ) \mathbin{@} (\mathtt{Fix}_{\mathtt{w}} \mathbin{@} xs) \mathbin{@} \mathtt{Lam}_2(x)$

Inlining

Definition (inlining + narrowing)

replaces a rule $l \to C[\mathtt{f}(t_1, \ldots, t_k)] \in \mathcal{A}$ by

$$\{(l \to C[r])\mu \mid \exists \mathtt{f}(l_1, \ldots, l_k) \to r \in \mathcal{A}, \mathtt{f}(t_1, \ldots, t_k) \approx_\mu \mathtt{f}(l_1, \ldots, l_k)\} \; .$$

Inlining

Definition (inlining + narrowing)

replaces a rule $l \to C[\mathtt{f}(t_1, \ldots, t_k)] \in \mathcal{A}$ by

$$\{(l \to C[r])\mu \mid \exists \mathtt{f}(l_1, \ldots, l_k) \to r \in \mathcal{A}, \mathtt{f}(t_1, \ldots, t_k) \approx_\mu \mathtt{f}(l_1, \ldots, l_k)\}\ .$$

Traps

1. mixes evaluation-order
2. not cost-neutral in general, even asymptotically
   - inline $\mathtt{f}(n) \to \mathtt{0}$ in $\mathtt{g}(m) \to \mathtt{f}(\mathtt{g}(m))$
3. narrowing cause subtle issue when inlined function partially defined
   - inline $\mathtt{f}(n, \mathtt{0}) \to n$ in $\mathtt{g}(\mathtt{S}(m)) \to \mathtt{f}(\mathtt{g}(m), m)$

Inlining

Definition (inlining + narrowing)

replaces a rule $l \to C[\mathtt{f}(t_1, \dots, t_k)] \in \mathcal{A}$ by

$$\{(l \to C[r])\mu \mid \exists \mathtt{f}(l_1, \dots, l_k) \to r \in \mathcal{A}, \mathtt{f}(t_1, \dots, t_k) \approx_\mu \mathtt{f}(l_1, \dots, l_k)\} \,.$$

Traps

1. mixes evaluation-order
2. not cost-neutral in general, even asymptotically
   - inline $\mathtt{f}(n) \to \mathtt{0}$ in $\mathtt{g}(m) \to \mathtt{f}(\mathtt{g}(m))$
3. narrowing cause subtle issue when inlined function partially defined
   - inline $\mathtt{f}(n, \mathtt{0}) \to n$ in $\mathtt{g}(\mathtt{S}(m)) \to \mathtt{f}(\mathtt{g}(m), m)$

Theorem

For non-ambiguous $\mathcal{A}$, redex-preserving inlining of sufficiently defined function $\mathtt{f}$ is asymptotic complexity-reflecting.

Overall...

(1)	$\texttt{Rev} \mathbin{@} l \to \texttt{Fix}_{\texttt{w}} \mathbin{@} l \mathbin{@} \texttt{[]}$	(6)	$(\circ) \mathbin{@} f \to (\circ)_1(f)$
(2)	$\texttt{Fix}_{\texttt{w}} \mathbin{@} l \to \texttt{Lam}_1 \mathbin{@} l$	(7)	$(\circ)_1(f) \mathbin{@} g \to \texttt{Lam}_3(f, g)$
(3)	$\texttt{Lam}_1 \mathbin{@} l \to \texttt{Match}_{\texttt{w}} \mathbin{@} l$	(8)	$\texttt{Lam}_3(f, g) \mathbin{@} z \to f \mathbin{@} (g \mathbin{@} z)$
(4)	$\texttt{Match}_{\texttt{w}} \mathbin{@} \texttt{[]} \to \texttt{Id}$	(9)	$\texttt{Id} \mathbin{@} ys \to ys$
(5)	$\texttt{Match}_{\texttt{w}} \mathbin{@} (x :: xs) \to (\circ) \mathbin{@} (\texttt{Fix}_{\texttt{w}} \mathbin{@} xs) \mathbin{@} \texttt{Lam}_2(x)$	(10)	$\texttt{Lam}_2(x) \mathbin{@} ys \to x :: ys$

$$\Downarrow$$

(1)	$\texttt{Rev} \mathbin{@} l \to \texttt{Fix}_{\texttt{w}} \mathbin{@} l \mathbin{@} \texttt{[]}$	(8)	$\texttt{Lam}_3(f, g) \mathbin{@} z \to f \mathbin{@} (g \mathbin{@} z)$
(2a)	$\texttt{Fix}_{\texttt{w}} \mathbin{@} \texttt{[]} \to \texttt{Id}$	(9)	$\texttt{Id} \mathbin{@} ys \to ys$
(2b)	$\texttt{Fix}_{\texttt{w}} \mathbin{@} (x :: xs) \to \texttt{Lam}_3(\texttt{Fix}_{\texttt{w}} \mathbin{@} xs, \texttt{Lam}_2(x))$	(10)	$\texttt{Lam}_2(x) \mathbin{@} ys \to x :: ys$

Overall...

