Automated Complexity Analysis via Term Rewriting Techniques

Martin Avanzini Georg Moser

Institute of Computer Science
University of Innsbruck, Austria

September 6, 2013

Term Rewrite Systems (TRSs)

Example

TRS $\mathcal{R}_{\text{rev}}$ consists of rules

$$\begin{array}{llll} 1: & [] +\!\!+ ys \to ys & 3: & \text{rev}([]) \to [] \\ 2: & (x:xs) +\!\!+ ys \to x:(xs +\!\!+ ys) & 4: & \text{rev}(x:xs) \to \text{rev}(xs) +\!\!+ (x:[]) \end{array}$$

Rewriting

$$\underline{\text{rev}(1:(2:(3:[])))} \to_{\mathcal{R}_{\text{rev}}} \text{rev}(2:(3:[])) +\!\!+ (1:[]) \to^*_{\mathcal{R}_{\text{rev}}} 3:(2:(1:[]))$$

Semantics

Term Rewrite Systems (TRSs)

Example

TRS $\mathcal{R}_{\text{rev}}$ consists of rules

$$\begin{array}{llll} 1: & [] \mathbin{+\!\!+} ys \to ys & 3: & \text{rev}([]) \to [] \\ 2: & (x:xs) \mathbin{+\!\!+} ys \to x:(xs \mathbin{+\!\!+} ys) & 4: & \text{rev}(x:xs) \to \text{rev}(xs) \mathbin{+\!\!+} (x:[]) \end{array}$$

Rewriting

$$\underline{\text{rev}(1:(2:(3:[])))} \to_{\mathcal{R}_{\text{rev}}} \text{rev}(2:(3:[])) \mathbin{+\!\!+} (1:[]) \to^*_{\mathcal{R}_{\text{rev}}} 3:(2:(1:[]))$$

Semantics

Term Rewrite Systems (TRSs)

Example

TRS $\mathcal{R}_{\text{rev}}$ consists of rules

$$\begin{array}{llll} 1: & [] \mathbin{+\!\!+} ys \to ys & 3: & \text{rev}([]) \to [] \\ 2: & (x:xs) \mathbin{+\!\!+} ys \to x:(xs \mathbin{+\!\!+} ys) & 4: & \text{rev}(x:xs) \to \text{rev}(xs) \mathbin{+\!\!+} (x:[]) \end{array}$$

Rewriting

$$\underline{\text{rev}(1:(2:(3:[])))} \to_{\mathcal{R}_{\text{rev}}} \text{rev}(2:(3:[])) \mathbin{+\!\!+} (1:[]) \to^*_{\mathcal{R}_{\text{rev}}} 3:(2:(1:[]))$$

Semantics

Term Rewrite Systems (TRSs)

Example

TRS $\mathcal{R}_{\text{rev}}$ consists of rules

$$\begin{array}{llll} 1: & [] \mathbin{+\!\!+} ys \to ys & 3: & \text{rev}([]) \to [] \\ 2: & (x:xs) \mathbin{+\!\!+} ys \to x:(xs \mathbin{+\!\!+} ys) & 4: & \text{rev}(x:xs) \to \text{rev}(xs) \mathbin{+\!\!+} [x] \end{array}$$

Rewriting

$$\underline{\text{rev}([1,2,3])} \to_{\mathcal{R}_{\text{rev}}} \text{rev}([2,3]) \mathbin{+\!\!+} [1] \to^*_{\mathcal{R}_{\text{rev}}} [3,2,1]$$

Semantics

Term Rewrite Systems (TRSs)

Example

TRS $\mathcal{R}_{\text{rev}}$ consists of rules

$$\begin{array}{llll} 1: & [] \mathbin{+\!\!+} ys \to ys & 3: & \text{rev}([]) \to [] \\ 2: & (x:xs) \mathbin{+\!\!+} ys \to x:(xs \mathbin{+\!\!+} ys) & 4: & \text{rev}(x:xs) \to \text{rev}(xs) \mathbin{+\!\!+} [x] \end{array}$$

Rewriting

$$\underline{\text{rev}([1,2,3])} \to_{\mathcal{R}_{\text{rev}}} \text{rev}([2,3]) \mathbin{+\!\!+} [1] \to^*_{\mathcal{R}_{\text{rev}}} [3,2,1]$$

Semantics

$\mathcal{V}\text{al} := \text{constructor terms}$

for each k-ary f define (partial) function $[\![\text{f}]\!] : \mathcal{V}\text{al}^k \to \mathcal{V}\text{al}$ s.t.

$$[\![\text{f}]\!](u_1, \ldots, u_k) = v \quad :\Longleftrightarrow \quad \text{f}(u_1, \ldots, u_k) \to^*_{\mathcal{R}} v \text{ and } v \in \mathcal{V}\text{al}$$

Term Rewrite Systems (TRSs)

Example

TRS $\mathcal{R}_{\text{rev}}$ consists of rules

$$\begin{array}{llll} 1: & [] \mathbin{+\!\!+} ys \to ys & 3: & \text{rev}([]) \to [] \\ 2: & (x:xs) \mathbin{+\!\!+} ys \to x:(xs \mathbin{+\!\!+} ys) & 4: & \text{rev}(x:xs) \to \text{rev}(xs) \mathbin{+\!\!+} [x] \end{array}$$

Rewriting

$$\underline{\text{rev}([1,2,3])} \to_{\mathcal{R}_{\text{rev}}} \text{rev}([2,3]) \mathbin{+\!\!+} [1] \to^*_{\mathcal{R}_{\text{rev}}} [3,2,1]$$

Semantics

$$[\![\mathbin{+\!\!+}]\!]([e_1,\ldots,e_n],[f_1,\ldots,f_m]) := [e_1,\ldots,e_n,f_1,\ldots,f_m]$$
$$[\![\text{rev}]\!]([e_1,\ldots,e_n]) := [e_n,\ldots,e_1]$$

Complexity of TRSs

- **derivation height** of term t with respect to relation $\to$

$$\mathsf{dh}(t, \to) = \max\{\ell \mid \exists(t_1, \ldots, t_\ell).\ t \to t_1 \to \cdots \to t_\ell\}$$

1. **derivational complexity** of TRS $\mathcal{R}$

$$\mathsf{dc}_\mathcal{R}(n) = \max\{\, \mathsf{dh}(f(\vec{s}), \to_\mathcal{R}) \mid f(\vec{s}) \text{ of size upto } n \,\}$$

2. **runtime complexity** of TRS $\mathcal{R}$

$$\mathsf{rc}_\mathcal{R}(n) = \max\{\, \mathsf{dh}(f(\vec{v}), \to_\mathcal{R}) \mid \underbrace{f(\vec{v}) \text{ and } \vec{v} \text{ are values}}_{\text{basic term}} \text{ of size upto } n \,\}$$

Tyrolean Complexity Tool TCT

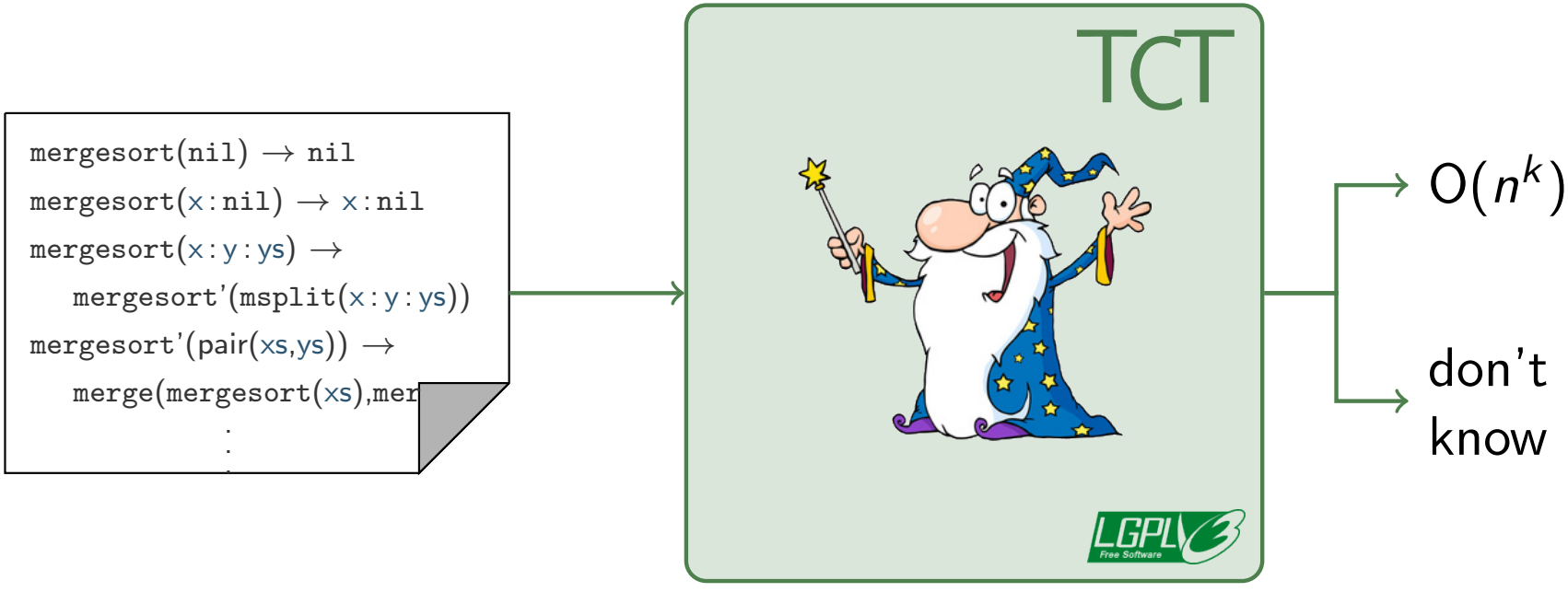
- (runtime) complexity analyser for term rewrite systems (TRSs)

  `http://cl-informatik.uibk.ac.at/software/tct`

Tyrolean Complexity Tool TCT

- (runtime) complexity analyser for term rewrite systems (TRSs)

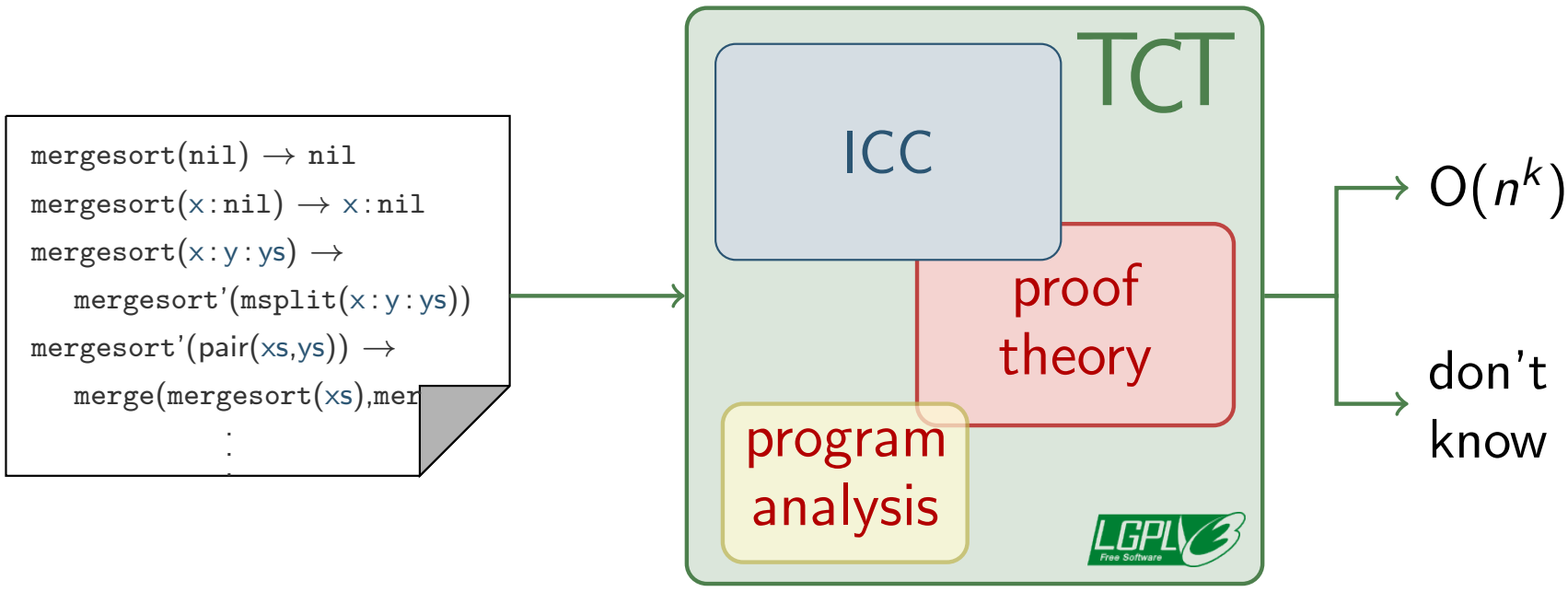
  http://cl-informatik.uibk.ac.at/software/tct



Tyrolean Complexity Tool TCT

- (runtime) complexity analyser for term rewrite systems (TRSs)

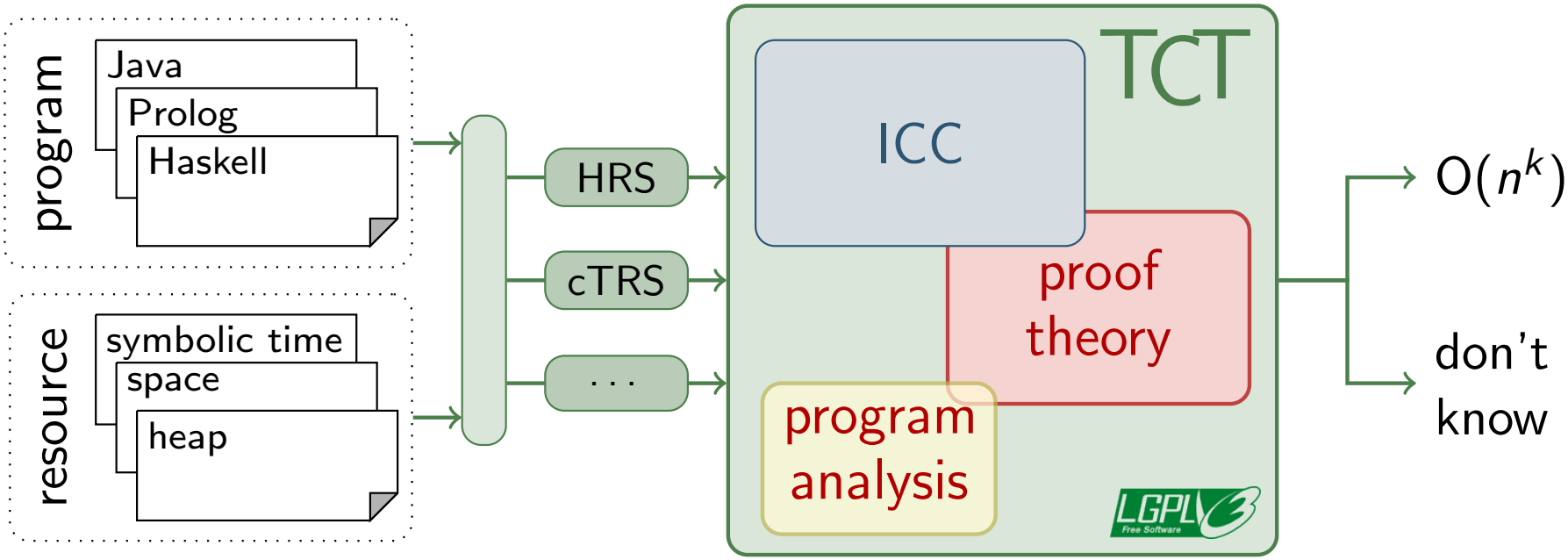
  http://cl-informatik.uibk.ac.at/software/tct



