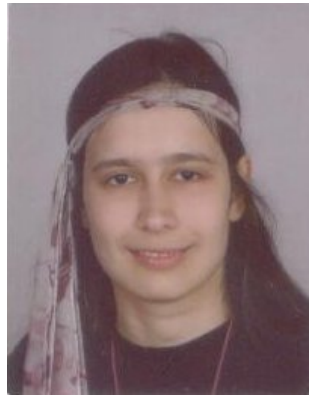
Improving Static Dependency Pairs for Higher-Order Rewriting

Carsten Fuhs



Cynthia Kop

Birkbeck, University of London, United Kingdom

Radboud Universiteit Nijmegen, The Netherlands

16th Workshop on Termination

Oxford, UK

19 July 2018

Why is termination of *higher-order* rewriting interesting?

Recently:

- Broader interest in termination analysis tools for higher-order rewriting
- Categories at the **Termination Competition**
  - "Higher-order rewriting union β": since 2010
  - "Rewriting with higher-order pattern matching on β-normal η-long forms (HRS)": under discussion
- New tools at TermComp 2017 (`SOL`) and 2018 (`SizeChangeTool`)
- Growing number of benchmarks in the **TPDB**
- Tools also used as back-end oracles in **Confluence Competition**

Higher-order rewriting in practice

- Specification language (with syntactic sugar) for **compiler synthesis**:
  CRSX [Rose, *RTA'11*] (successor tool `TransScript` used at IBM)

  `http://crsx.sourceforge.net/`

  Termination of higher-order specification
  $\Rightarrow$ **termination of the executable compiler**

- Termination analysis for functional programs

- Nuances in definitions of higher-order rewriting matter

  $\rightarrow$ discussion on `termtools` mailing list about
  plain matching vs matching *modulo beta*
  at TermComp 2018

What, exactly, do you *mean* by higher-order rewriting?

Many different formalisms for higher-order rewriting in the literature

What, exactly, do you *mean* by higher-order rewriting?

Many different formalisms for higher-order rewriting in the literature, inspired by

- first-order term rewriting
- typed λ-calculi
- functional programming languages
- ...

What, exactly, do you *mean* by higher-order rewriting?

Many different formalisms for higher-order rewriting in the literature, inspired by

- first-order term rewriting
- typed λ-calculi
- functional programming languages
- ...

Here: (simplified version of) Algebraic Functional Systems with Meta-Variables (AFSMs) used in termination tool `Wanda`

`http://wandahot.sourceforge.net/`

- allows transformations from many formalisms of interest for termination
- none of those inconvenient non-patterns
- no automatic beta-reduction (matching is syntactic)
- AFSMs directly contain the format in TermComp
- also covers (second-order) pattern HRSs ($\rightarrow$ CoCo)

What, exactly, do you *mean* by higher-order rewriting?

Many different formalisms for higher-order rewriting in the literature, inspired by

- first-order term rewriting
- typed λ-calculi
- functional programming languages
- ...

Here: (simplified version of) Algebraic Functional Systems with Meta-Variables (AFSMs) used in termination tool `Wanda`

`http://wandahot.sourceforge.net/`

- allows transformations from many formalisms of interest for termination
- none of those inconvenient non-patterns
- no automatic beta-reduction (matching is syntactic)
- AFSMs directly contain the format in TermComp
- also covers (second-order) pattern HRSs ($\rightarrow$ CoCo)

What, exactly, do you *mean* by higher-order rewriting?

Many different formalisms for higher-order rewriting in the literature, inspired by

- first-order term rewriting
- typed λ-calculi
- functional programming languages
- ...

Here: (simplified version of) Algebraic Functional Systems with Meta-Variables (AFSMs) used in termination tool `Wanda`

`http://wandahot.sourceforge.net/`

- allows transformations from many formalisms of interest for termination
- none of those inconvenient non-patterns
- no automatic beta-reduction (matching is syntactic)
- AFSMs directly contain the format in TermComp
- also covers (second-order) pattern HRSs ($\rightarrow$ CoCo)

What, exactly, do you *mean* by higher-order rewriting?

Many different formalisms for higher-order rewriting in the literature, inspired by

- first-order term rewriting
- typed λ-calculi
- functional programming languages
- ...