(1) $\texttt{Rev} \mathbin{@} l \to \texttt{Fix}_{\texttt{w}} \mathbin{@} l \mathbin{@} []$

(2) $\texttt{Fix}_{\texttt{w}} \mathbin{@} l \to \texttt{Lam}_1 \mathbin{@} l$

(3) $\texttt{Lam}_1 \mathbin{@} l \to \texttt{Match}_{\texttt{w}} \mathbin{@} l$

(4) $\texttt{Match}_{\texttt{w}} \mathbin{@} [] \to \texttt{Id}$

(5) $\texttt{Match}_{\texttt{w}} \mathbin{@} (x::xs) \to (\circ) \mathbin{@} (\texttt{Fix}_{\texttt{w}} \mathbin{@} xs) \mathbin{@} \texttt{Lam}_2(x)$

(6) $(\circ) \mathbin{@} f \to (\circ)_1(f)$

(7) $(\circ)_1(f) \mathbin{@} g \to \texttt{Lam}_3(f, g)$

(8) $\texttt{Lam}_3(f, g) \mathbin{@} z \to f \mathbin{@} (g \mathbin{@} z)$

(9) $\texttt{Id} \mathbin{@} ys \to ys$

(10) $\texttt{Lam}_2(x) \mathbin{@} ys \to x::ys$

$\Downarrow$

(1) $\texttt{Rev} \mathbin{@} l \to \texttt{Fix}_{\texttt{w}} \mathbin{@} l \mathbin{@} []$

(2a) $\texttt{Fix}_{\texttt{w}} \mathbin{@} [] \to \texttt{Id}$

(2b) $\texttt{Fix}_{\texttt{w}} \mathbin{@} (x::xs) \to \texttt{Lam}_3(\texttt{Fix}_{\texttt{w}} \mathbin{@} xs, \texttt{Lam}_2(x))$

(8) $\texttt{Lam}_3(f, g) \mathbin{@} z \to f \mathbin{@} (g \mathbin{@} z)$

(9) $\texttt{Id} \mathbin{@} ys \to ys$

(10) $\texttt{Lam}_2(x) \mathbin{@} ys \to x::ys$

★ runtime of Rev coincide, up to constant speed-up

Overall...

(1)	$\mathtt{Rev} \; @ \; l \to \mathtt{Fix_w} \; @ \; l \; @ \; []$	(6)	$(\circ) \; @ \; f \to (\circ)_1(f)$
(2)	$\mathtt{Fix_w} \; @ \; l \to \mathtt{Lam_1} \; @ \; l$	(7)	$(\circ)_1(f) \; @ \; g \to \mathtt{Lam_3}(f, g)$
(3)	$\mathtt{Lam_1} \; @ \; l \to \mathtt{Match_w} \; @ \; l$	(8)	$\mathtt{Lam_3}(f, g) \; @ \; z \to f \; @ \; (g \; @ \; z)$
(4)	$\mathtt{Match_w} \; @ \; [] \to \mathtt{Id}$	(9)	$\mathtt{Id} \; @ \; ys \to ys$
(5)	$\mathtt{Match_w} \; @ \; (x::xs) \to (\circ) \; @ \; (\mathtt{Fix_w} \; @ \; xs) \; @ \; \mathtt{Lam_2}(x)$	(10)	$\mathtt{Lam_2}(x) \; @ \; ys \to x::ys$

$\Downarrow$

(1)	$\mathtt{Rev} \; @ \; l \to \mathtt{Fix_w} \; @ \; l \; @ \; []$	(8)	$\mathtt{Lam_3}(f, g) \; @ \; z \to f \; @ \; (g \; @ \; z)$
(2a)	$\mathtt{Fix_w} \; @ \; [] \to \mathtt{Id}$	(9)	$\mathtt{Id} \; @ \; ys \to ys$
(2b)	$\mathtt{Fix_w} \; @ \; (x::xs) \to \mathtt{Lam_3}(\mathtt{Fix_w} \; @ \; xs, \mathtt{Lam_2}(x))$	(10)	$\mathtt{Lam_2}(x) \; @ \; ys \to x::ys$

- ⋆ runtime of Rev coincide, up to constant speed-up
- ⋆ Implementation Trap: inlining blows up program size/diverge
  - inline conservatively (calls to $\mathtt{Lam_*}$, $\mathtt{Match_*}$, and constants)

Overall...

(1)	$\texttt{Rev} \mathbin{@} l \to \texttt{Fix}_{\texttt{w}} \mathbin{@} l \mathbin{@} []$	(6)	$(\circ) \mathbin{@} f \to (\circ)_1(f)$
(2)	$\texttt{Fix}_{\texttt{w}} \mathbin{@} l \to \texttt{Lam}_1 \mathbin{@} l$	(7)	$(\circ)_1(f) \mathbin{@} g \to \texttt{Lam}_3(f, g)$
(3)	$\texttt{Lam}_1 \mathbin{@} l \to \texttt{Match}_{\texttt{w}} \mathbin{@} l$	(8)	$\texttt{Lam}_3(f, g) \mathbin{@} z \to f \mathbin{@} (g \mathbin{@} z)$
(4)	$\texttt{Match}_{\texttt{w}} \mathbin{@} [] \to \texttt{Id}$	(9)	$\texttt{Id} \mathbin{@} ys \to ys$
(5)	$\texttt{Match}_{\texttt{w}} \mathbin{@} (x :: xs) \to (\circ) \mathbin{@} (\texttt{Fix}_{\texttt{w}} \mathbin{@} xs) \mathbin{@} \texttt{Lam}_2(x)$	(10)	$\texttt{Lam}_2(x) \mathbin{@} ys \to x :: ys$