Tyrolean Complexity Tool TCT

- (runtime) complexity analyser for term rewrite systems (TRSs)

  `http://cl-informatik.uibk.ac.at/software/tct`



History

2008 **version 1.0** *extension to termination prover T_TT_2*

- 3 dedicated complexity techniques

2009 **version 1.5** *new implementation*

- in `Haskell`
- 9 methods implemented
- $\approx$ 3.400 lines of code

2013 **version 2.0** *current version*

- 23 methods implemented
- $\approx$ 13.000 lines of code / 4.000 lines of comment

History

2008 **version 1.0** *extension to termination prover T_TT_2*

- 3 dedicated complexity techniques

2009 **version 1.5** *new implementation*

- in `Haskell`
- 9 methods i
- $\approx$ 3.400 lines of code

demo

2013 **version 2.0** *current version*

- 23 methods implemented
- $\approx$ 13.000 lines of code / 4.000 lines of comment

- introduction to TCT

combination framework of TCT

- dependency pairs for complexity analysis

Complexity Problem

abstract

- **complexity problem** $\mathcal{P}$ is tuple $\langle \to, \mathcal{T} \rangle$
  1. $\to$ is binary relation on terms
  2. $\mathcal{T}$ is set of starting terms
- **complexity function** of $\mathcal{P}$ is

$$\mathsf{cp}_{\mathcal{P}}(n) := \max\{\mathsf{dh}(t, \to) \mid t \in \mathcal{T} \text{ is term of size upto } n\}$$

Complexity Problem

concrete

- **complexity problem** $\mathcal{P}$ is tuple $\langle \mathcal{S}/\mathcal{W}, \mathcal{Q}, \mathcal{T} \rangle$
  1. $\to$ is binary relation on terms

$$\xrightarrow{\mathcal{Q}}_{\mathcal{S}/\mathcal{W}} := \xrightarrow{\mathcal{Q}}_{\mathcal{W}}^{*} \cdot \xrightarrow{\mathcal{Q}}_{\mathcal{S}} \cdot \xrightarrow{\mathcal{Q}}_{\mathcal{W}}^{*}$$

     - $\mathcal{S}$,$\mathcal{W}$ are TRSs
     - $s \xrightarrow{\mathcal{Q}}_{\mathcal{R}} t$ if $s \to_{\mathcal{R}} t$ and arguments of redex in s are $\mathcal{Q}$ normal forms

  2. $\mathcal{T}$ is set of starting terms

- **complexity function** of $\mathcal{P}$ is

$$\mathsf{cp}_{\mathcal{P}}(n) := \max\{\mathsf{dh}(t, \xrightarrow{\mathcal{Q}}_{\mathcal{S}/\mathcal{W}}) \mid t \in \mathcal{T} \text{ is term of size upto } n\}$$

- **canonical complexity problem**

$$\langle \mathcal{R}/\varnothing, \varnothing, \texttt{basic terms} \rangle$$

Complexity Problem

concrete

- **complexity problem** $\mathcal{P}$ is tuple $\langle \mathcal{S}/\mathcal{W}, \mathcal{Q}, \mathcal{T} \rangle$
  1. $\to$ is binary relation on terms

$$\xrightarrow{\mathcal{Q}}_{\mathcal{S}/\mathcal{W}} := \xrightarrow{\mathcal{Q}}_{\mathcal{W}}^{*} \cdot \xrightarrow{\mathcal{Q}}_{\mathcal{S}} \cdot \xrightarrow{\mathcal{Q}}_{\mathcal{W}}^{*}$$

     - $\mathcal{S}$,$\mathcal{W}$ are TRSs
     - $s \xrightarrow{\mathcal{Q}}_{\mathcal{R}} t$ if $s \to_{\mathcal{R}} t$ and arguments of redex in s are $\mathcal{Q}$ normal forms

  2. $\mathcal{T}$ is set of starting terms

- **complexity function** of $\mathcal{P}$ is

$$\mathrm{cp}_{\mathcal{P}}(n) := \max\{\mathrm{dh}(t, \xrightarrow{\mathcal{Q}}_{\mathcal{S}/\mathcal{W}}) \mid t \in \mathcal{T} \text{ is term of size upto } n\}$$

- **canonical complexity problem**

$$\langle \mathcal{R}/\varnothing, \varnothing, \texttt{basic terms} \rangle$$

Complexity Judgements, Processors and Proofs

complexity processor is inference rule

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

complexity judgement is statement $\vdash \mathcal{P} : f$

- $\mathcal{P}$ is a complexity problem
- $f : \mathbb{N} \to \mathbb{N}$ is bounding function
- valid if $\mathrm{cp}_{\mathcal{P}}(n) \in \mathrm{O}(f(n))$

complexity proof of $\vdash \mathcal{P} : f$ is a deduction using sound processors only.

Complexity Judgements, Processors and Proofs

complexity processor is inference rule

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

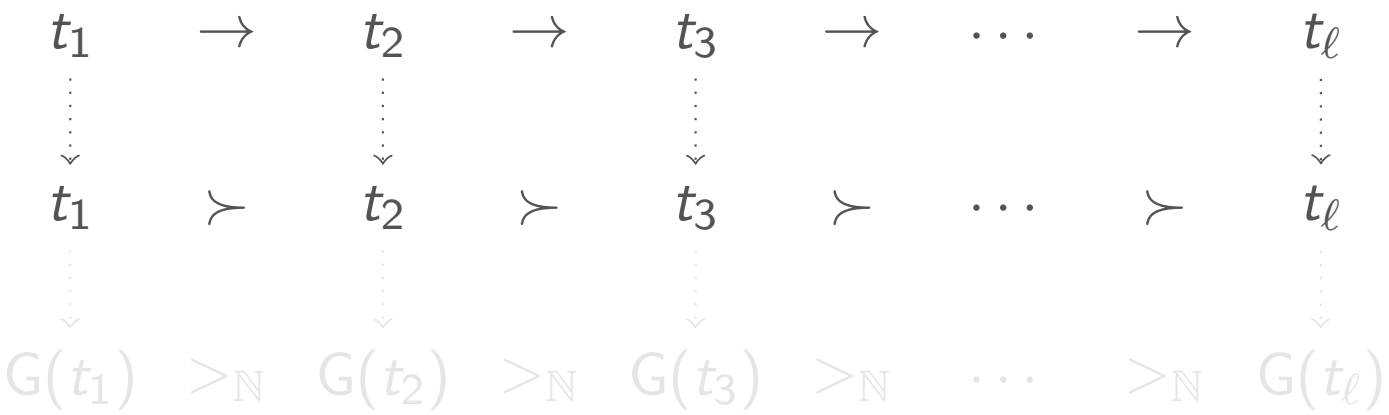
complexity judgement is statement $\vdash \mathcal{P} : f$

- $\mathcal{P}$ is a complexity problem
- $f : \mathbb{N} \to \mathbb{N}$ is bounding function
- valid if $\mathrm{cp}_{\mathcal{P}}(n) \in \mathrm{O}(f(n))$

complexity proof of $\vdash \mathcal{P} : f$ is a deduction using sound processors only.

Orders for Complexity

idea



- order called collapsible on $\to$ if for mapping $\mathsf{G} : \mathtt{terms} \to \mathbb{N}$

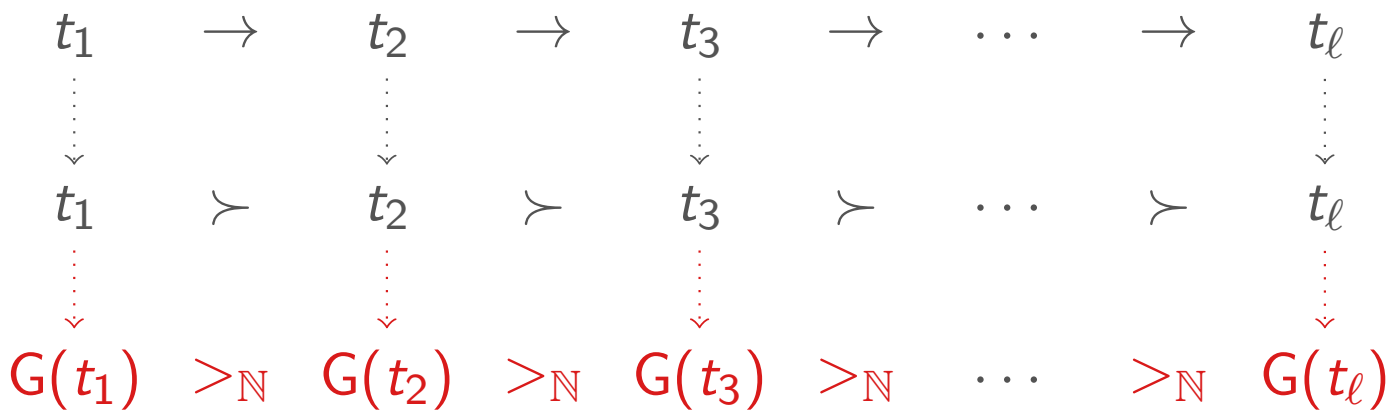
  $s \succ t$ implies $\mathsf{G}(s) >_{\mathbb{N}} \mathsf{G}(t)$ for all steps $s \to t$

- order $\succ$ induces complexity bound $f : \mathbb{N} \to \mathbb{N}$ on terms $\mathcal{T}$

  $\mathsf{G}(t) \leqslant f(\mathsf{size}(t))$ for all $t \in \mathcal{T}$

Orders for Complexity

idea



- order called collapsible on $\to$ if for mapping $\mathsf{G} : \mathtt{terms} \to \mathbb{N}$

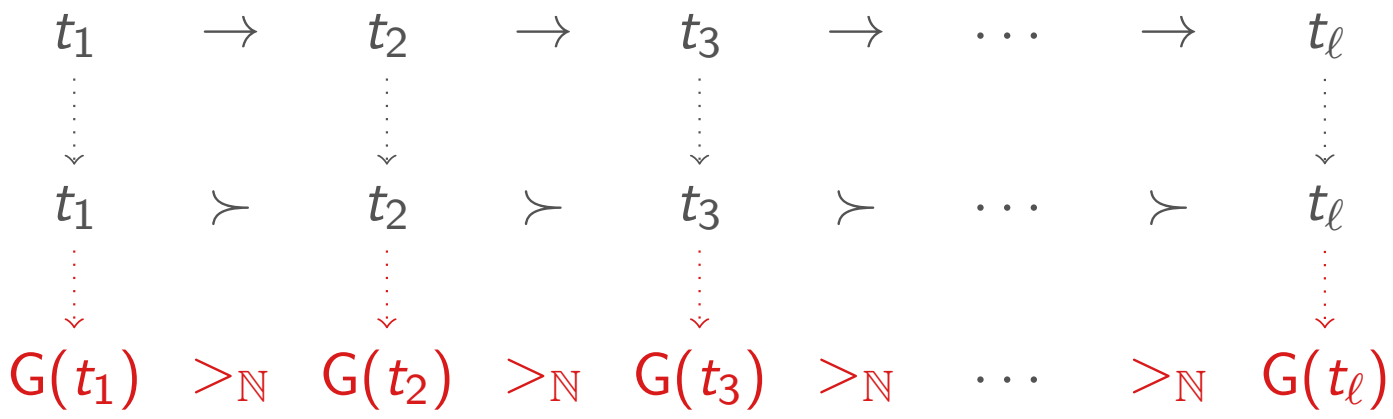
  $s \succ t$ implies $\mathsf{G}(s) >_{\mathbb{N}} \mathsf{G}(t)$ for all steps $s \to t$

- order $\succ$ induces complexity bound $f : \mathbb{N} \to \mathbb{N}$ on terms $\mathcal{T}$

  $\mathsf{G}(t) \leqslant f(\mathsf{size}(t))$ for all $t \in \mathcal{T}$

Orders for Complexity

idea



- order called collapsible on $\to$ if for mapping $\mathsf{G} : \mathtt{terms} \to \mathbb{N}$

  $s \succ t$ implies $\mathsf{G}(s) >_{\mathbb{N}} \mathsf{G}(t)$ for all steps $s \to t$

- order $\succ$ induces complexity bound $f : \mathbb{N} \to \mathbb{N}$ on terms $\mathcal{T}$

  $\mathsf{G}(t) \leqslant f(\mathsf{size}(t))$ for all $t \in \mathcal{T}$

Orders for Complexity

complexity pair processor

$$\frac{\mathcal{S} \subseteq \succ \quad \mathcal{W} \subseteq \succcurlyeq}{\vdash \langle \mathcal{S}/\mathcal{W}, \mathcal{T} \rangle : f}$$

1. order $\succ$ induces complexity $f : \mathbb{N} \to \mathbb{N}$ on $\mathcal{T}$
2. $(\succcurlyeq, \succ)$ are stable under substitutions and monotone
   - $\rightsquigarrow$ order $\succcurlyeq$ "*monotone under $\mathcal{W}$-reducible argument positions*"
   - $\rightsquigarrow$ order $\succ$ "*monotone under $\mathcal{S}$-reducible argument positions*"
3. $\succcurlyeq$ preorder with $\succcurlyeq \cdot \succ \cdot \succcurlyeq \subseteq \succ$

$$\begin{array}{ccccccccc}
t_i & \to_{\mathcal{W}}^* & \cdot & \to_{\mathcal{S}} & \cdot & \to_{\mathcal{W}}^* & t_{i+1} & & \\
\vdots & & \vdots & & \vdots & & \vdots & & \\
t_i & \succcurlyeq^* & \cdot & \succ & \cdot & \succcurlyeq^* & t_{i+1} & & \text{by } ② \\
\vdots & & & & & & \vdots & & \\
t_i & & & \succ & & & t_{i+1} & & \text{by } ③
\end{array}$$

Orders for Complexity

complexity pair processor

$$\frac{\mathcal{S} \subseteq \succ \quad \mathcal{W} \subseteq \succcurlyeq}{\vdash \langle \mathcal{S}/\mathcal{W}, \mathcal{T} \rangle : f}$$

1. order $\succ$ induces complexity $f : \mathbb{N} \to \mathbb{N}$ on $\mathcal{T}$
2. $(\succcurlyeq, \succ)$ are stable under substitutions and $\mathcal{P}$-monotone
   - $\sim$ order $\succcurlyeq$ "*monotone under $\mathcal{W}$-reducible argument positions*"
   - $\sim$ order $\succ$ "*monotone under $\mathcal{S}$-reducible argument positions*"
3. $\succcurlyeq$ preorder with $\succcurlyeq \cdot \succ \cdot \succcurlyeq \subseteq \succ$

$$\begin{array}{ccccccc} t_i & \to_{\mathcal{W}}^* & \cdot & \to_{\mathcal{S}} & \cdot & \to_{\mathcal{W}}^* & t_{i+1} \\ \vdots & & \vdots & & \vdots & & \vdots \\ t_i & \succcurlyeq^* & \cdot & \succ & \cdot & \succcurlyeq^* & t_{i+1} \\ \vdots & & & & & & \vdots \\ t_i & & & \succ & & & t_{i+1} \end{array} \qquad \begin{array}{l} \\ \\ \text{by } ② \\ \\ \text{by } ③ \end{array}$$

Orders for Complexity

instances

recursive path orders

$$f(\mathsf{s}(x); y) >_{\mathsf{pop}*} g(x; f(x; y)) \qquad \text{if } f > g$$

interpretation $[\![\cdot]\!]$ maps terms into domain A, equipped with order $>_A$

$$s \succ t := [\![s]\!]_\alpha >_A [\![t]\!]_\alpha \qquad \text{for all assignments } \alpha$$
$$s \succcurlyeq t := [\![s]\!]_\alpha \geqslant_A [\![t]\!]_\alpha \qquad \text{for all assignments } \alpha$$

1. **polynomial interpretations** over $\mathbb{N}$ $\qquad [\![f]\!](x, y, z) = x^2 + 2xy + 1$
2. **matrix interpretation** over $\mathbb{N}^n$

$$[\![f]\!](x, y, z) = \begin{pmatrix} 0 & 1 \\ 3 & 1 \end{pmatrix} \cdot x + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \cdot y + \begin{pmatrix} 3 \\ 0 \end{pmatrix}$$

3. ...