Here: (simplified version of) Algebraic Functional Systems with Meta-Variables (AFSMs) used in termination tool `Wanda`

`http://wandahot.sourceforge.net/`

- allows transformations from many formalisms of interest for termination
- none of those inconvenient non-patterns
- no automatic beta-reduction (matching is syntactic)
- AFSMs directly contain the format in TermComp
- also covers (second-order) pattern HRSs ($\rightarrow$ CoCo)

What, exactly, do you *mean* by higher-order rewriting?

Many different formalisms for higher-order rewriting in the literature, inspired by

- first-order term rewriting
- typed λ-calculi
- functional programming languages
- ...

Here: (simplified version of) Algebraic Functional Systems with Meta-Variables (AFSMs) used in termination tool `Wanda`

`http://wandahot.sourceforge.net/`

- allows transformations from many formalisms of interest for termination
- none of those inconvenient non-patterns
- no automatic beta-reduction (matching is syntactic)
- AFSMs directly contain the format in TermComp
- also covers (second-order) pattern HRSs ($\rightarrow$ CoCo)

What, exactly, do you *mean* by higher-order rewriting?

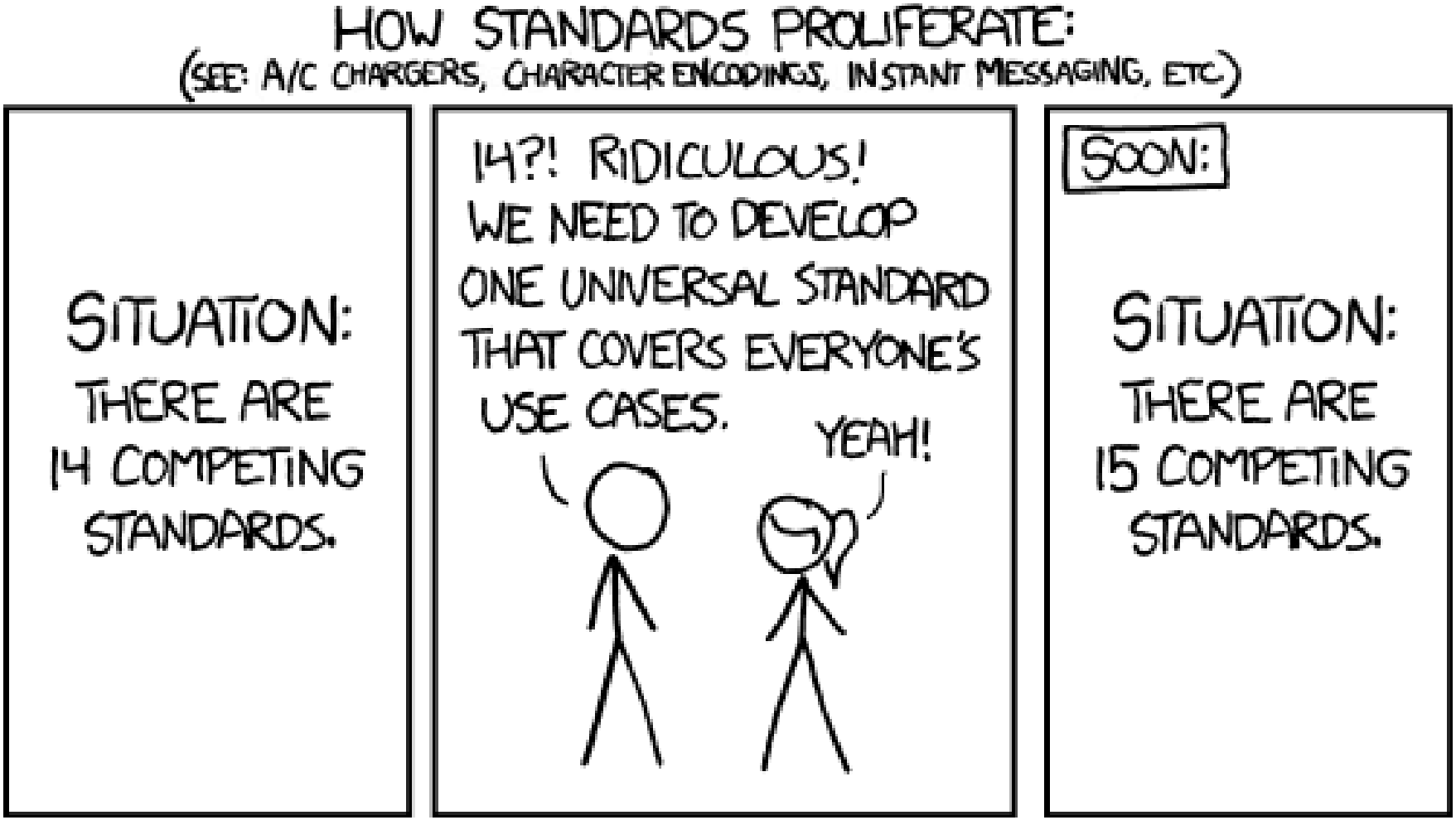
Many different formalisms for higher-order rewriting in the literature, inspired by

- first-order term rewriting
- typed λ-calculi
- functional programming languages
- ...