$\Downarrow$

(1)	$\texttt{Rev} \mathbin{@} l \to \texttt{Fix}_{\texttt{w}} \mathbin{@} l \mathbin{@} []$	(8)	$\texttt{Lam}_3(f, g) \mathbin{@} z \to f \mathbin{@} (g \mathbin{@} z)$
(2a)	$\texttt{Fix}_{\texttt{w}} \mathbin{@} [] \to \texttt{Id}$	(9)	$\texttt{Id} \mathbin{@} ys \to ys$
(2b)	$\texttt{Fix}_{\texttt{w}} \mathbin{@} (x :: xs) \to \texttt{Lam}_3(\texttt{Fix}_{\texttt{w}} \mathbin{@} xs, \texttt{Lam}_2(x))$	(10)	$\texttt{Lam}_2(x) \mathbin{@} ys \to x :: ys$

- ⋆ runtime of Rev coincide, up to constant speed-up
- ⋆ Implementation Trap: inlining blows up program size/diverge
  - – inline conservatively (calls to $\texttt{Lam}_*$, $\texttt{Match}_*$, and constants)
- ⋆ troublesome rule *(8)* still present

Instantiation of Higher-Order Variables

(1)	$\mathtt{Rev} \mathbin{@} l \to \mathtt{Fix_w} \mathbin{@} l \mathbin{@} []$	(8)	$\mathtt{Lam_3}(f, g) \mathbin{@} z \to f \mathbin{@} (g \mathbin{@} z)$
(2a)	$\mathtt{Fix_w} \mathbin{@} [] \to \mathtt{Id}$	(9)	$\mathtt{Id} \mathbin{@} ys \to ys$
(2b)	$\mathtt{Fix_w} \mathbin{@} (x :: xs) \to \mathtt{Lam_3}(\mathtt{Fix_w} \mathbin{@} xs, \mathtt{Lam_2}(x))$	(10)	$\mathtt{Lam_2}(x) \mathbin{@} ys \to x :: ys$

Central Observation:

- ★ seen in isolation, variables f and g can be instantiated arbitrarily
- ★ not so when considering only calls to Rev

Instantiation of Higher-Order Variables

(1) $\texttt{Rev} \mathbin{@} l \to \texttt{Fix}_{\texttt{w}} \mathbin{@} l \mathbin{@} []$	(8) $\texttt{Lam}_3(f, g) \mathbin{@} z \to f \mathbin{@} (g \mathbin{@} z)$
(2a) $\texttt{Fix}_{\texttt{w}} \mathbin{@} [] \to \texttt{Id}$	(9) $\texttt{Id} \mathbin{@} ys \to ys$
(2b) $\texttt{Fix}_{\texttt{w}} \mathbin{@} (x :: xs) \to \texttt{Lam}_3(\texttt{Fix}_{\texttt{w}} \mathbin{@} xs, \texttt{Lam}_2(x))$	(10) $\texttt{Lam}_2(x) \mathbin{@} ys \to x :: ys$

Central Observation:

- ★ seen in isolation, variables f and g can be instantiated arbitrarily
- ★ not so when considering only calls to Rev
- ★ determining precise set of instances undecidable
- ★ but can be efficiently approximated, e.g., with tree automata techniques

N. D. Jones. *"Flow Analysis of Lazy Higher-order Functional Programs"*. *TCS, Vol. 375*, pp. 120–136, 2007.

J. Kochems and L. Ong. *"Improved Functional Flow and Reachability Analyses Using Indexed Linear Tree Grammars"*. In *Proc. of 22nd RTA*, pp. 187–202, 2011.

Instantiation of Higher-Order Variables

(1)	$\mathtt{Rev} \mathbin{@} l \rightarrow \mathtt{Fix_w} \mathbin{@} l \mathbin{@} \mathtt{[]}$	(8)	$\mathtt{Lam_3}(f, g) \mathbin{@} z \rightarrow f \mathbin{@} (g \mathbin{@} z)$
(2a)	$\mathtt{Fix_w} \mathbin{@} \mathtt{[]} \rightarrow \mathtt{Id}$	(9)	$\mathtt{Id} \mathbin{@} ys \rightarrow ys$
(2b)	$\mathtt{Fix_w} \mathbin{@} (x :: xs) \rightarrow \mathtt{Lam_3}(\mathtt{Fix_w} \mathbin{@} xs, \mathtt{Lam_2}(x))$	(10)	$\mathtt{Lam_2}(x) \mathbin{@} ys \rightarrow x :: ys$

$S \rightarrow \mathtt{Rev} \mathbin{@} \star$ $\quad\quad \star \rightarrow \mathtt{[]} \mid \star :: \star$

(1)	$R_1 \rightarrow R_8 \mid R_9$	$L_1 \rightarrow \star$		
(2a)	$R_{2a} \rightarrow \mathtt{Id}$			
(2b)	$R_{2b} \rightarrow \mathtt{Lam_3}(R_{2a}, \mathtt{Lam_2}(X_{2b})) \mid \mathtt{Lam_3}(R_{2b}, \mathtt{Lam_2}(X_{2b}))$	$X_{2b} \rightarrow \star$	$XS_{2b} \rightarrow \star$	
(8)	$R_8 \rightarrow R_8 \mid R_{10}$	$F_8 \rightarrow R_{2a} \mid R_{2b}$	$G_8 \rightarrow \mathtt{Lam_2}(X_{2b})$	$Z_8 \rightarrow \mathtt{[]} \mid R_{10}$
(9)	$R_9 \rightarrow \mathtt{[]} \mid X_{10} \mid YS_{10}$	$YS_9 \rightarrow \mathtt{[]} \mid R_{10}$		
(10)	$R_{10} \rightarrow \mathtt{[]} \mid X_{10} \mid YS_{10}$	$X_{10} \rightarrow X_{2b}$	$YS_{10} \rightarrow \mathtt{[]} \mid R_{10}$	

Tree automaton over-approximating collecting semantics.