Orders for Complexity

instances

recursive path orders

$$f(\mathsf{s}(x); y) >_{\mathsf{pop}*} g(x; f(x; y)) \qquad \text{if } f > g$$

interpretation $[\![\cdot]\!]$ maps terms into domain A, equipped with order $>_A$

$$s \succ t := [\![s]\!]_\alpha >_A [\![t]\!]_\alpha \qquad \text{for all assignments } \alpha$$
$$s \succcurlyeq t := [\![s]\!]_\alpha \geqslant_A [\![t]\!]_\alpha \qquad \text{for all assignments } \alpha$$

1. **polynomial interpretations** over $\mathbb{N}$ $\qquad [\![f]\!](x, y, z) = x^2 + 2xy + 1$
2. **matrix interpretation** over $\mathbb{N}^n$

$$[\![f]\!](x, y, z) = \begin{pmatrix} 0 & 1 \\ 3 & 1 \end{pmatrix} \cdot x + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \cdot y + \begin{pmatrix} 3 \\ 0 \end{pmatrix}$$

3. ...

Orders for Complexity

instances

recursive path orders

$$f(\mathsf{s}(x); y) >_{\mathsf{pop}*} g(x; f(x; y)) \qquad \text{if } f > g$$

interpretation $[\![\cdot]\!]$ maps terms into domain A, equipped with order $>_A$

$$s \succ t := [\![s]\!]_\alpha >_A [\![t]\!]_\alpha \qquad \text{for all assignments } \alpha$$
$$s \succcurlyeq t := [\![s]\!]_\alpha \geqslant_A [\![t]\!]_\alpha \qquad \text{for all assignments } \alpha$$

1. **polynomial interpretations** over $\mathbb{N}$ $\qquad [\![f]\!](x, y, z) = x^2 + 2xy + 1$
2. **matrix interpretation** over $\mathbb{N}^n$
$$[\![f]\!](x, y, z) = \begin{pmatrix} 0 & 1 \\ 3 & 1 \end{pmatrix} \cdot x + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \cdot y + \begin{pmatrix} 3 \\ 0 \end{pmatrix}$$
3. ...

Orders For Complexity

practice

+ very powerful in theory
− difficult to synthesise
  1. disaster for systems with many rules
  2. implementations usually do not go beyond induced complexity n^3

⇒ decomposition into manageable pieces

  1. simplify rules
  2. reduce complexity

Experimental Evaluation

testsuite

straight forward translations of resource aware ML programs

machine

workstation with 12 Intel® Core™ i7-3930K (3.20GHz)

Experimental Evaluation

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

Orders For Complexity

practice

+ very powerful in theory
− very very very difficult to synthesise efficiently
  1. disaster for systems with many rules
  2. implementations usually do not go beyond induced complexity n^3

$\Rightarrow$ decomposition into manageable pieces

  1. simplify rules
  2. reduce complexity

Additive Decomposition

$$\frac{\vdash \langle \mathcal{S}_1/\mathcal{S}_2 \cup \mathcal{W}, \mathcal{T}\rangle : f_1 \qquad \vdash \langle \mathcal{S}_2/\mathcal{S}_1 \cup \mathcal{W}, \mathcal{T}\rangle : f_2}{\vdash \langle \mathcal{S}_1 \cup \mathcal{S}_2/\mathcal{W}, \mathcal{T}\rangle : f_1 + f_2}$$

Harald Zankl and Martin Korp
Modular Complexity Analysis via Relative Complexity.
Proc. 21st RTA, pages 385–400, 2010

Additive Decomposition

$$\dfrac{\dfrac{\mathcal{S}_1 \subseteq \succ \quad \mathcal{S}_2 \cup \mathcal{W} \subseteq \succcurlyeq}{\vdash \langle \mathcal{S}_1 / \mathcal{S}_2 \cup \mathcal{W}, \mathcal{T} \rangle : f_1} \qquad \vdash \langle \mathcal{S}_2 / \mathcal{S}_1 \cup \mathcal{W}, \mathcal{T} \rangle : f_2}{\vdash \langle \mathcal{S}_1 \cup \mathcal{S}_2 / \mathcal{W}, \mathcal{T} \rangle : f_1 + f_2}$$

Harald Zankl and Martin Korp
Modular Complexity Analysis via Relative Complexity.
Proc. 21st RTA, pages 385–400, 2010

Experimental Evaluation

Input	#rules	direct	decompose	DG decompose	secs
appendAll	12	$O(n^2)$	$O(n^2)$	$O(n)$	2.9
bfs	57	?	?	$O(n)$	27.9
bft mmult	59	?	?	$O(n^3)$	55.3
bitonic	78	?	?	$O(n^4)$	143.0
bitvectors	148	?	?	$O(n^2)$	35.5
clevermmult	39	?	?	$O(n^2)$	220.0
duplicates	37	?	$O(n^2)$	$O(n^2)$	3.5
dyade	31	?	?	$O(n^2)$	1.0
eratosthenes	74	?	$O(n^3)$	$O(n^2)$	15.8
flatten	31	?	?	$O(n^2)$	25.0
insertionsort	36	?	$O(n^3)$	$O(n^2)$	6.7
listsort	56	?	?	$O(n^2)$	22.8
lcs	87	?	?	$O(n^2)$	24.5
matrix	74	?	?	$O(n^3)$	72.4
mergesort	35	?	?	$O(n^3)$	29.0
minsort	26	?	$O(n^3)$	$O(n^2)$	2.4
queue	35	?	?	$O(n^5)$	148.4
quicksort	46	?	?	$O(n^2)$	38.4
rationalPotential	14	$O(n)$	$O(n)$	$O(n)$	0.27
splitandsort	70	?	?	$O(n^3)$	61.4
subtrees	8	?	$O(n^2)$	$O(n^2)$	3.6
tuples	33	?	?	?	–

- introduction to TCT
- combination framework of TCT

dependency pairs for complexity analysis

Dependency Pair for Complexity Analysis

- dependency pair for termination is rule $l^\sharp \to r^\sharp$
  - $l, r \in \mathcal{T}(\mathcal{F}, \mathcal{V})$
  - $f(t_1, \ldots, t_n)^\sharp = f^\sharp(t_1, \ldots, t_n)$ and $x^\sharp = x$ otherwise

Example

dependency pairs of $\mathcal{R}_{\mathsf{rev}}$ are

$$a: \quad \mathsf{rev}^\sharp(x:xs) \to \mathsf{rev}(xs) +\!\!+^\sharp [x] \qquad c: \quad \mathsf{rev}^\sharp(x:xs) \to \mathsf{rev}^\sharp(xs)$$

$$b: \quad (x:xs) +\!\!+^\sharp ys \to xs +\!\!+^\sharp ys$$

$$\mathsf{SN}(\to_{\mathcal{R}}) \quad \Leftrightarrow \quad \mathsf{SN}(\to_{\mathsf{DP}(\mathcal{R})/\mathcal{R}})$$

Dependency Pair for Complexity Analysis

- dependency pair for complexity is rule $l^\sharp \to \mathsf{c}_n(r_1^\sharp, \ldots, r_n^\sharp)$
  - $l, r_1, \ldots, r_n \in \mathcal{T}(\mathcal{F}, \mathcal{V})$
  - $f(t_1, \ldots, t_n)^\sharp = f^\sharp(t_1, \ldots, t_n)$ and $x^\sharp = x$ otherwise

Example

dependency pairs of $\mathcal{R}_{\mathsf{rev}}$ are

$$a: \quad \mathsf{rev}^\sharp(x : xs) \to \mathsf{c}_2(\mathsf{rev}(xs) +\!\!+^\sharp [x], \mathsf{rev}^\sharp(xs))$$

$$b: \quad (x : xs) +\!\!+^\sharp ys \to xs +\!\!+^\sharp ys$$

Dependency Pair Complexity Problems

DP complexity problem is $\langle \mathcal{S}^\sharp / \mathcal{W}^\sharp \cup \mathcal{W}, \mathcal{T}^\sharp \rangle$

- $\mathcal{S}^\sharp, \mathcal{W}^\sharp$ are two sets of dependency pairs
- $\mathcal{W}$ is TRS as before
- $\mathcal{T}^\sharp$ some marked and basic terms

Dependency Pair Complexity Problems

DP complexity problem is $\langle \mathcal{S}^\sharp / \mathcal{W}^\sharp \cup \mathcal{W}, \mathcal{T}^\sharp \rangle$

- $\mathcal{S}^\sharp, \mathcal{W}^\sharp$ are two sets of dependency pairs
- $\mathcal{W}$ is TRS as before
- $\mathcal{T}^\sharp$ some marked and basic terms

canonical DP problems

Nao Hirokawa and Georg Moser
Automated Complexity Analysis Based on the Dependency Pair Method
Proc. 4th IJCAR, pages 364–379, 2008.

Lars Noschinski, Fabian Emmes, and Jürgen Giesl
A Dependency Pair Framework for Innermost Complexity Analysis of Term Rewrite Systems
Proc. 23th CADE, pages 422–438, 2011.

Dependency Pair Complexity Problems

DP complexity problem is $\langle \mathcal{S}^\sharp / \mathcal{W}^\sharp \cup \mathcal{W}, \mathcal{T}^\sharp \rangle$

- $\mathcal{S}^\sharp, \mathcal{W}^\sharp$ are two sets of dependency pairs
- $\mathcal{W}$ is TRS as be
- $\mathcal{T}^\sharp$ some marke

canonical DP prob

$$\frac{\vdash \langle \{\mathtt{a}, \mathtt{b}\} / \mathcal{R}_{\mathsf{rev}}, \mathtt{basic\ terms}^\sharp \rangle : f}{\begin{array}{c}\vdots\end{array}}$$
$$\vdash \langle \mathcal{R}_{\mathsf{rev}} / \varnothing, \mathtt{basic\ terms} \rangle : f$$

a : $\mathsf{rev}^\sharp(x : xs) \to \mathsf{c}_2(\mathsf{rev}(xs) +\!\!+^\sharp [x], \mathsf{rev}^\sharp(xs))$

b : $(x : xs) +\!\!+^\sharp ys \to xs +\!\!+^\sharp ys$

Nao Hirokawa a
Automated Com
Method
Proc. 4th IJCAR

Lars Noschinski, Fabian Emmes, and Jürgen Giesl
A Dependency Pair Framework for Innermost Complexity Analysis of Term Rewrite Systems
Proc. 23th CADE, pages 422–438, 2011.

Dependency Pair Processors

- **reduction pairs**
- **simplifications**
  usable rules, narrowing, simplification of right-hand sides, ...
- **dependency graph analysis**
  path analysis, decomposition techniques, ...

Dependency Pair Processors

- **reduction pairs**
- **simplifications**
  usable rules, narrowing, simplification of right-hand sides, ...
- **dependency graph analysis**
  path analysis, decomposition techniques, ...