Here: (simplified version of) Algebraic Functional Systems with Meta-Variables (AFSMs) used in termination tool `Wanda`

`http://wandahot.sourceforge.net/`

- allows transformations from many formalisms of interest for termination
- none of those inconvenient non-patterns
- no automatic beta-reduction (matching is syntactic)
- AFSMs directly contain the format in TermComp
- also covers (second-order) pattern HRSs ($\rightarrow$ CoCo)



https://xkcd.com/927/

AFSMs: definition by example

$$\begin{array}{rcl} \text{map } F \text{ nil} & \to & \text{nil} \\ \text{map } F \text{ (cons } x \text{ } xs) & \to & \text{cons } (F \text{ } x) \text{ (map } F \text{ } xs) \\ \text{twice } \lambda x.G[x] & \to & \lambda y.G[G[y]] \end{array}$$

- variables for binding, meta-variables for matching
- matching using meta-variables (with arity) $G[x] : \text{nat}$
- application $F \cdot x$
- beta-rule always present $(\lambda x.s) \cdot t \to_\beta s[x := t]$
- simply-typed terms
- functional output types allowed $\text{twice } \lambda x.G[x] : \text{nat} \Rightarrow \text{nat}$
- rules may have functional type $\text{twice } \lambda x.G[x] \to \lambda y.G[G[y]]$
- higher-order meta-variables do not *have* to take arguments

$$\text{twice } G \to \lambda y.G \cdot (G \cdot y)$$

AFSMs: definition by example

$$\begin{array}{rcl} \text{map } F \text{ nil} & \to & \text{nil} \\ \text{map } F \text{ (cons } x \text{ } xs) & \to & \text{cons } (F \text{ } x) \text{ (map } F \text{ } xs) \\ \text{twice } \lambda x.G[x] & \to & \lambda y.G[G[y]] \end{array}$$

- variables for binding, meta-variables for matching
- matching using meta-variables (with arity) $G[x] : \text{nat}$
- application $F \cdot x$
- beta-rule always present $(\lambda x.s) \cdot t \to_\beta s[x := t]$
- simply-typed terms
- functional output types allowed $\text{twice } \lambda x.G[x] : \text{nat} \Rightarrow \text{nat}$
- rules may have functional type $\text{twice } \lambda x.G[x] \to \lambda y.G[G[y]]$
- higher-order meta-variables do not *have* to take arguments

$$\text{twice } G \to \lambda y.G \cdot (G \cdot y)$$

AFSMs: definition by example

$$\begin{array}{rcl} \text{map } F \text{ nil} & \to & \text{nil} \\ \text{map } F \text{ (cons } x \text{ } xs) & \to & \text{cons } (F \text{ } x) \text{ (map } F \text{ } xs) \\ \text{twice } \lambda x.G[x] & \to & \lambda y.G[G[y]] \end{array}$$

- variables for binding, meta-variables for matching
- matching using meta-variables (with arity) $G[x] : \text{nat}$
- application $F \cdot x$
- beta-rule always present $(\lambda x.s) \cdot t \to_\beta s[x := t]$
- simply-typed terms
- functional output types allowed $\text{twice } \lambda x.G[x] : \text{nat} \Rightarrow \text{nat}$
- rules may have functional type $\text{twice } \lambda x.G[x] \to \lambda y.G[G[y]]$
- higher-order meta-variables do not *have* to take arguments

$$\text{twice } G \to \lambda y.G \cdot (G \cdot y)$$

AFSMs: definition by example

$$\begin{array}{rcl} \text{map } F \text{ nil} & \to & \text{nil} \\ \text{map } F \text{ (cons } x \text{ } xs) & \to & \text{cons } (F \text{ } x) \text{ (map } F \text{ } xs) \\ \text{twice } \lambda x.G[x] & \to & \lambda y.G[G[y]] \end{array}$$