Instantiation of Higher-Order Variables

(1)	$\texttt{Rev} \mathbin{@} l \to \texttt{Fix}_{\texttt{w}} \mathbin{@} l \mathbin{@} \texttt{[]}$	(8)	$\texttt{Lam}_3(f, g) \mathbin{@} z \to f \mathbin{@} (g \mathbin{@} z)$
(2a)	$\texttt{Fix}_{\texttt{w}} \mathbin{@} \texttt{[]} \to \texttt{Id}$	(9)	$\texttt{Id} \mathbin{@} ys \to ys$
(2b)	$\texttt{Fix}_{\texttt{w}} \mathbin{@} (x :: xs) \to \texttt{Lam}_3(\texttt{Fix}_{\texttt{w}} \mathbin{@} xs, \texttt{Lam}_2(x))$	(10)	$\texttt{Lam}_2(x) \mathbin{@} ys \to x :: ys$

$S \to \texttt{Rev} \mathbin{@} \star \qquad \star \to \texttt{[]} \mid \star :: \star$

(1)	$R_1 \to R_8 \mid R_9$	$L_1 \to \star$		
(2a)	$R_{2a} \to \texttt{Id}$			
(2b)	$R_{2b} \to \texttt{Lam}_3(R_{2a}, \texttt{Lam}_2(X_{2b})) \mid \texttt{Lam}_3(R_{2b}, \texttt{Lam}_2(X_{2b}))$	$X_{2b} \to \star$	$XS_{2b} \to \star$	
(8)	$R_8 \to R_8 \mid R_{10}$	$F_8 \to R_{2a} \mid R_{2b}$	$G_8 \to \texttt{Lam}_2(X_{2b})$	$Z_8 \to \texttt{[]} \mid R_{10}$
(9)	$R_9 \to \texttt{[]} \mid X_{10} \mid YS_{10}$	$YS_9 \to \texttt{[]} \mid R_{10}$		
(10)	$R_{10} \to \texttt{[]} \mid X_{10} \mid YS_{10}$	$X_{10} \to X_{2b}$	$YS_{10} \to \texttt{[]} \mid R_{10}$	

Tree automaton over-approximating collecting semantics.

$f \mapsto \texttt{Id} \mid \texttt{Lam}_3(f, g) \qquad g \mapsto \texttt{Lam}_2(x)$

Variable bindings extracted from tree automaton.

Instantiation of Higher-Order Variables (II)

(1)	$\texttt{Rev} \mathbin{@} l \longrightarrow \texttt{Fix}_{\texttt{w}} \mathbin{@} l \mathbin{@} \texttt{[]}$	(8)	$\texttt{Lam}_3(f, g) \mathbin{@} z \longrightarrow f \mathbin{@} (g \mathbin{@} z)$
(2a)	$\texttt{Fix}_{\texttt{w}} \mathbin{@} \texttt{[]} \longrightarrow \texttt{Id}$	(9)	$\texttt{Id} \mathbin{@} ys \longrightarrow ys$
(2b)	$\texttt{Fix}_{\texttt{w}} \mathbin{@} (x::xs) \longrightarrow \texttt{Lam}_3(\texttt{Fix}_{\texttt{w}} \mathbin{@} xs, \texttt{Lam}_2(x))$	(10)	$\texttt{Lam}_2(x) \mathbin{@} ys \longrightarrow x::ys$

$+$

$f \mapsto \texttt{Id} \mid \texttt{Lam}_3(f, g)$ $\qquad$ $g \mapsto \texttt{Lam}_2(x)$

$\Downarrow$ instantiate *(8)*

(1)	$\texttt{Rev} \mathbin{@} l \longrightarrow \texttt{Fix}_{\texttt{w}} \mathbin{@} l \mathbin{@} \texttt{[]}$
(2a)	$\texttt{Fix}_{\texttt{w}} \mathbin{@} \texttt{[]} \longrightarrow \texttt{Id}$
(2b)	$\texttt{Fix}_{\texttt{w}} \mathbin{@} (x::xs) \longrightarrow \texttt{Lam}_3(\texttt{Fix}_{\texttt{w}} \mathbin{@} xs, \texttt{Lam}_2(x))$
(8a)	$\texttt{Lam}_3(\texttt{Id}, \texttt{Lam}_2(x)) \mathbin{@} z \longrightarrow \texttt{Id} \mathbin{@} (\texttt{Lam}_2(x) \mathbin{@} z)$
(8b)	$\texttt{Lam}_3(\texttt{Lam}_3(f, g), \texttt{Lam}_2(x)) \mathbin{@} z \longrightarrow \texttt{Lam}_3(f, g) \mathbin{@} (\texttt{Lam}_2(x) \mathbin{@} z)$
(9)	$\texttt{Id} \mathbin{@} ys \longrightarrow ys$
(10)	$\texttt{Lam}_2(x) \mathbin{@} ys \longrightarrow x::ys$

Instantiation of Higher-Order Variables (II)