Dependency Graph

dependency graph of $\langle \mathcal{S}^\sharp / \mathcal{W}^\sharp \cup \mathcal{W}, \mathcal{T}^\sharp \rangle$ is graphs such that

1. nodes are dependency pairs $\mathcal{S}^\sharp \cup \mathcal{W}^\sharp$
2. there is edge

$$(s^\sharp \to \mathsf{c}_n(t_1^\sharp, \ldots, t_n^\sharp)) \;\to\; (u^\sharp \to \mathsf{c}_m(v_1^\sharp, \ldots, v_m^\sharp))$$

if there exists substitutions σ, τ such that $t_i^\sharp \sigma \to^*_{\mathcal{S} \cup \mathcal{W}} u^\sharp \tau$

Dependency Graph

dependency graph of $\langle \mathcal{S}^\sharp / \mathcal{W}^\sharp \cup \mathcal{W}, \mathcal{T}^\sharp \rangle$ is graphs such that

1. nodes are dependency pairs $\mathcal{S}^\sharp \cup \mathcal{W}^\sharp$
2. there is edge

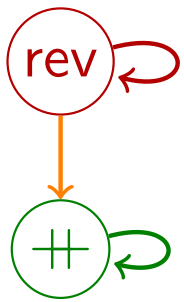
$$(s^\sharp \to \mathsf{c}_n(t_1^\sharp, \ldots, t_n^\sharp)) \;\to\; (u^\sharp \to \mathsf{c}_m(v_1^\sharp, \ldots, v_m^\sharp))$$

if there exists substitutions σ, τ such that $t_i^\sharp \sigma \to^*_{\mathcal{S} \cup \mathcal{W}} u^\sharp \tau$

Dependency Graph Decomposition

example

$\mathsf{rev}^\sharp([\mathbf{1},\mathbf{2},\mathbf{3}])$



$$\mathcal{S}^\sharp_\uparrow := \{\mathsf{rev}^\sharp(x:xs) \to \mathsf{c}_2(\mathsf{rev}(xs) +\!\!+^\sharp [x], \mathsf{rev}^\sharp(xs))\}$$

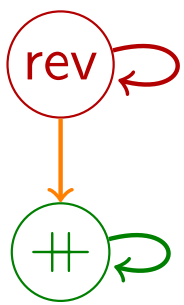
$$\mathcal{S}^\sharp_\downarrow := \{(x:xs) +\!\!+^\sharp ys \to xs +\!\!+^\sharp ys\}$$

$$\mathsf{sep}(\mathcal{S}^\sharp_\uparrow) := \left\{ \begin{array}{l} \mathsf{rev}^\sharp(x:xs) \to \mathsf{rev}^\sharp(xs) \\ \mathsf{rev}^\sharp(x:xs) \to \mathsf{rev}(xs) +\!\!+^\sharp [x]) \end{array} \right\}$$

Dependency Graph Decomposition

example

$\mathsf{rev}^\sharp([\mathbf{1},\mathbf{2},\mathbf{3}])$



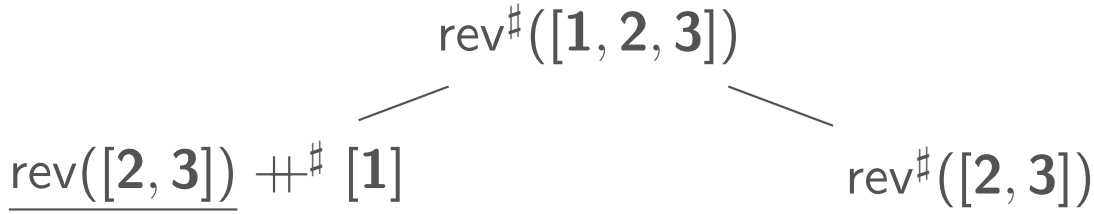
$$\mathcal{S}^\sharp_\uparrow := \{\mathsf{rev}^\sharp(x:xs) \to \mathsf{c}_2(\mathsf{rev}(xs) +\!\!+^\sharp [x], \mathsf{rev}^\sharp(xs))\}$$

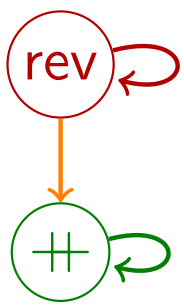
$$\mathcal{S}^\sharp_\downarrow := \{(x:xs) +\!\!+^\sharp ys \to xs +\!\!+^\sharp ys\}$$

$$\mathsf{sep}(\mathcal{S}^\sharp_\uparrow) := \left\{ \begin{array}{l} \mathsf{rev}^\sharp(x:xs) \to \mathsf{rev}^\sharp(xs) \\ \mathsf{rev}^\sharp(x:xs) \to \mathsf{rev}(xs) +\!\!+^\sharp [x]) \end{array} \right\}$$

Dependency Graph Decomposition

example





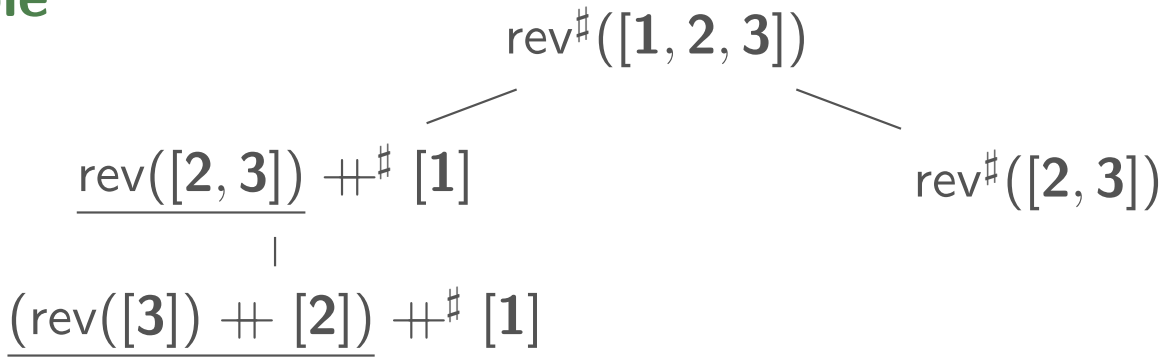
$$\mathcal{S}^\sharp_\uparrow := \left\{ \mathsf{rev}^\sharp(x : xs) \to \mathsf{c}_2(\mathsf{rev}(xs)\ +\!\!+^\sharp\ [x], \mathsf{rev}^\sharp(xs)) \right\}$$

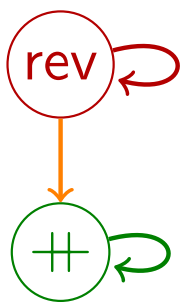
$$\mathcal{S}^\sharp_\downarrow := \left\{ (x : xs)\ +\!\!+^\sharp\ ys \to xs\ +\!\!+^\sharp\ ys \right\}$$

$$\mathsf{sep}(\mathcal{S}^\sharp_\uparrow) := \left\{ \begin{array}{l} \mathsf{rev}^\sharp(x : xs) \to \mathsf{rev}^\sharp(xs) \\ \mathsf{rev}^\sharp(x : xs) \to \mathsf{rev}(xs)\ +\!\!+^\sharp\ [x]) \end{array} \right\}$$

Dependency Graph Decomposition

example





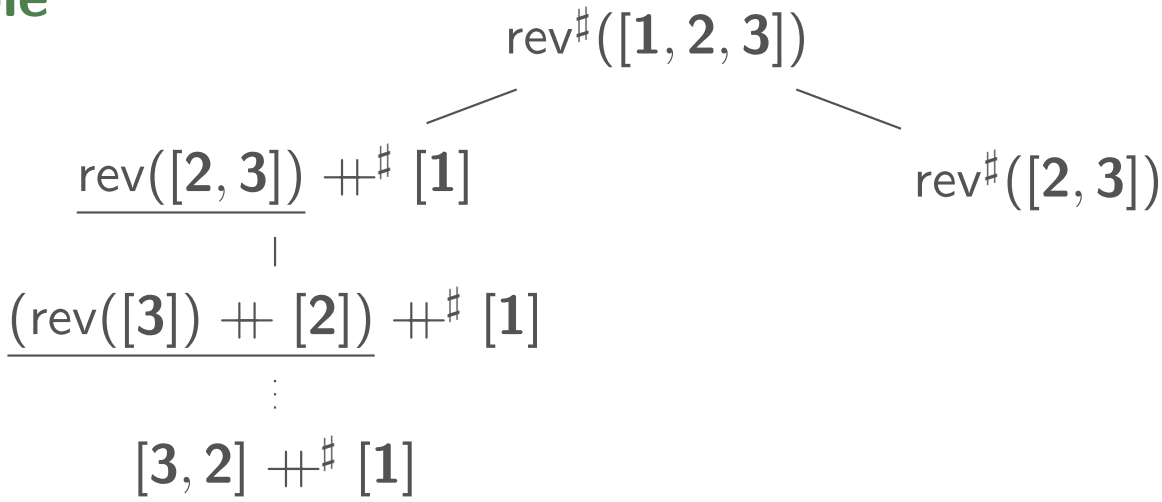
$$\mathcal{S}^\sharp_\uparrow := \{\mathsf{rev}^\sharp(x:xs) \to \mathsf{c}_2(\mathsf{rev}(xs) +\!\!+^\sharp [x], \mathsf{rev}^\sharp(xs))\}$$

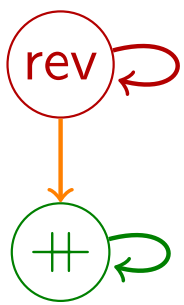
$$\mathcal{S}^\sharp_\downarrow := \{(x:xs) +\!\!+^\sharp ys \to xs +\!\!+^\sharp ys\}$$

$$\mathsf{sep}(\mathcal{S}^\sharp_\uparrow) := \left\{ \begin{array}{l} \mathsf{rev}^\sharp(x:xs) \to \mathsf{rev}^\sharp(xs) \\ \mathsf{rev}^\sharp(x:xs) \to \mathsf{rev}(xs) +\!\!+^\sharp [x]) \end{array} \right\}$$

Dependency Graph Decomposition

example





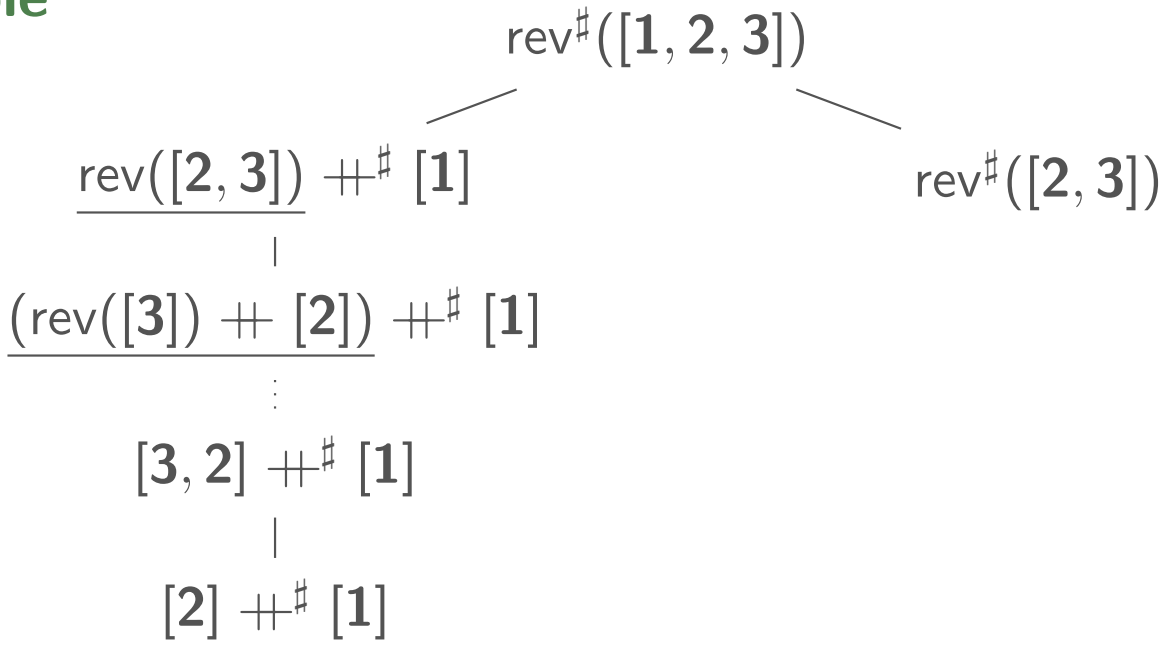
$$\mathcal{S}^\sharp_\uparrow := \{\mathsf{rev}^\sharp(x:xs) \to \mathsf{c}_2(\mathsf{rev}(xs) +\!\!+^\sharp [x], \mathsf{rev}^\sharp(xs))\}$$

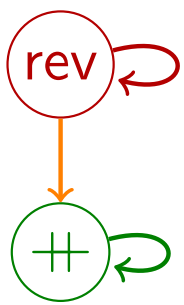
$$\mathcal{S}^\sharp_\downarrow := \{(x:xs) +\!\!+^\sharp ys \to xs +\!\!+^\sharp ys\}$$

$$\mathsf{sep}(\mathcal{S}^\sharp_\uparrow) := \left\{ \begin{array}{l} \mathsf{rev}^\sharp(x:xs) \to \mathsf{rev}^\sharp(xs) \\ \mathsf{rev}^\sharp(x:xs) \to \mathsf{rev}(xs) +\!\!+^\sharp [x]) \end{array} \right\}$$

Dependency Graph Decomposition

example





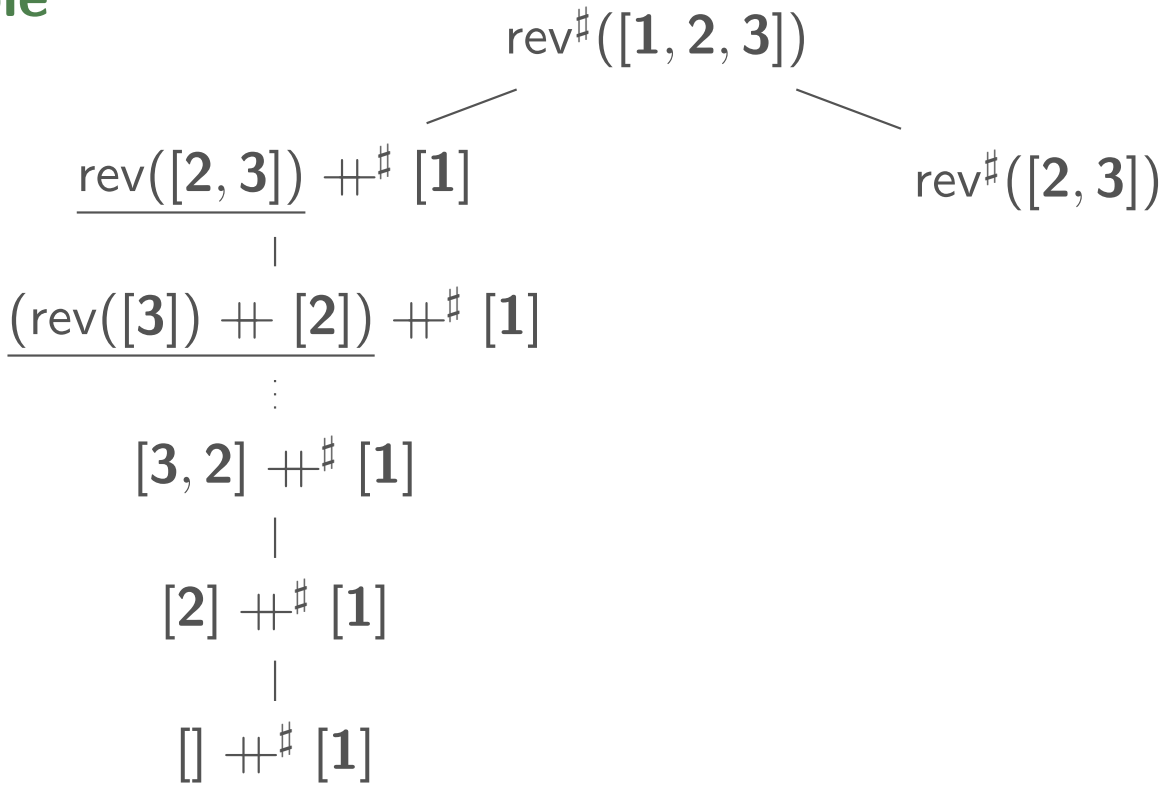
$$\mathcal{S}^\sharp_\uparrow := \{\mathsf{rev}^\sharp(x:xs) \to \mathsf{c}_2(\mathsf{rev}(xs) +\!\!+^\sharp [x], \mathsf{rev}^\sharp(xs))\}$$

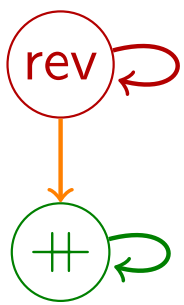
$$\mathcal{S}^\sharp_\downarrow := \{(x:xs) +\!\!+^\sharp ys \to xs +\!\!+^\sharp ys\}$$

$$\mathsf{sep}(\mathcal{S}^\sharp_\uparrow) := \left\{ \begin{array}{l} \mathsf{rev}^\sharp(x:xs) \to \mathsf{rev}^\sharp(xs) \\ \mathsf{rev}^\sharp(x:xs) \to \mathsf{rev}(xs) +\!\!+^\sharp [x]) \end{array} \right\}$$