- variables for binding, meta-variables for matching
- matching using meta-variables (with arity) $G[x] : \text{nat}$
- application $F \cdot x$
- beta-rule always present $(\lambda x.s) \cdot t \to_\beta s[x := t]$
- simply-typed terms
- functional output types allowed $\text{twice } \lambda x.G[x] : \text{nat} \Rightarrow \text{nat}$
- rules may have functional type $\text{twice } \lambda x.G[x] \to \lambda y.G[G[y]]$
- higher-order meta-variables do not *have* to take arguments

$$\text{twice } G \to \lambda y.G \cdot (G \cdot y)$$

AFSMs: definition by example

$$\begin{array}{rcl} \text{map } F \text{ nil} & \to & \text{nil} \\ \text{map } F \text{ (cons } x \text{ } xs) & \to & \text{cons } (F \text{ } x) \text{ (map } F \text{ } xs) \\ \text{twice } \lambda x.G[x] & \to & \lambda y.G[G[y]] \end{array}$$

- variables for binding, meta-variables for matching
- matching using meta-variables (with arity) $G[x] : \text{nat}$
- application $F \cdot x$
- beta-rule always present $(\lambda x.s) \cdot t \to_\beta s[x := t]$
- simply-typed terms
- functional output types allowed twice $\lambda x.G[x] : \text{nat} \Rightarrow \text{nat}$
- rules may have functional type twice $\lambda x.G[x] \to \lambda y.G[G[y]]$
- higher-order meta-variables do not *have* to take arguments

  twice $G \to \lambda y.G \cdot (G \cdot y)$

AFSMs: definition by example

$$\begin{array}{rcl} \text{map } F \text{ nil} & \to & \text{nil} \\ \text{map } F \text{ (cons } x \text{ } xs) & \to & \text{cons } (F \text{ } x) \text{ (map } F \text{ } xs) \\ \text{twice } \lambda x.G[x] & \to & \lambda y.G[G[y]] \end{array}$$

- variables for binding, meta-variables for matching
- matching using meta-variables (with arity) $G[x] : \text{nat}$
- application $F \cdot x$
- beta-rule always present $(\lambda x.s) \cdot t \to_\beta s[x := t]$
- simply-typed terms
- functional output types allowed $\text{twice } \lambda x.G[x] : \text{nat} \Rightarrow \text{nat}$
- rules may have functional type $\text{twice } \lambda x.G[x] \to \lambda y.G[G[y]]$
- higher-order meta-variables do not *have* to take arguments

$$\text{twice } G \to \lambda y.G \cdot (G \cdot y)$$

AFSMs: definition by example

$$\begin{array}{rcl} \text{map } F \text{ nil} & \to & \text{nil} \\ \text{map } F \text{ (cons } x \text{ } xs) & \to & \text{cons } (F \text{ } x) \text{ (map } F \text{ } xs) \\ \text{twice } \lambda x.G[x] & \to & \lambda y.G[G[y]] \end{array}$$

- variables for binding, meta-variables for matching
- matching using meta-variables (with arity) $G[x] : \text{nat}$
- application $F \cdot x$
- beta-rule always present $(\lambda x.s) \cdot t \to_\beta s[x := t]$
- simply-typed terms
- functional output types allowed twice $\lambda x.G[x] : \text{nat} \Rightarrow \text{nat}$
- rules may have functional type twice $\lambda x.G[x] \to \lambda y.G[G[y]]$
- higher-order meta-variables do not *have* to take arguments

  twice $G \to \lambda y.G \cdot (G \cdot y)$

AFSMs: definition by example

$$\begin{array}{rcl} \text{map } F \text{ nil} & \to & \text{nil} \\ \text{map } F \text{ (cons } x \text{ } xs) & \to & \text{cons } (F\ x)\ (\text{map } F\ xs) \\ \text{twice } \lambda x.G[x] & \to & \lambda y.G[G[y]] \end{array}$$