(1) $\mathtt{Rev} \mathbin{@} l \to \mathtt{Fix_w} \mathbin{@} l \mathbin{@} []$

(2a) $\mathtt{Fix_w} \mathbin{@} [] \to \mathtt{Id}$

(2b) $\mathtt{Fix_w} \mathbin{@} (x::xs) \to \mathtt{Lam_3}(\mathtt{Fix_w} \mathbin{@} xs, \mathtt{Lam_2}(x))$

(8) $\mathtt{Lam_3}(f, g) \mathbin{@} z \to f \mathbin{@} (g \mathbin{@} z)$

(9) $\mathtt{Id} \mathbin{@} ys \to ys$

(10) $\mathtt{Lam_2}(x) \mathbin{@} ys \to x::ys$

+

$f \mapsto \mathtt{Id} \mid \mathtt{Lam_3}(f, g)$ $\qquad$ $g \mapsto \mathtt{Lam_2}(x)$

⇓ instantiate *(8)*, simplify

(1) $\mathtt{Rev} \mathbin{@} l \to \mathtt{Fix_w} \mathbin{@} l \mathbin{@} []$

(2a) $\mathtt{Fix_w} \mathbin{@} [] \to \mathtt{Id}$

(2b) $\mathtt{Fix_w} \mathbin{@} (x::xs) \to \mathtt{Lam_3}(\mathtt{Fix_w} \mathbin{@} xs, \mathtt{Lam_2}(x))$

(8a) $\mathtt{Lam_3}(\mathtt{Id}, \mathtt{Lam_2}(x)) \mathbin{@} z \to x::z$

(8b) $\mathtt{Lam_3}(\mathtt{Lam_3}(f, g), \mathtt{Lam_2}(x)) \mathbin{@} z \to \mathtt{Lam_3}(f, g) \mathbin{@} (x::z)$

(9) $\mathtt{Id} \mathbin{@} ys \to ys$

Instantiation of Higher-Order Variables (II)

(1)	$\mathtt{Rev} \mathbin{@} l \to \mathtt{Fix}_{\mathtt{w}} \mathbin{@} l \mathbin{@} \mathtt{[]}$	(8)	$\mathtt{Lam}_3(f, g) \mathbin{@} z \to f \mathbin{@} (g \mathbin{@} z)$
(2a)	$\mathtt{Fix}_{\mathtt{w}} \mathbin{@} \mathtt{[]} \to \mathtt{Id}$	(9)	$\mathtt{Id} \mathbin{@} ys \to ys$
(2b)	$\mathtt{Fix}_{\mathtt{w}} \mathbin{@} (x :: xs) \to \mathtt{Lam}_3(\mathtt{Fix}_{\mathtt{w}} \mathbin{@} xs, \mathtt{Lam}_2(x))$	(10)	$\mathtt{Lam}_2(x) \mathbin{@} ys \to x :: ys$

$+$

$f \mapsto \mathtt{Id} \mid \mathtt{Lam}_3(f, g)$ $\qquad$ $g \mapsto \mathtt{Lam}_2(x)$

$\Downarrow$ instantiate *(8)*, simplify

(1)	$\mathtt{Rev} \mathbin{@} l \to \mathtt{Fix}_{\mathtt{w}} \mathbin{@} l \mathbin{@} \mathtt{[]}$
(2a)	$\mathtt{Fix}_{\mathtt{w}} \mathbin{@} \mathtt{[]} \to \mathtt{Id}$
(2b)	$\mathtt{Fix}_{\mathtt{w}} \mathbin{@} (x :: xs) \to \mathtt{Lam}_3(\mathtt{Fix}_{\mathtt{w}} \mathbin{@} xs, \mathtt{Lam}_2(x))$
(8a)	$\mathtt{Lam}_3(\mathtt{Id}, \mathtt{Lam}_2(x)) \mathbin{@} z \to x :: z$
(8b)	$\mathtt{Lam}_3(\mathtt{Lam}_3(f, g), \mathtt{Lam}_2(x)) \mathbin{@} z \to \mathtt{Lam}_3(f, g) \mathbin{@} (x :: z)$
(9)	$\mathtt{Id} \mathbin{@} ys \to ys$

★ resulting ATRS head-variable free; applied functions statically known

Uncurrying

$$\mathtt{C}(\vec{s}) \mathbin{\texttt{@}} t_1 \mathbin{\texttt{@}} \cdots \mathbin{\texttt{@}} t_n \qquad \Longrightarrow \qquad \mathtt{C}_n(\vec{s}, t_1, \ldots, t_n)$$

Uncurrying

$$\mathtt{C}(\vec{s}) \mathbin{@} t_1 \mathbin{@} \cdots \mathbin{@} t_n \quad \Longrightarrow \quad \mathtt{C}_n(\vec{s}, t_1, \ldots, t_n)$$

$$\begin{array}{lrcl} (1) & \mathtt{Rev} \mathbin{@} l & \rightarrow & \mathtt{Fix_w} \mathbin{@} l \mathbin{@} [\,] \\ (2a) & \mathtt{Fix_w} \mathbin{@} [\,] & \rightarrow & \mathtt{Id} \\ (2b) & \mathtt{Fix_w} \mathbin{@} (x::xs) & \rightarrow & \mathtt{Lam_3}(\ldots) \\ & & \vdots & \end{array} \quad \Longrightarrow \quad \begin{array}{lrcl} (1) & \mathtt{Rev_1}(l) & \rightarrow & \boxed{\mathtt{Fix^2_w}(l, [\,])} \\ (2a) & \mathtt{Fix^1_w}([\,]) & \rightarrow & \mathtt{Id} \\ (2b) & \mathtt{Fix^1_w}(x::xs) & \rightarrow & \mathtt{Lam_3}(\ldots) \\ & & \vdots & \end{array}$$