Dependency Graph Decomposition

example





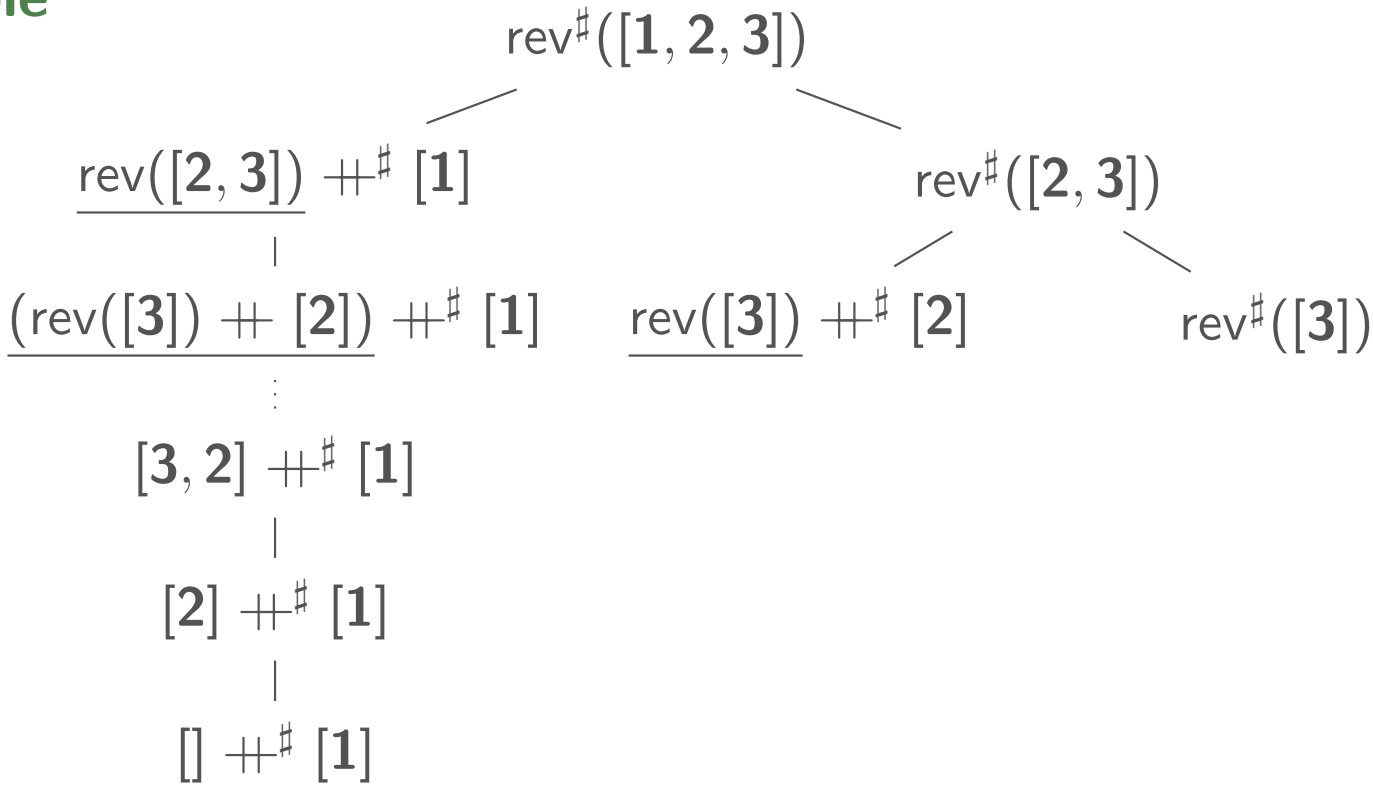
$$\mathcal{S}^\sharp_\uparrow := \{\mathsf{rev}^\sharp(x:xs) \to \mathsf{c}_2(\mathsf{rev}(xs) +\!\!+^\sharp [x], \mathsf{rev}^\sharp(xs))\}$$

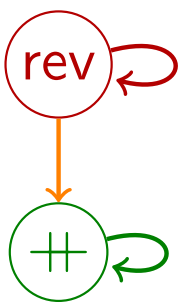
$$\mathcal{S}^\sharp_\downarrow := \{(x:xs) +\!\!+^\sharp ys \to xs +\!\!+^\sharp ys\}$$

$$\mathsf{sep}(\mathcal{S}^\sharp_\uparrow) := \left\{ \begin{array}{l} \mathsf{rev}^\sharp(x:xs) \to \mathsf{rev}^\sharp(xs) \\ \mathsf{rev}^\sharp(x:xs) \to \mathsf{rev}(xs) +\!\!+^\sharp [x]) \end{array} \right\}$$

Dependency Graph Decomposition

example





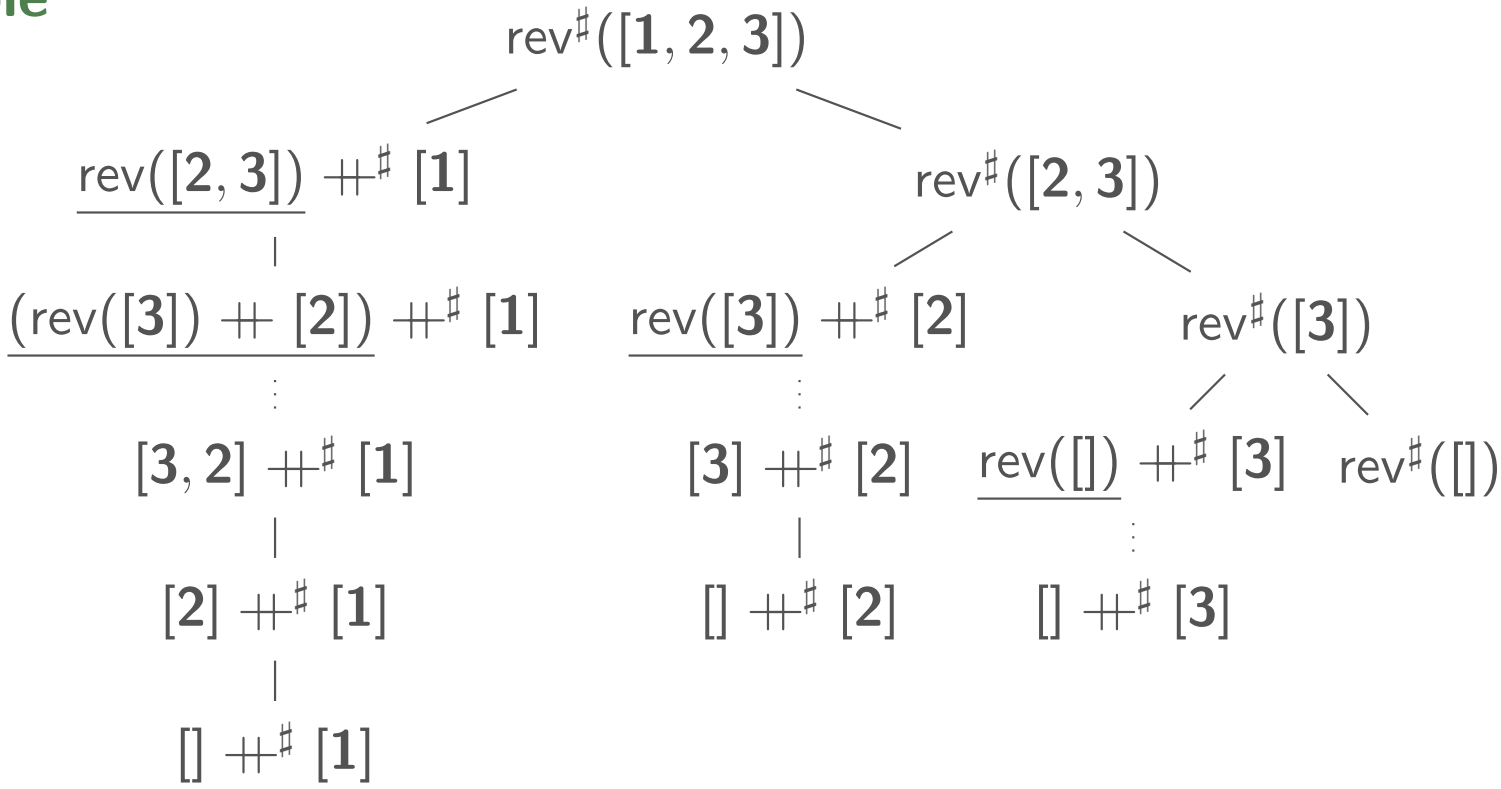
$$\mathcal{S}^\sharp_\uparrow := \{\mathsf{rev}^\sharp(x:xs) \to \mathsf{c}_2(\mathsf{rev}(xs) +\!\!+^\sharp [x], \mathsf{rev}^\sharp(xs))\}$$

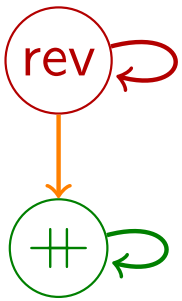
$$\mathcal{S}^\sharp_\downarrow := \{(x:xs) +\!\!+^\sharp ys \to xs +\!\!+^\sharp ys\}$$

$$\mathsf{sep}(\mathcal{S}^\sharp_\uparrow) := \left\{ \begin{array}{l} \mathsf{rev}^\sharp(x:xs) \to \mathsf{rev}^\sharp(xs) \\ \mathsf{rev}^\sharp(x:xs) \to \mathsf{rev}(xs) +\!\!+^\sharp [x]) \end{array} \right\}$$

Dependency Graph Decomposition

example





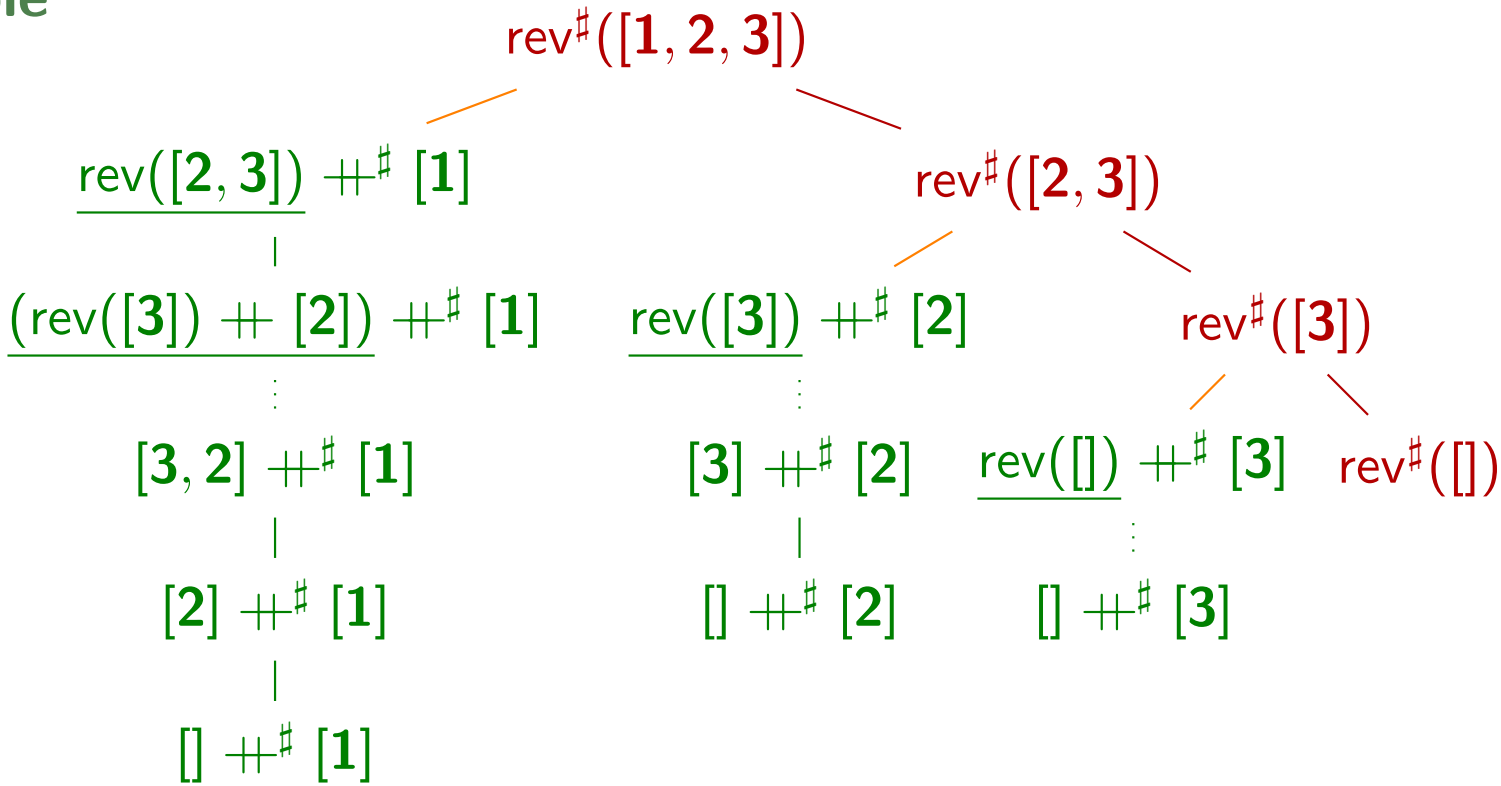
$$\mathcal{S}^\sharp_\uparrow := \{\mathsf{rev}^\sharp(x:xs) \to \mathsf{c}_2(\mathsf{rev}(xs) +\!\!+^\sharp [x], \mathsf{rev}^\sharp(xs))\}$$

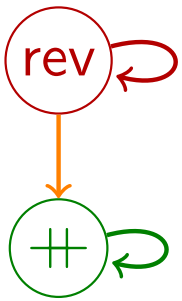
$$\mathcal{S}^\sharp_\downarrow := \{(x:xs) +\!\!+^\sharp ys \to xs +\!\!+^\sharp ys\}$$

$$\mathsf{sep}(\mathcal{S}^\sharp_\uparrow) := \left\{ \begin{array}{l} \mathsf{rev}^\sharp(x:xs) \to \mathsf{rev}^\sharp(xs) \\ \mathsf{rev}^\sharp(x:xs) \to \mathsf{rev}(xs) +\!\!+^\sharp [x]) \end{array} \right\}$$