- variables for binding, meta-variables for matching
- matching using meta-variables (with arity) $G[x] : \text{nat}$
- application $F \cdot x$
- beta-rule always present $(\lambda x.s) \cdot t \to_\beta s[x := t]$
- simply-typed terms
- functional output types allowed twice $\lambda x.G[x] : \text{nat} \Rightarrow \text{nat}$
- rules may have functional type twice $\lambda x.G[x] \to \lambda y.G[G[y]]$
- higher-order meta-variables do not *have* to take arguments

twice $G \to \lambda y.G \cdot (G \cdot y)$

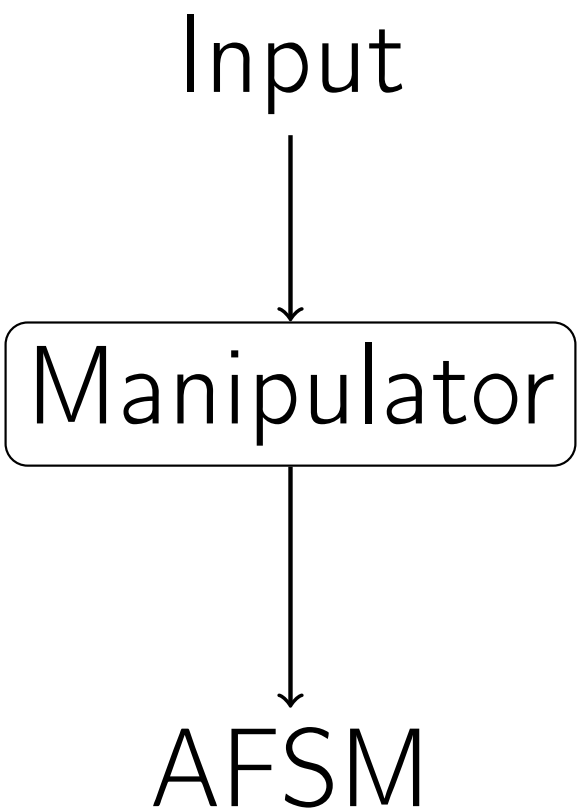
AFSMs: definition by example

$$\begin{array}{rcl} \text{map } F \text{ nil} & \to & \text{nil} \\ \text{map } F \text{ (cons } x \text{ } xs) & \to & \text{cons } (F \text{ } x) \text{ (map } F \text{ } xs) \\ \text{twice } \lambda x.G[x] & \to & \lambda y.G[G[y]] \end{array}$$

- variables for binding, meta-variables for matching
- matching using meta-variables (with arity) $G[x] : \text{nat}$
- application $F \cdot x$
- beta-rule always present $(\lambda x.s) \cdot t \to_\beta s[x := t]$
- simply-typed terms
- functional output types allowed twice $\lambda x.G[x] : \text{nat} \Rightarrow \text{nat}$
- rules may have functional type twice $\lambda x.G[x] \to \lambda y.G[G[y]]$
- higher-order meta-variables do not *have* to take arguments

  twice $G \to \lambda y.G \cdot (G \cdot y)$

AFSMs in Wanda at TermComp, CoCo, ...



Termination of First-Order Term Rewriting: append

Example (append)

$$\mathcal{R}: \quad \begin{array}{rcl} \mathsf{append}\ \mathsf{nil}\ ys & \to & ys \\ \mathsf{append}\ (\mathsf{cons}\ x\ xs)\ ys & \to & \mathsf{cons}\ x\ (\mathsf{append}\ xs\ ys) \end{array}$$

$$\mathsf{0} : \mathsf{nat}, \quad \mathsf{s} : \mathsf{nat} \Rightarrow \mathsf{nat}, \quad \mathsf{nil} : \mathsf{list}, \quad \mathsf{cons} : \mathsf{nat} \Rightarrow \mathsf{list} \Rightarrow \mathsf{list}$$
$$\mathsf{append} : \mathsf{list} \Rightarrow \mathsf{list} \Rightarrow \mathsf{list}$$

Termination of First-Order Term Rewriting: append

Example (append)

$\mathcal{R}$:

$$\begin{array}{rcl} \mathsf{append}\ \mathsf{nil}\ ys & \to & ys \\ \mathsf{append}\ (\mathsf{cons}\ x\ xs)\ ys & \to & \mathsf{cons}\ x\ (\mathsf{append}\ xs\ ys) \end{array}$$

$$\mathsf{0} : \mathsf{nat}, \quad \mathsf{s} : \mathsf{nat} \Rightarrow \mathsf{nat}, \quad \mathsf{nil} : \mathsf{list}, \quad \mathsf{cons} : \mathsf{nat} \Rightarrow \mathsf{list} \Rightarrow \mathsf{list}$$
$$\mathsf{append} : \mathsf{list} \Rightarrow \mathsf{list} \Rightarrow \mathsf{list}$$