Uncurrying

$$\mathtt{C}(\vec{s}) \mathbin{@} t_1 \mathbin{@} \cdots \mathbin{@} t_n \quad \Longrightarrow \quad \mathtt{C}_n(\vec{s}, t_1, \ldots, t_n)$$

(1)	$\mathtt{Rev} \mathbin{@} l \to \mathtt{Fix}_{\mathtt{w}} \mathbin{@} l \mathbin{@} \mathtt{[]}$	
(2a)	$\mathtt{Fix}_{\mathtt{w}} \mathbin{@} \mathtt{[]} \to \mathtt{Id}$	
(2b)	$\mathtt{Fix}_{\mathtt{w}} \mathbin{@} (x :: xs) \to \mathtt{Lam}_3(\ldots)$	
	$\vdots$	

$\Longrightarrow$

(1)	$\mathtt{Rev}_1(l) \to \mathtt{Fix}^2_{\mathtt{w}}(l, \mathtt{[]})$
(2a)	$\mathtt{Fix}^1_{\mathtt{w}}(\mathtt{[]}) \to \mathtt{Id}$
(2b)	$\mathtt{Fix}^1_{\mathtt{w}}(x :: xs) \to \mathtt{Lam}_3(\ldots)$
	$\vdots$

$+\ \eta$-saturate

(2a′)	$\mathtt{Fix}_{\mathtt{w}} \mathbin{@} \mathtt{[]} \mathbin{@} z \to \mathtt{Id} \mathbin{@} z$
(2b′)	$\mathtt{Fix}_{\mathtt{w}} \mathbin{@} (x :: xs) \mathbin{@} z \to \mathtt{Lam}_3(\ldots) \mathbin{@} z$

$\Longrightarrow$

(2a′)	$\mathtt{Fix}^2_{\mathtt{w}}(\mathtt{[]}, z) \to \mathtt{Id}(z)$
(2b′)	$\mathtt{Fix}^2_{\mathtt{w}}(x :: xs, z) \to \mathtt{Lam}^1_3(\ldots, z)$

N. Hirokawa, A. Middeldorp, and H. Zankl. *"Uncurrying for Termination"*. In *Proc. of 15th LPAR*, 2008.

Uncurrying (II)

Definition (η-saturation)

- application arity aa($\mathtt{C}$) is maximal number of arguments applied to $\mathtt{C}$
- ATRS $\mathcal{A}$ is η-saturated if

$$\mathtt{C}(\vec{s}) \mathbin{@} t_1 \mathbin{@} \cdots \mathbin{@} t_n \to r \in \mathcal{A} \implies \mathtt{C}(\vec{s}) \mathbin{@} t_1 \mathbin{@} \cdots \mathbin{@} t_n \mathbin{@} z \to r \mathbin{@} z \in \mathcal{A}$$

  whenever $n < \mathrm{aa}(\mathtt{C})$, with z fresh variable
- η-saturation of $\mathcal{A}$ is least η-saturated extension of $\mathcal{A}$

Uncurrying (II)

Definition (η-saturation)

- application arity $\mathsf{aa}(\mathtt{C})$ is maximal number of arguments applied to $\mathtt{C}$
- ATRS $\mathcal{A}$ is η-saturated if

$$\mathtt{C}(\vec{s}) \mathbin{@} t_1 \mathbin{@} \cdots \mathbin{@} t_n \to r \in \mathcal{A} \implies \mathtt{C}(\vec{s}) \mathbin{@} t_1 \mathbin{@} \cdots \mathbin{@} t_n \mathbin{@} z \to r \mathbin{@} z \in \mathcal{A}$$

  whenever $n < \mathsf{aa}(\mathtt{C})$, with z fresh variable
- η-saturation of $\mathcal{A}$ is least η-saturated extension of $\mathcal{A}$

Theorem (η-Saturation & Uncurrying)

1. *η-saturation finite if $\mathcal{A}$ "well-typed"*
2. *η-saturation is complexity preserving & reflecting*
3. *uncurrying head-variable free, η-saturated ATRS is complexity preserving & reflecting*

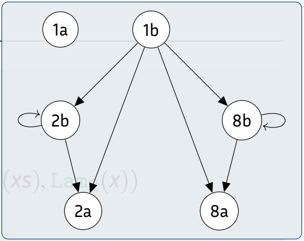
Uncurry (III)

$$\begin{aligned}
&(1) & \mathtt{Rev}_1(l) &\to \mathtt{Fix}^2_{\mathtt{w}}(l, [\,]) \\
&(2a) & \mathtt{Fix}^1_{\mathtt{w}}([\,]) &\to \mathtt{Id} \\
&(2b) & \mathtt{Fix}^1_{\mathtt{w}}(x::xs) &\to \mathtt{Lam}_3(\mathtt{Fix}^1_{\mathtt{w}}(xs), \mathtt{Lam}_2(x)) \\
&(2a') & \mathtt{Fix}^2_{\mathtt{w}}([\,], z) &\to \mathtt{Id}_1(z) \\
&(2b') & \mathtt{Fix}^2_{\mathtt{w}}(x::xs, z) &\to \mathtt{Lam}^1_3(\mathtt{Fix}^1_{\mathtt{w}}(xs), \mathtt{Lam}_2(x), z) \\
&(8a) & \mathtt{Lam}^1_3(\mathtt{Id}, \mathtt{Lam}_2(x), z) &\to x::z \\
&(8b) & \mathtt{Lam}^1_3(\mathtt{Lam}_3(f, g), \mathtt{Lam}_2(x), z) &\to \mathtt{Lam}^1_3(f, g, \mathtt{Lam}^1_2(x, z)) \\
&(9) & \mathtt{Id}_1(ys) &\to ys
\end{aligned}$$