Dependency Graph Decomposition

example





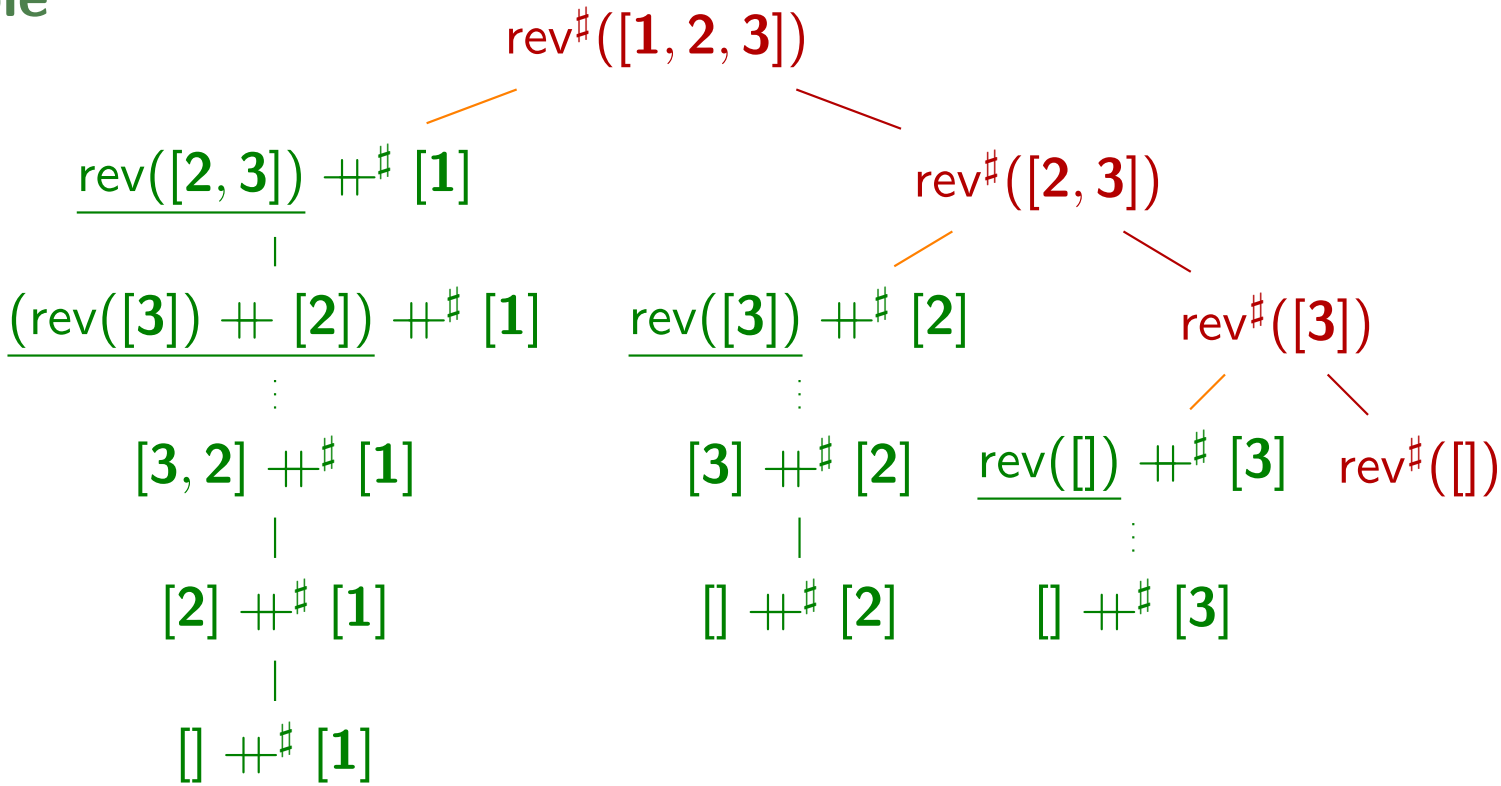
$$\mathcal{S}^\sharp_\uparrow := \{\mathsf{rev}^\sharp(x:xs) \to \mathsf{c}_2(\mathsf{rev}(xs) +\!\!+^\sharp [x], \mathsf{rev}^\sharp(xs))\}$$

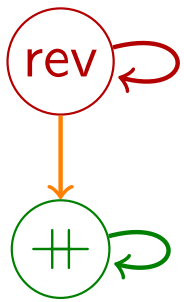
$$\mathcal{S}^\sharp_\downarrow := \{(x:xs) +\!\!+^\sharp ys \to xs +\!\!+^\sharp ys\}$$

$$\mathsf{sep}(\mathcal{S}^\sharp_\uparrow) := \left\{ \begin{array}{l} \mathsf{rev}^\sharp(x:xs) \to \mathsf{rev}^\sharp(xs) \\ \mathsf{rev}^\sharp(x:xs) \to \mathsf{rev}(xs) +\!\!+^\sharp [x]) \end{array} \right\}$$

Dependency Graph Decomposition

example





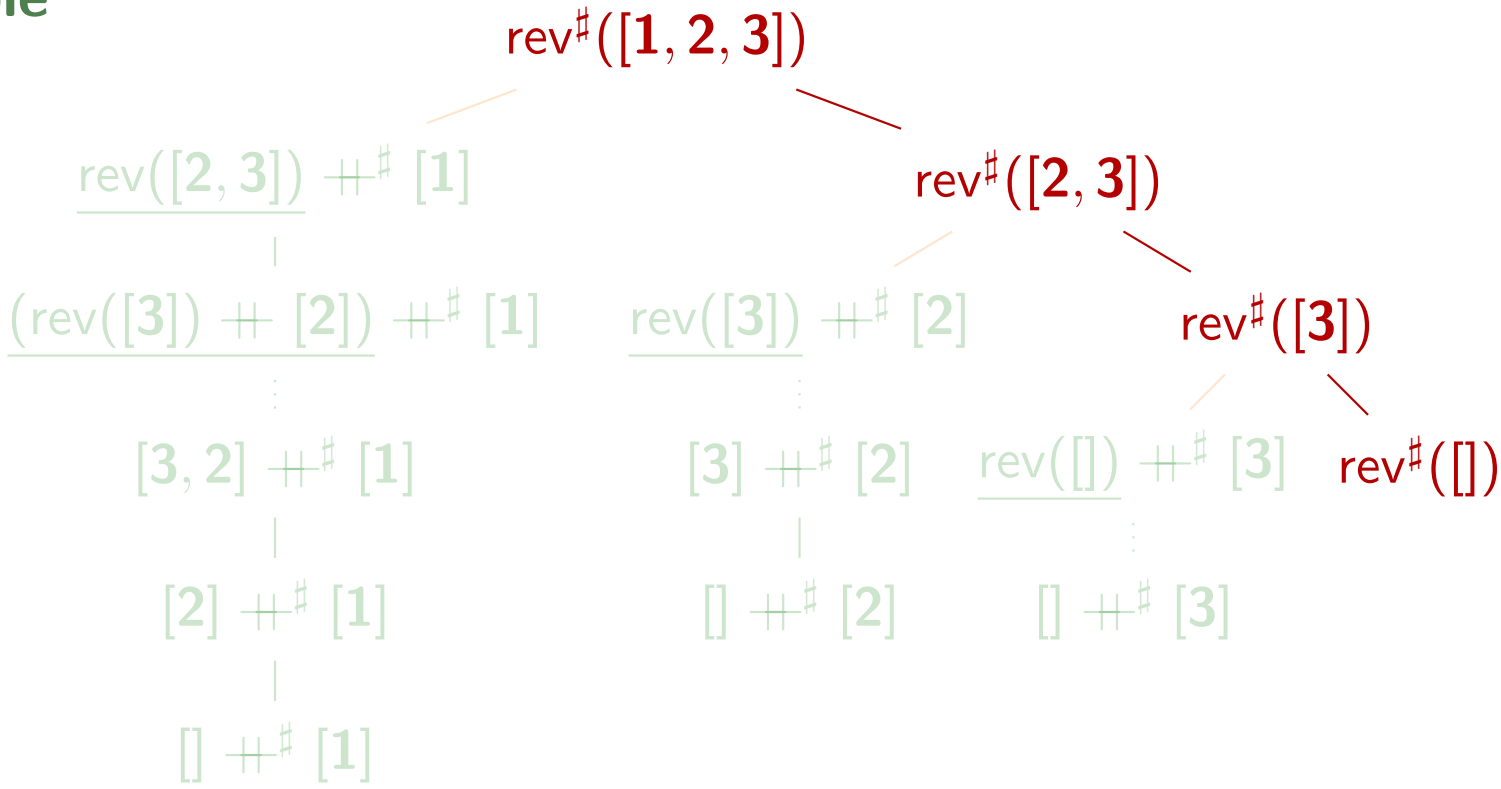
$$\mathcal{S}^\sharp_\uparrow := \{\mathsf{rev}^\sharp(x:xs) \to \mathsf{c}_2(\mathsf{rev}(xs) +\!\!+^\sharp [x], \mathsf{rev}^\sharp(xs))\}$$

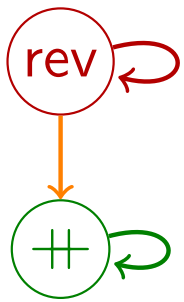
$$\mathcal{S}^\sharp_\downarrow := \{(x:xs) +\!\!+^\sharp ys \to xs +\!\!+^\sharp ys\}$$

$$\mathsf{sep}(\mathcal{S}^\sharp_\uparrow) := \left\{ \begin{array}{l} \mathsf{rev}^\sharp(x:xs) \to \mathsf{rev}^\sharp(xs) \\ \mathsf{rev}^\sharp(x:xs) \to \mathsf{rev}(xs) +\!\!+^\sharp [x]) \end{array} \right\}$$

Dependency Graph Decomposition

example





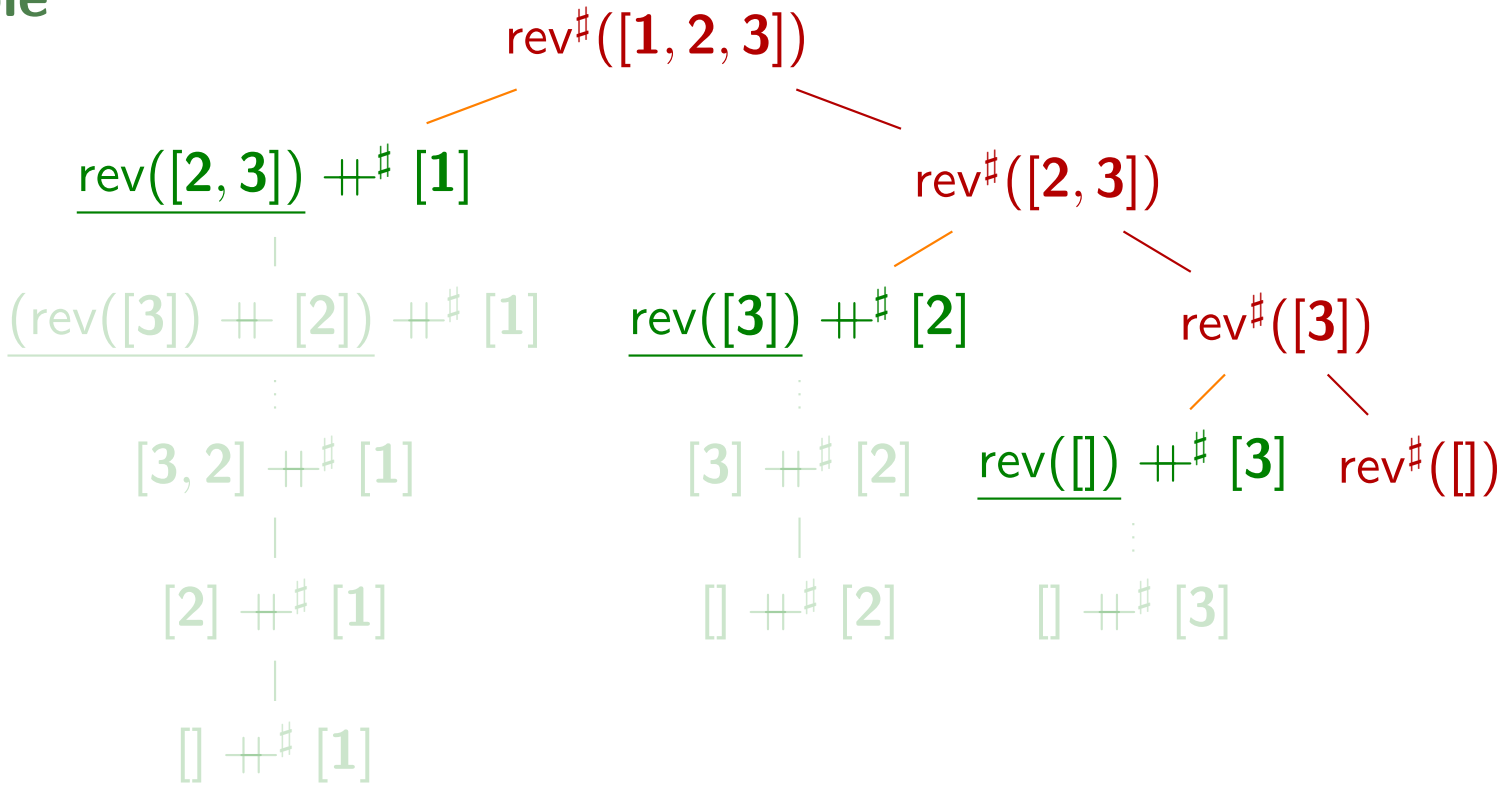
$$\mathcal{S}^\sharp_\uparrow := \{\mathsf{rev}^\sharp(x:xs) \to \mathsf{c}_2(\mathsf{rev}(xs) +\!\!+^\sharp [x], \mathsf{rev}^\sharp(xs))\}$$

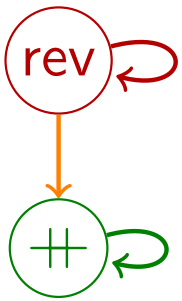
$$\mathcal{S}^\sharp_\downarrow := \{(x:xs) +\!\!+^\sharp ys \to xs +\!\!+^\sharp ys\}$$

$$\mathsf{sep}(\mathcal{S}^\sharp_\uparrow) := \left\{ \begin{array}{l} \mathsf{rev}^\sharp(x:xs) \to \mathsf{rev}^\sharp(xs) \\ \mathsf{rev}^\sharp(x:xs) \to \mathsf{rev}(xs) +\!\!+^\sharp [x]) \end{array} \right\}$$