First-Order Rewriting: Evaluate terms by applying rules from $\mathcal{R}$

$$\mathsf{append}\ (\mathsf{cons}\ \mathsf{0}\ \mathsf{nil})\ \mathsf{nil} \to_{\mathcal{R}} \mathsf{cons}\ \mathsf{0}\ (\mathsf{append}\ \mathsf{nil}\ \mathsf{nil}) \to_{\mathcal{R}} \mathsf{cons}\ \mathsf{0}\ \mathsf{nil}$$

Termination of First-Order Term Rewriting: append

Example (append)

$\mathcal{R}$:

$$\begin{array}{rcl} \mathsf{append}\ \mathsf{nil}\ ys & \to & ys \\ \mathsf{append}\ (\mathsf{cons}\ x\ xs)\ ys & \to & \mathsf{cons}\ x\ (\mathsf{append}\ xs\ ys) \end{array}$$

$$\mathsf{0} : \mathsf{nat}, \quad \mathsf{s} : \mathsf{nat} \Rightarrow \mathsf{nat}, \quad \mathsf{nil} : \mathsf{list}, \quad \mathsf{cons} : \mathsf{nat} \Rightarrow \mathsf{list} \Rightarrow \mathsf{list}$$

$$\mathsf{append} : \mathsf{list} \Rightarrow \mathsf{list} \Rightarrow \mathsf{list}$$

First-Order Rewriting: Evaluate terms by applying rules from $\mathcal{R}$

$$\mathsf{append}\ (\mathsf{cons}\ \mathsf{0}\ \mathsf{nil})\ \mathsf{nil} \to_{\mathcal{R}} \mathsf{cons}\ \mathsf{0}\ (\mathsf{append}\ \mathsf{nil}\ \mathsf{nil}) \to_{\mathcal{R}} \mathsf{cons}\ \mathsf{0}\ \mathsf{nil}$$

Termination (aka Strong Normalisation): no infinite $\to_{\mathcal{R}}$ sequences

Termination of First-Order Term Rewriting: append

Example (append)

$\mathcal{R}$:

$$\begin{array}{rcl} \mathsf{append}\ \mathsf{nil}\ ys & \to & ys \\ \mathsf{append}\ (\mathsf{cons}\ x\ xs)\ ys & \to & \mathsf{cons}\ x\ (\mathsf{append}\ xs\ ys) \end{array}$$

$$\mathsf{0} : \mathsf{nat}, \quad \mathsf{s} : \mathsf{nat} \Rightarrow \mathsf{nat}, \quad \mathsf{nil} : \mathsf{list}, \quad \mathsf{cons} : \mathsf{nat} \Rightarrow \mathsf{list} \Rightarrow \mathsf{list}$$

$$\mathsf{append} : \mathsf{list} \Rightarrow \mathsf{list} \Rightarrow \mathsf{list}$$

First-Order Rewriting: Evaluate terms by applying rules from $\mathcal{R}$

$$\mathsf{append}\ (\mathsf{cons}\ \mathsf{0}\ \mathsf{nil})\ \mathsf{nil} \to_{\mathcal{R}} \mathsf{cons}\ \mathsf{0}\ (\mathsf{append}\ \mathsf{nil}\ \mathsf{nil}) \to_{\mathcal{R}} \mathsf{cons}\ \mathsf{0}\ \mathsf{nil}$$

Termination (aka Strong Normalisation): no infinite $\to_{\mathcal{R}}$ sequences

Show termination using Dependency Pairs [Arts, Giesl, *TCS'00*]:

Termination of First-Order Term Rewriting: append

Example (append)

$\mathcal{R}$:

$$\begin{array}{rcl} \text{append nil } ys & \to & ys \\ \text{append } (\text{cons } x\ xs)\ ys & \to & \text{cons } x\ (\text{append } xs\ ys) \end{array}$$

$$\text{0 : nat}, \quad \text{s : nat} \Rightarrow \text{nat}, \quad \text{nil : list}, \quad \text{cons : nat} \Rightarrow \text{list} \Rightarrow \text{list}$$
$$\text{append : list} \Rightarrow \text{list} \Rightarrow \text{list}$$

First-Order Rewriting: Evaluate terms by applying rules from $\mathcal{R}$

$$\text{append (cons 0 nil) nil} \to_{\mathcal{R}} \text{cons 0 (append nil nil)} \to_{\mathcal{R}} \text{cons 0 nil}$$