$\Downarrow$ simplify & rename

$$\begin{aligned}
&(1a) & \mathtt{rev}([\,]) &\to [\,] \\
&(1b) & \mathtt{rev}(x::xs) &\to \mathtt{eval}(\mathtt{walk}(xs), \mathtt{Cons}(x), [\,]) \\
&(2a) & \mathtt{walk}([\,]) &\to \mathtt{Id} \\
&(2b) & \mathtt{walk}(x::xs) &\to \mathtt{Comp}(\mathtt{walk}(xs), \mathtt{Cons}(x)) \\
&(8a) & \mathtt{eval}(\mathtt{Id}, \mathtt{Cons}(x), z) &\to x::z \\
&(8b) & \mathtt{eval}(\mathtt{Comp}(f, g), \mathtt{Cons}(x), z) &\to \mathtt{eval}(f, g, x::z)
\end{aligned}$$

Uncurry (III)



$$\texttt{Rev}_1(l) \to \texttt{Fix}_{\texttt{w}}^2(l, [\,]) \tag{1}$$

$$\texttt{Fix}_{\texttt{w}}^1([\,]) \to \texttt{Id} \tag{2a}$$

$$\texttt{Fix}_{\texttt{w}}^1(x::xs) \to \texttt{Lam}_3(\texttt{Fix}_{\texttt{w}}^1(xs), \texttt{Lam}_2(x)) \tag{2b}$$

$$\texttt{Fix}_{\texttt{w}}^2([\,], z) \to \texttt{Id}_1(z) \tag{2a'}$$

$$\texttt{Fix}_{\texttt{w}}^2(x::xs, z) \to \texttt{Lam}_3^1(\texttt{Fix}_{\texttt{w}}^1(xs), \texttt{Lam}_2(x), z) \tag{2b'}$$

$$\texttt{Lam}_3^1(\texttt{Id}, \texttt{Lam}_2(x), z) \to x::z \tag{8a}$$

$$\texttt{Lam}_3^1(\texttt{Lam}_3(f, g), \texttt{Lam}_2(x), z) \to \texttt{Lam}_3^1(f, g, \texttt{Lam}_2^1(x, z)) \tag{8b}$$

$$\texttt{Id}_1(ys) \to ys \tag{9}$$

⇓ simplify & rename

$$\texttt{rev}([\,]) \to [\,] \tag{1a}$$

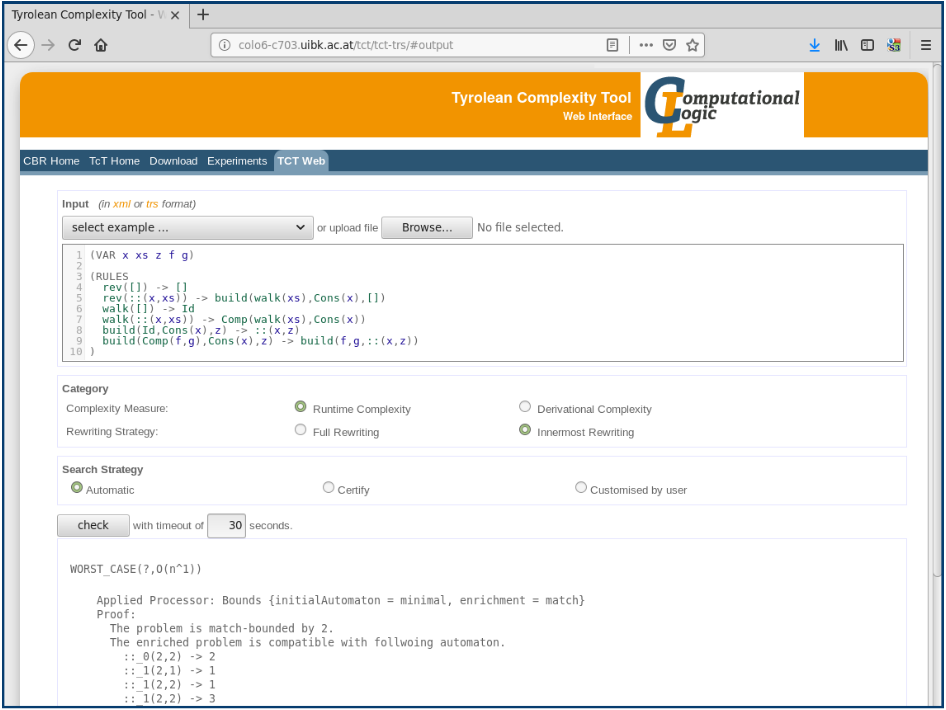
$$\texttt{rev}(x::xs) \to \texttt{eval}(\texttt{walk}(xs), \texttt{Cons}(x), [\,]) \tag{1b}$$

$$\texttt{walk}([\,]) \to \texttt{Id} \tag{2a}$$

$$\texttt{walk}(x::xs) \to \texttt{Comp}(\texttt{walk}(xs), \texttt{Cons}(x)) \tag{2b}$$

$$\texttt{eval}(\texttt{Id}, \texttt{Cons}(x), z) \to x::z \tag{8a}$$

$$\texttt{eval}(\texttt{Comp}(f, g), \texttt{Cons}(x), z) \to \texttt{eval}(f, g, x::z) \tag{8b}$$