Dependency Graph Decomposition

example





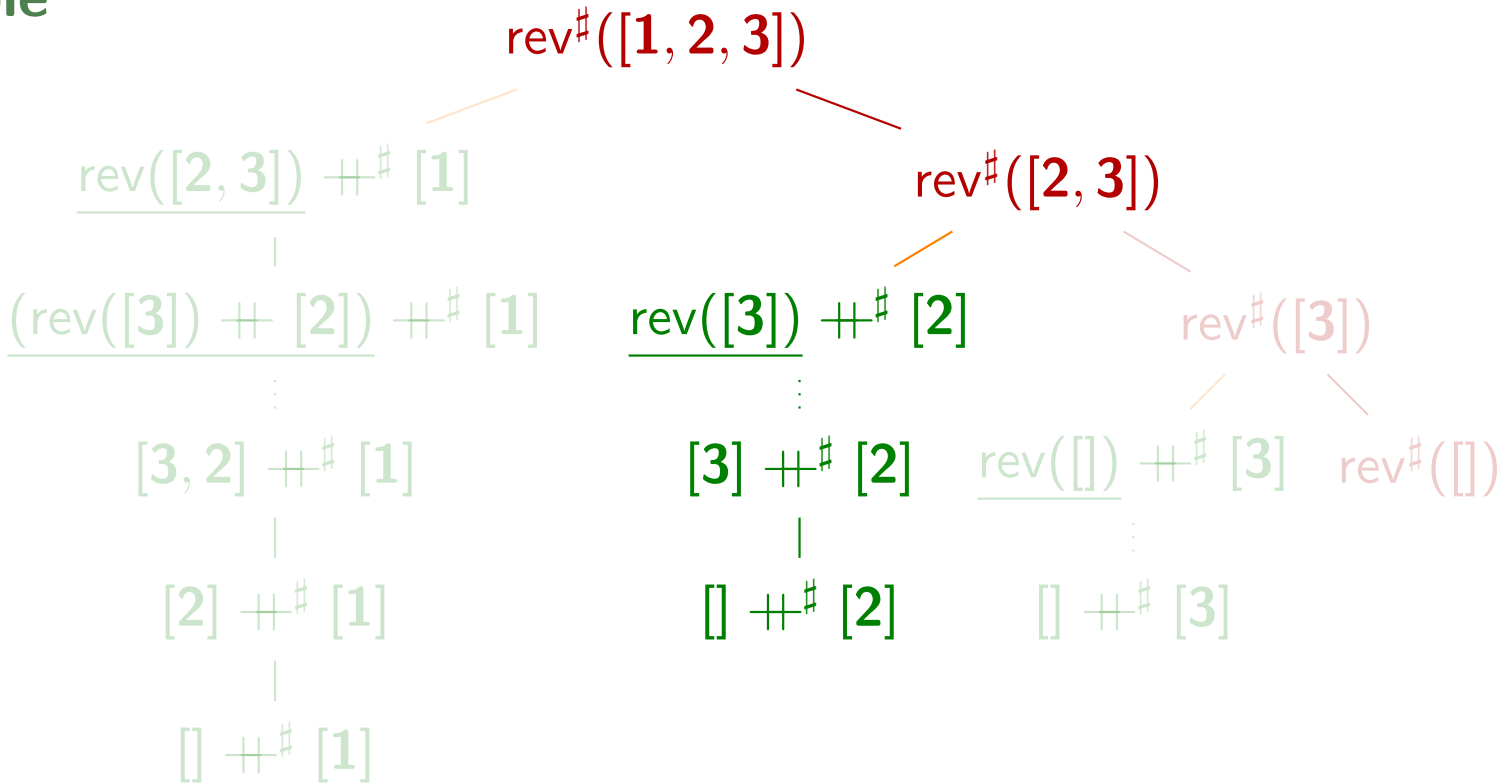
$$\mathcal{S}^\sharp_\uparrow := \{\mathsf{rev}^\sharp(x:xs) \to \mathsf{c}_2(\mathsf{rev}(xs) +\!\!+^\sharp [x], \mathsf{rev}^\sharp(xs))\}$$

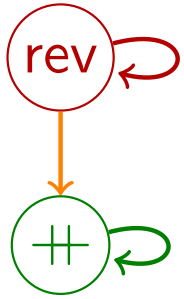
$$\mathcal{S}^\sharp_\downarrow := \{(x:xs) +\!\!+^\sharp ys \to xs +\!\!+^\sharp ys\}$$

$$\mathsf{sep}(\mathcal{S}^\sharp_\uparrow) := \left\{ \begin{array}{l} \mathsf{rev}^\sharp(x:xs) \to \mathsf{rev}^\sharp(xs) \\ \mathsf{rev}^\sharp(x:xs) \to \mathsf{rev}(xs) +\!\!+^\sharp [x]) \end{array} \right\}$$

Dependency Graph Decomposition

example





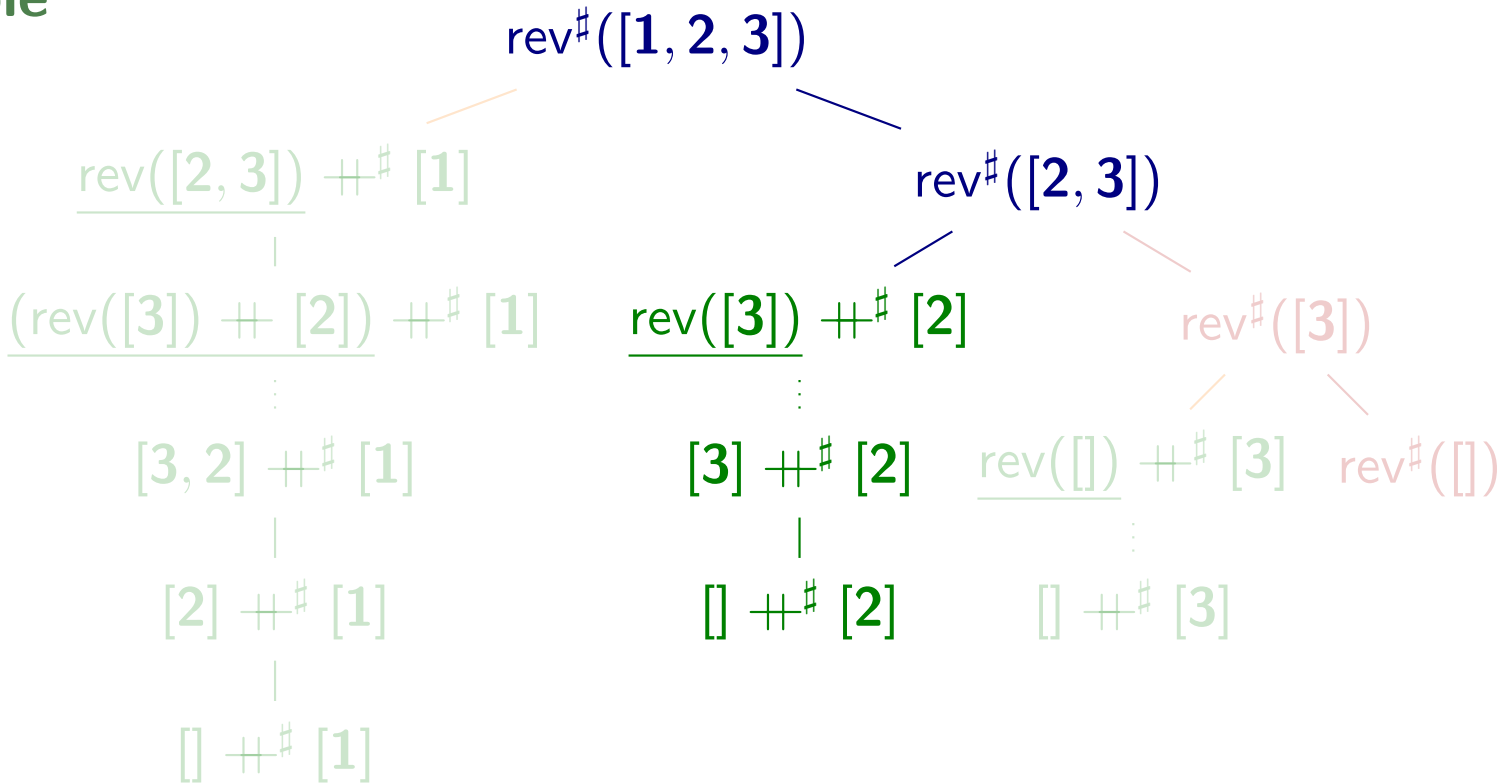
$$\mathcal{S}^\sharp_\uparrow := \{\mathsf{rev}^\sharp(x:xs) \to \mathsf{c}_2(\mathsf{rev}(xs) +\!\!+^\sharp [x], \mathsf{rev}^\sharp(xs))\}$$

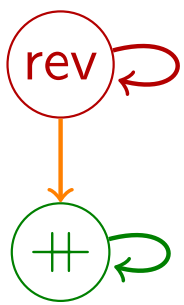
$$\mathcal{S}^\sharp_\downarrow := \{(x:xs) +\!\!+^\sharp ys \to xs +\!\!+^\sharp ys\}$$

$$\mathsf{sep}(\mathcal{S}^\sharp_\uparrow) := \left\{ \begin{array}{l} \mathsf{rev}^\sharp(x:xs) \to \mathsf{rev}^\sharp(xs) \\ \mathsf{rev}^\sharp(x:xs) \to \mathsf{rev}(xs) +\!\!+^\sharp [x] \end{array} \right\}$$

Dependency Graph Decomposition

example





$$\mathcal{S}^\sharp_\uparrow := \{\mathsf{rev}^\sharp(x:xs) \to \mathsf{c}_2(\mathsf{rev}(xs) +\!\!+^\sharp [x], \mathsf{rev}^\sharp(xs))\}$$

$$\mathcal{S}^\sharp_\downarrow := \{(x:xs) +\!\!+^\sharp ys \to xs +\!\!+^\sharp ys\}$$

$$\mathsf{sep}(\mathcal{S}^\sharp_\uparrow) := \left\{ \begin{array}{l} \mathsf{rev}^\sharp(x:xs) \to \mathsf{rev}^\sharp(xs) \\ \mathsf{rev}^\sharp(x:xs) \to \mathsf{rev}(xs) +\!\!+^\sharp [x]) \end{array} \right\}$$

Dependency Graph Decomposition

processor

consider $\mathcal{P} = \langle \mathcal{S}^\sharp / \mathcal{W}^\sharp \cup \mathcal{W}, \mathcal{T}^\sharp \rangle$ with dependency graph $\mathcal{G}$.

- partition $\mathcal{S}^\sharp = \mathcal{S}^\sharp_\downarrow \cup \mathcal{S}^\sharp_\uparrow$ and $\mathcal{W}^\sharp = \mathcal{W}^\sharp_\downarrow \cup \mathcal{W}^\sharp_\uparrow$
- suppose $\mathcal{S}^\sharp_\downarrow \cup \mathcal{W}^\sharp_\downarrow$ closed under successors in $\mathcal{G}$
- define $\mathsf{sep}(l^\sharp \to \mathsf{c}(r_1^\sharp, \ldots, r_m^\sharp)) := \{l^\sharp \to r_i^\sharp \mid i = 1, \ldots, m\}$
- predecessors of $\mathcal{S}^\sharp_\downarrow \cup \mathcal{W}^\sharp_\downarrow$ in $\mathcal{G}$ contained in $\mathcal{S}^\sharp_\uparrow$

$$\frac{\vdash \langle \mathcal{S}^\sharp_\uparrow \cup \mathcal{S} / \mathcal{W}^\sharp_\uparrow \cup \mathcal{W}, \mathcal{T} \rangle : f \quad \vdash \langle \mathcal{S}^\sharp_\downarrow \cup \mathcal{S} / \mathcal{W}^\sharp_\downarrow \cup \mathsf{sep}(\mathcal{S}^\sharp_\uparrow \cup \mathcal{W}^\sharp_\uparrow) \cup \mathcal{W}, \mathcal{T} \rangle : g}{\vdash \langle \mathcal{S}^\sharp / \mathcal{W}^\sharp \cup \mathcal{W}, \mathcal{T} \rangle : \lambda n. f(n) * g(n)}$$

Dependency Graph Decomposition

processor

consider $\mathcal{P} = \langle \mathcal{S}^\sharp / \mathcal{W}^\sharp \cup \mathcal{W}, \mathcal{T}^\sharp \rangle$ with dependency graph $\mathcal{G}$.

- partition $\mathcal{S}^\sharp = \mathcal{S}^\sharp_\downarrow \cup \mathcal{S}^\sharp_\uparrow$ and $\mathcal{W}^\sharp = \mathcal{W}^\sharp_\downarrow \cup \mathcal{W}^\sharp_\uparrow$
- suppose $\mathcal{S}^\sharp_\downarrow \cup \mathcal{W}^\sharp_\downarrow$ closed under successors in $\mathcal{G}$
- define $\mathsf{sep}(l^\sharp \to \mathsf{c}(r_1^\sharp, \ldots, r_m^\sharp)) := \{l^\sharp \to r_i^\sharp \mid i = 1, \ldots, m\}$
- predecessors of $\mathcal{S}^\sharp_\downarrow \cup \mathcal{W}^\sharp_\downarrow$ in $\mathcal{G}$ contained in $\mathcal{S}^\sharp_\uparrow$

$$\frac{\vdash \langle \mathcal{S}^\sharp_\uparrow \cup \mathcal{S} / \mathcal{W}^\sharp_\uparrow \cup \mathcal{W}, \mathcal{T} \rangle : f \quad \vdash \langle \mathcal{S}^\sharp_\downarrow \cup \mathcal{S} / \mathcal{W}^\sharp_\downarrow \cup \mathsf{sep}(\mathcal{S}^\sharp_\uparrow \cup \mathcal{W}^\sharp_\uparrow) \cup \mathcal{W}, \mathcal{T} \rangle : g}{\vdash \langle \mathcal{S}^\sharp / \mathcal{W}^\sharp \cup \mathcal{W}, \mathcal{T} \rangle : \lambda n. f(n) * g(n)}$$

Dependency Graph Decomposition

example

$$\dfrac{\vdash \langle \{\mathsf{a}, \mathsf{b}\}/\mathcal{R}_{\mathsf{rev}}, \mathtt{basics}^\sharp \rangle : f}{\vdots}$$
$$\vdash \langle \mathcal{R}_{\mathsf{rev}}/\varnothing, \mathtt{basics} \rangle : f$$

a : $\quad \mathsf{rev}^\sharp(x : xs) \to \mathsf{c}_2(\mathsf{rev}(xs) +\!\!+^\sharp [x], \mathsf{rev}^\sharp(xs))$

b : $(x : xs) +\!\!+^\sharp ys \to xs +\!\!+^\sharp ys$

Dependency Graph Decomposition

example

$$\dfrac{\dfrac{\vdash \langle \{\mathsf{a}\}/\mathcal{R}_{\mathsf{rev}}, \mathtt{basics}^\sharp \rangle : f_1 \qquad \vdash \langle \{\mathsf{b}\}/\{\mathsf{a}_1, \mathsf{a}_2\} \cup \mathcal{R}_{\mathsf{rev}}, \mathtt{basics}^\sharp \rangle : f_2}{\vdash \langle \{\mathsf{a}, \mathsf{b}\}/\mathcal{R}_{\mathsf{rev}}, \mathtt{basics}^\sharp \rangle : f \;\; = f_1 \cdot f_2}}{\vdots}}{\vdash \langle \mathcal{R}_{\mathsf{rev}}/\varnothing, \mathtt{basics} \rangle : f}$$

$$\begin{aligned}
\mathsf{a} &: \quad \mathsf{rev}^\sharp(x : xs) \to \mathsf{c}_2(\mathsf{rev}(xs) +\!\!+^\sharp [x], \mathsf{rev}^\sharp(xs)) \\
\mathsf{b} &: (x : xs) +\!\!+^\sharp ys \to xs +\!\!+^\sharp ys \\
\mathsf{a}_1 &: \quad \mathsf{rev}^\sharp(x : xs) \to \mathsf{rev}^\sharp(xs) \\
\mathsf{a}_2 &: \quad \mathsf{rev}^\sharp(x : xs) \to \mathsf{rev}(xs) +\!\!+^\sharp [x]
\end{aligned}$$