Tyrolean Complexity Tool
colo6-c703.uibk.ac.at/tct/tct-trs/#output
Tyrolean Complexity Tool
Web Interface
Computational Logic
CBR Home
TcT Home
Download
Experiments
TCT Web
Input (in xml or trs format)
select example ...
or upload file
Browse...
No file selected.
(VAR x xs z f g)
(RULES
rev([]) -> []
rev(::(x,xs)) -> build(walk(xs),Cons(x),[])
walk([]) -> Id
walk(::(x,xs)) -> Comp(walk(xs),Cons(x))
build(Id,Cons(x),z) -> ::(x,z)
build(Comp(f,g),Cons(x),z) -> build(f,g,::(x,z))
)
Category
Complexity Measure:
Runtime Complexity
Derivational Complexity
Rewriting Strategy:
Full Rewriting
Innermost Rewriting
Search Strategy
Automatic
Certify
Customised by user
check
with timeout of
30
seconds.
WORST_CASE(?,O(n^1))
Applied Processor: Bounds {initialAutomaton = minimal, enrichment = match}
Proof:
The problem is match-bounded by 2.
The enriched problem is compatible with follwoing automaton.
::_0(2,2) -> 2
::_1(2,1) -> 1
::_1(2,2) -> 1
::_1(2,2) -> 3

Experimental Evaluation

- ⋆ Implementation: `http://cbr.uibk.ac.at/tools/hoca/`
- ⋆ FOP: TCTv2 for complexity, TTT2 for termination (SN)
- ⋆ Testbed: 25 higher-order functions from literature on FP
  - higher-order sorting functions, list & tree traversals (maps, folds, ...), Okasaki's parser combinators, ...

		constant	linear	quadratic	poly	SN
RaML	# systems	2	4	8	—	—
	avg. ET (secs)	2.79	0.32	1.55	—	—
Defunctionalize	# systems	2	5	5	5	8
	FOP avg. ET (secs)	1.71	4.82	4.82	4.82	1.38
Simplify	# systems	**2**	**14**	**18**	**20**	**25**
	HoCA avg. ET (secs)	2.28	0.54	0.43	0.42	0.87
	FOP avg. ET (secs)	0.51	2.53	6.30	10.94	1.43

Table: Experimental evaluation on 25 higher-order examples. Defunctionalize: Amortized, type-based analysis with RaML prototoype (`http://raml.co/`). Simplify: FOP on defunctionalized ATRS. RaML: FOP on defunctionalized & simplified ATRS.

Some Relevant Cases

- ⋆ standard examples from literature on functional programming
  - the presented reverse function
  - insert sort defined by `fold`; comparison passed as argument
  - DFS tree flattening via difference lists
  - maximum sequence sum defined via `scanr`
  - ...

⇒ optimal asymptotic bound could be inferred for all examples

Some Relevant Cases

- ⋆ standard examples from literature on functional programming
  - the presented reverse function
  - insert sort defined by `fold`; comparison passed as argument
  - DFS tree flattening via difference lists
  - maximum sequence sum defined via `scanr`
  - ...

  ⇒ optimal asymptotic bound could be inferred for all examples

- ⋆ examples where we can only show termination
  - merge sort
    - instantiation of higher-order divide and conquer combinator [Bird'89]
  - Okasaki's parsing combinators [Okasaki'98]
    - combinators reach order 7
  - lazy/memoized computation of Fibonacci numbers

Conclusion

higher-order functional programs can be effectively analysed with first order tools

Conclusion

higher-order functional programs can be effectively analysed with first order tools

Pros:

- ⋆ fully automatic analysis; no user annotation required
- ⋆ to date, most expressive runtime complexity analysis for higher-order programs

Conclusion

higher-order functional programs can be effectively analysed with first order tools

Pros:

- ⋆ fully automatic analysis; no user annotation required
- ⋆ to date, most expressive runtime complexity analysis for higher-order programs

Cons:

- ⋆ defunctionalisation and CFA analysis ⇒ translation non-modular
  - nowadays, no problem even for compilers (e.g., MLton)
  - modularity within the back-end
- ⋆ fully automatic analysis; no user annotation possible

Conclusion

higher-order functional programs can be effectively analysed with first order tools

Pros:

- ⋆ fully automatic analysis; no user annotation required
- ⋆ to date, most expressive runtime complexity analysis for higher-order programs

Cons:

- ⋆ defunctionalisation and CFA analysis ⇒ translation non-modular
  - nowadays, no problem even for compilers (e.g., MLton)
  - modularity within the back-end
- ⋆ fully automatic analysis; no user annotation possible

- ⋆ same applies to other approaches (e.g. for JBC or Prolog)

Thank You!

- ⋆ HoCA http://cbr.uibk.ac.at/tools/hoca
- ⋆ TCT http://cl-informatik.uibk.ac.at/software/tct