Dependency Graph Decomposition

example

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdash \langle \{\mathsf{a}'\}/\mathcal{R}_{\mathsf{rev}}, \mathtt{basics}^\sharp\rangle : f_1}{\vdash \langle \{\mathsf{a}\}/\mathcal{R}_{\mathsf{rev}}, \mathtt{basics}^\sharp\rangle : f_1} \quad \vdash \langle \{\mathsf{b}\}/\{\mathsf{a}_1, \mathsf{a}_2\} \cup \mathcal{R}_{\mathsf{rev}}, \mathtt{basics}^\sharp\rangle : f_2}{\vdash \langle \{\mathsf{a}, \mathsf{b}\}/\mathcal{R}_{\mathsf{rev}}, \mathtt{basics}^\sharp\rangle : f \;\; = f_1 \cdot f_2}}{\vdots}}{\vdash \langle \mathcal{R}_{\mathsf{rev}}/\varnothing, \mathtt{basics}\rangle : f}$$

a': $\quad \mathsf{rev}^\sharp(x : xs) \to \mathsf{rev}^\sharp(xs)$

b : $(x : xs) +\!\!+^\sharp ys \to xs +\!\!+^\sharp ys$

a_1 : $\quad \mathsf{rev}^\sharp(x : xs) \to \mathsf{rev}^\sharp(xs)$

a_2 : $\quad \mathsf{rev}^\sharp(x : xs) \to \mathsf{rev}(xs) +\!\!+^\sharp [x]$

Dependency Graph Decomposition

example

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdash \langle \{\mathsf{a}'\}/\varnothing, \mathtt{basics}^\sharp \rangle : f_1}{\vdash \langle \{\mathsf{a}'\}/\mathcal{R}_{\mathsf{rev}}, \mathtt{basics}^\sharp \rangle : f_1}}{\vdash \langle \{\mathsf{a}\}/\mathcal{R}_{\mathsf{rev}}, \mathtt{basics}^\sharp \rangle : f_1} \qquad \vdash \langle \{\mathsf{b}\}/\{\mathsf{a}_1, \mathsf{a}_2\} \cup \mathcal{R}_{\mathsf{rev}}, \mathtt{basics}^\sharp \rangle : f_2}{\vdash \langle \{\mathsf{a}, \mathsf{b}\}/\mathcal{R}_{\mathsf{rev}}, \mathtt{basics}^\sharp \rangle : f \;\; = f_1 \cdot f_2}}{\vdots}}{\vdash \langle \mathcal{R}_{\mathsf{rev}}/\varnothing, \mathtt{basics} \rangle : f}$$

a': $\quad \mathsf{rev}^\sharp(x : xs) \to \mathsf{rev}^\sharp(xs)$

b : $(x : xs) +\!\!+^\sharp ys \to xs +\!\!+^\sharp ys$

a_1 : $\quad \mathsf{rev}^\sharp(x : xs) \to \mathsf{rev}^\sharp(xs)$

a_2 : $\quad \mathsf{rev}^\sharp(x : xs) \to \mathsf{rev}(xs) +\!\!+^\sharp [x]$

Dependency Graph Decomposition

example

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\{\mathsf{a}'\} \subseteq >_{\mathcal{A}}}{\vdash \langle \{\mathsf{a}'\}/\varnothing, \mathtt{basics}^\sharp \rangle : f_1}}{\vdash \langle \{\mathsf{a}'\}/\mathcal{R}_{\mathsf{rev}}, \mathtt{basics}^\sharp \rangle : f_1}}{\vdash \langle \{\mathsf{a}\}/\mathcal{R}_{\mathsf{rev}}, \mathtt{basics}^\sharp \rangle : f_1} \qquad \vdash \langle \{\mathsf{b}\}/\{\mathsf{a}_1, \mathsf{a}_2\} \cup \mathcal{R}_{\mathsf{rev}}, \mathtt{basics}^\sharp \rangle : f_2}{\vdash \langle \{\mathsf{a}, \mathsf{b}\}/\mathcal{R}_{\mathsf{rev}}, \mathtt{basics}^\sharp \rangle : f \;\; = f_1 \cdot f_2}}{\vdots}}{\vdash \langle \mathcal{R}_{\mathsf{rev}}/\varnothing, \mathtt{basics} \rangle : f}$$

$$\begin{array}{lll}
\mathsf{a}': & \mathsf{rev}^\sharp(x : xs) \to \mathsf{rev}^\sharp(xs) & \mathsf{rev}^\sharp_{\mathcal{A}}(xs) = xs \\
\mathsf{b}: & (x : xs) +\!\!+^\sharp ys \to xs +\!\!+^\sharp ys & \\
\mathsf{a}_1: & \mathsf{rev}^\sharp(x : xs) \to \mathsf{rev}^\sharp(xs) & []_{\mathcal{B}} = 0 \\
\mathsf{a}_2: & \mathsf{rev}^\sharp(x : xs) \to \mathsf{rev}(xs) +\!\!+^\sharp [x] & x :_{\mathcal{A}} xs = xs + 1
\end{array}$$

Dependency Graph Decomposition

example

$$\dfrac{\dfrac{\dfrac{\dfrac{\{\mathsf{a}'\} \subseteq >_{\mathcal{A}}}{\vdash \langle \{\mathsf{a}'\}/\varnothing, \mathtt{basics}^\sharp \rangle : n}}{\vdash \langle \{\mathsf{a}'\}/\mathcal{R}_{\mathsf{rev}}, \mathtt{basics}^\sharp \rangle : n}}{\vdash \langle \{\mathsf{a}\}/\mathcal{R}_{\mathsf{rev}}, \mathtt{basics}^\sharp \rangle : n} \qquad \vdash \langle \{\mathsf{b}\}/\{\mathsf{a}_1, \mathsf{a}_2\} \cup \mathcal{R}_{\mathsf{rev}}, \mathtt{basics}^\sharp \rangle : f_2}{\dfrac{\vdash \langle \{\mathsf{a}, \mathsf{b}\}/\mathcal{R}_{\mathsf{rev}}, \mathtt{basics}^\sharp \rangle : f \quad = n \cdot f_2}{\vdots}}{\vdash \langle \mathcal{R}_{\mathsf{rev}}/\varnothing, \mathtt{basics} \rangle : f}$$

a': $\mathsf{rev}^\sharp(x : xs) \to \mathsf{rev}^\sharp(xs)$ $\qquad \mathsf{rev}^\sharp_{\mathcal{A}}(xs) = xs$

b : $(x : xs) +\!\!+^\sharp ys \to xs +\!\!+^\sharp ys$

a_1 : $\mathsf{rev}^\sharp(x : xs) \to \mathsf{rev}^\sharp(xs)$ $\qquad []_{\mathcal{B}} = 0$

a_2 : $\mathsf{rev}^\sharp(x : xs) \to \mathsf{rev}(xs) +\!\!+^\sharp [x]$ $\quad x :_{\mathcal{A}} xs = xs + 1$

Dependency Graph Decomposition

example

$$\dfrac{\dfrac{\dfrac{\dfrac{\{\mathsf{a}'\} \subseteq >_{\mathcal{A}}}{\vdash \langle \{\mathsf{a}'\}/\varnothing, \mathtt{basics}^\sharp \rangle : n}}{\vdash \langle \{\mathsf{a}'\}/\mathcal{R}_{\mathsf{rev}}, \mathtt{basics}^\sharp \rangle : n}}{\vdash \langle \{\mathsf{a}\}/\mathcal{R}_{\mathsf{rev}}, \mathtt{basics}^\sharp \rangle : n} \quad \dfrac{\{\mathsf{b}\} \subseteq >_{\mathcal{B}} \quad \{\mathsf{a}_1, \mathsf{a}_2\} \cup \mathcal{R}_{\mathsf{rev}} \subseteq \geqslant_{\mathcal{B}}}{\vdash \langle \{\mathsf{b}\}/\{\mathsf{a}_1, \mathsf{a}_2\} \cup \mathcal{R}_{\mathsf{rev}}, \mathtt{basics}^\sharp \rangle : f_2}}{\dfrac{\vdash \langle \{\mathsf{a}, \mathsf{b}\}/\mathcal{R}_{\mathsf{rev}}, \mathtt{basics}^\sharp \rangle : f \quad = n \cdot f_2}{\vdots}}{\vdash \langle \mathcal{R}_{\mathsf{rev}}/\varnothing, \mathtt{basics} \rangle : f}$$

$$\mathsf{a}' : \quad \mathsf{rev}^\sharp(x : xs) \to \mathsf{rev}^\sharp(xs) \qquad \mathsf{rev}^\sharp_{\mathcal{B}}(xs) = xs$$

$$\mathsf{b} : (x : xs) +\!\!+^\sharp ys \to xs +\!\!+^\sharp ys \qquad xs +\!\!+^\sharp_{\mathcal{B}} ys = xs$$

$$\mathsf{a}_1 : \quad \mathsf{rev}^\sharp(x : xs) \to \mathsf{rev}^\sharp(xs) \qquad []_{\mathcal{B}} = 0 \qquad xs +\!\!+_{\mathcal{B}} ys = xs + ys$$

$$\mathsf{a}_2 : \quad \mathsf{rev}^\sharp(x : xs) \to \mathsf{rev}(xs) +\!\!+^\sharp [x] \quad x :_{\mathcal{B}} xs = xs + 1 \quad \mathsf{rev}_{\mathcal{B}}(xs) = xs$$

Dependency Graph Decomposition

example

$$\dfrac{\dfrac{\dfrac{\dfrac{\{\mathsf{a}'\} \subseteq >_{\mathcal{A}}}{\vdash \langle \{\mathsf{a}'\}/\varnothing, \mathtt{basics}^\sharp \rangle : n}}{\vdash \langle \{\mathsf{a}'\}/\mathcal{R}_{\mathsf{rev}}, \mathtt{basics}^\sharp \rangle : n}}{\vdash \langle \{\mathsf{a}\}/\mathcal{R}_{\mathsf{rev}}, \mathtt{basics}^\sharp \rangle : n} \quad \dfrac{\{\mathsf{b}\} \subseteq >_{\mathcal{B}} \quad \{\mathsf{a}_1, \mathsf{a}_2\} \cup \mathcal{R}_{\mathsf{rev}} \subseteq \geqslant_{\mathcal{B}}}{\vdash \langle \{\mathsf{b}\}/\{\mathsf{a}_1, \mathsf{a}_2\} \cup \mathcal{R}_{\mathsf{rev}}, \mathtt{basics}^\sharp \rangle : n}}{\dfrac{\vdash \langle \{\mathsf{a}, \mathsf{b}\}/\mathcal{R}_{\mathsf{rev}}, \mathtt{basics}^\sharp \rangle : n^2 = n \cdot n}{\vdots}}{\vdash \langle \mathcal{R}_{\mathsf{rev}}/\varnothing, \mathtt{basics} \rangle : n^2}$$

$$\mathsf{a}' : \quad \mathsf{rev}^\sharp(x : xs) \to \mathsf{rev}^\sharp(xs) \qquad \mathsf{rev}^\sharp_{\mathcal{B}}(xs) = xs$$

$$\mathsf{b} : (x : xs) +\!\!+^\sharp ys \to xs +\!\!+^\sharp ys \qquad xs +\!\!+^\sharp_{\mathcal{B}} ys = xs$$

$$\mathsf{a}_1 : \quad \mathsf{rev}^\sharp(x : xs) \to \mathsf{rev}^\sharp(xs) \qquad []_{\mathcal{B}} = 0 \qquad xs +\!\!+_{\mathcal{B}} ys = xs + ys$$

$$\mathsf{a}_2 : \quad \mathsf{rev}^\sharp(x : xs) \to \mathsf{rev}(xs) +\!\!+^\sharp [x] \quad x :_{\mathcal{B}} xs = xs + 1 \quad \mathsf{rev}_{\mathcal{B}}(xs) = xs$$

Experimental Evaluation

Input	#rules	direct	decompose	DG decompose	secs
appendAll	12	$O(n^2)$	$O(n^2)$	$O(n)$	2.9
bfs	57	?	?	$O(n)$	27.9
bft mmult	59	?	?	$O(n^3)$	55.3
bitonic	78	?	?	$O(n^4)$	143.0
bitvectors	148	?	?	$O(n^2)$	35.5
clevermmult	39	?	?	$O(n^2)$	220.0
duplicates	37	?	$O(n^2)$	$O(n^2)$	3.5
dyade	31	?	?	$O(n^2)$	1.0
eratosthenes	74	?	$O(n^3)$	$O(n^2)$	15.8
flatten	31	?	?	$O(n^2)$	25.0
insertionsort	36	?	$O(n^3)$	$O(n^2)$	6.7
listsort	56	?	?	$O(n^2)$	22.8
lcs	87	?	?	$O(n^2)$	24.5
matrix	74	?	?	$O(n^3)$	72.4
mergesort	35	?	?	$O(n^3)$	29.0
minsort	26	?	$O(n^3)$	$O(n^2)$	2.4
queue	35	?	?	$O(n^5)$	148.4
quicksort	46	?	?	$O(n^2)$	38.4
rationalPotential	14	$O(n)$	$O(n)$	$O(n)$	0.27
splitandsort	70	?	?	$O(n^3)$	61.4
subtrees	8	?	$O(n^2)$	$O(n^2)$	3.6
tuples	33	?	?	?	–

Experimental Evaluation

Input	#rules	direct	decompose	DG decompose	secs
appendAll	12	$O(n^2)$	$O(n^2)$	$O(n)$	2.9
bfs	57	?	?	$O(n)$	27.9
bft mmult	59	?	?	$O(n^3)$	55.3
bitonic	78	?	?	$O(n^4)$	143.0
bitvectors	148	?	?	$O(n^2)$	35.5
clevermmult	39	?	?	$O(n^2)$	220.0
duplicates	37	?	$O(n^2)$	$O(n^2)$	3.5
dyade	31	?	?	$O(n^2)$	1.0
eratosthenes	74	?	$O(n^3)$	$O(n^2)$	15.8
flatten	31	?	?	$O(n^2)$	25.0
insertionsort	36	?	$O(n^3)$	$O(n^2)$	6.7
listsort	56	?	?	$O(n^2)$	22.8
lcs	87	?	?	$O(n^2)$	24.5
matrix	74	?	?	$O(n^3)$	72.4
mergesort	35	?	?	$O(n^3)$	29.0
minsort	26	?	$O(n^3)$	$O(n^2)$	2.4
queue	35	?	?	$O(n^5)$	148.4
quicksort	46	?	?	$O(n^2)$	38.4
rationalPotential	14	$O(n)$	$O(n)$	$O(n)$	0.27
splitandsort	70	?	?	$O(n^3)$	61.4
subtrees	8	?	$O(n^2)$	$O(n^2)$	3.6
tuples	33	?	?	?	–