Termination (aka Strong Normalisation): no infinite $\to_{\mathcal{R}}$ sequences

Show termination using Dependency Pairs [Arts, Giesl, *TCS'00*]:

- For TRS $\mathcal{R}$ build dependency pairs $\mathcal{DP}$ ($\sim$ function calls)

Termination of First-Order Term Rewriting: append

Example (append)

$\mathcal{R}$:

$$\begin{aligned} \text{append nil } ys &\to ys \\ \text{append (cons } x\ xs)\ ys &\to \text{cons } x\ (\text{append } xs\ ys) \end{aligned}$$

$$\text{0 : nat},\quad \text{s : nat} \Rightarrow \text{nat},\quad \text{nil : list},\quad \text{cons : nat} \Rightarrow \text{list} \Rightarrow \text{list}$$
$$\text{append : list} \Rightarrow \text{list} \Rightarrow \text{list}$$

First-Order Rewriting: Evaluate terms by applying rules from $\mathcal{R}$

$$\text{append (cons 0 nil) nil} \to_{\mathcal{R}} \text{cons 0 (append nil nil)} \to_{\mathcal{R}} \text{cons 0 nil}$$

Termination (aka Strong Normalisation): no infinite $\to_{\mathcal{R}}$ sequences

Example (Dependency Pair for append)

$\mathcal{DP}$:

$$\text{append}^\sharp\ (\text{cons}\ x\ xs)\ ys \Rightarrow \text{append}^\sharp\ xs\ ys$$

Show termination using Dependency Pairs [Arts, Giesl, *TCS'00*]:

- For TRS $\mathcal{R}$ build dependency pairs $\mathcal{DP}$ ($\sim$ function calls)

Termination of First-Order Term Rewriting: append

Example (append)

$\mathcal{R}$:

$$\begin{array}{rcl} \text{append nil } ys & \to & ys \\ \text{append (cons } x\ xs)\ ys & \to & \text{cons } x\ (\text{append } xs\ ys) \end{array}$$

$$0 : \text{nat}, \quad s : \text{nat} \Rightarrow \text{nat}, \quad \text{nil} : \text{list}, \quad \text{cons} : \text{nat} \Rightarrow \text{list} \Rightarrow \text{list}$$
$$\text{append} : \text{list} \Rightarrow \text{list} \Rightarrow \text{list}$$

First-Order Rewriting: Evaluate terms by applying rules from $\mathcal{R}$

$$\text{append (cons 0 nil) nil} \to_{\mathcal{R}} \text{cons 0 (append nil nil)} \to_{\mathcal{R}} \text{cons 0 nil}$$

Termination (aka Strong Normalisation): no infinite $\to_{\mathcal{R}}$ sequences

Example (Dependency Pair for append)

$\mathcal{DP}$:

$$\text{append}^\sharp\ (\text{cons}\ x\ xs)\ ys \quad \Rightarrow \quad \text{append}^\sharp\ xs\ ys$$

Show termination using Dependency Pairs [Arts, Giesl, *TCS'00*]:

- For TRS $\mathcal{R}$ build dependency pairs $\mathcal{DP}$ ($\sim$ function calls)
- Show: No ∞ call sequence with $\mathcal{DP}$ (eval of $\mathcal{DP}$'s args via $\mathcal{R}$)

Dependency Pairs and the Subterm Criterion

Example (Dependency Pair for append)

$\mathcal{DP}$: $\quad \mathsf{append}^\sharp\ (\mathsf{cons}\ x\ xs)\ ys \quad \Rightarrow\!\!\!\!\Rightarrow \quad \mathsf{append}^\sharp\ xs\ ys$

Show termination using Dependency Pairs [Arts, Giesl, *TCS'00*]:

- Show: No ∞ call sequence with $\mathcal{DP}$ (eval of $\mathcal{DP}$'s args via $\mathcal{R}$)

Dependency Pairs and the Subterm Criterion

Example (Dependency Pair for append)

$\mathcal{DP}$: $\quad \mathsf{append}^\sharp\ (\mathsf{cons}\ x\ xs)\ ys \quad \Rrightarrow \quad \mathsf{append}^\sharp\ xs\ ys$

Show termination using Dependency Pairs [Arts, Giesl, *TCS'00*]:

- Show: No ∞ call sequence with $\mathcal{DP}$ (eval of $\mathcal{DP}$'s args via $\mathcal{R}$)
- Call sequence: infinite minimal chain (all t_i are $\mathcal{R}$-terminating)

$$s_1 \Rrightarrow t_1 \to_{\mathcal{R}}^* s_2 \Rrightarrow t_2 \to_{\mathcal{R}}^* s_3 \Rrightarrow t_3 \to_{\mathcal{R}}^* \cdots$$

Dependency Pairs and the Subterm Criterion

Example (Dependency Pair for append)

$\mathcal{DP}$: $\quad \mathsf{append}^\sharp\ (\mathsf{cons}\ x\ xs)\ ys \quad \Rrightarrow \quad \mathsf{append}^\sharp\ xs\ ys$

Show termination using Dependency Pairs [Arts, Giesl, *TCS'00*]:

- Show: No ∞ call sequence with $\mathcal{DP}$ (eval of $\mathcal{DP}$'s args via $\mathcal{R}$)
- Call sequence: infinite minimal chain (all t_i are $\mathcal{R}$-terminating)

$$s_1 \Rrightarrow t_1 \to_{\mathcal{R}}^* s_2 \Rrightarrow t_2 \to_{\mathcal{R}}^* s_3 \Rrightarrow t_3 \to_{\mathcal{R}}^* \cdots$$

Subterm criterion [Hirokawa, Middeldorp, *IC'07*]:

Dependency Pairs and the Subterm Criterion

Example (Dependency Pair for append)

$\mathcal{DP}$: $\quad \mathsf{append}^\sharp\ (\mathsf{cons}\ x\ xs)\ ys \quad \Rrightarrow \quad \mathsf{append}^\sharp\ xs\ ys$

Show termination using Dependency Pairs [Arts, Giesl, *TCS'00*]:

- Show: No ∞ call sequence with $\mathcal{DP}$ (eval of $\mathcal{DP}$'s args via $\mathcal{R}$)
- Call sequence: infinite minimal chain (all t_i are $\mathcal{R}$-terminating)

$$s_1 \Rrightarrow t_1 \to_{\mathcal{R}}^* s_2 \Rrightarrow t_2 \to_{\mathcal{R}}^* s_3 \Rrightarrow t_3 \to_{\mathcal{R}}^* \cdots$$

Subterm criterion [Hirokawa, Middeldorp, *IC'07*]:

- Use simple projection ν mapping each $\sharp$-symbol in $\mathcal{DP}$ to an argument

  $\nu(\mathsf{append}^\sharp) = 1$

Dependency Pairs and the Subterm Criterion

Example (Dependency Pair for append)

$\mathcal{DP}$: $\quad \mathsf{append}^\sharp\ (\mathsf{cons}\ x\ xs)\ ys \quad \Rightarrow \quad \mathsf{append}^\sharp\ xs\ ys$

Show termination using Dependency Pairs [Arts, Giesl, *TCS'00*]:

- Show: No ∞ call sequence with $\mathcal{DP}$ (eval of $\mathcal{DP}$'s args via $\mathcal{R}$)
- Call sequence: infinite minimal chain (all t_i are $\mathcal{R}$-terminating)

$$s_1 \ \Rightarrow \ t_1 \ \to_{\mathcal{R}}^* \ s_2 \ \Rightarrow \ t_2 \ \to_{\mathcal{R}}^* \ s_3 \ \Rightarrow \ t_3 \ \to_{\mathcal{R}}^* \ \cdots$$

Subterm criterion [Hirokawa, Middeldorp, *IC'07*]:

- Use simple projection ν mapping each $\sharp$-symbol in $\mathcal{DP}$ to an argument

  $\nu(\ \mathsf{append}^\sharp\ ) = 1$
- Extend to terms: $\nu(\ f^\sharp\ t_1\ \ldots\ t_n\ ) = t_{\nu(f^\sharp)}$

  $\nu(\ \mathsf{append}^\sharp\ (\mathsf{cons}\ x\ xs)\ ys\ ) = \mathsf{cons}\ x\ xs$

Dependency Pairs and the Subterm Criterion

Example (Dependency Pair for append)

$\mathcal{DP}$: $\quad \mathsf{append}^\sharp\ (\mathsf{cons}\ x\ xs)\ ys \quad \Rrightarrow \quad \mathsf{append}^\sharp\ xs\ ys$

Show termination using Dependency Pairs [Arts, Giesl, *TCS'00*]:

- Show: No ∞ call sequence with $\mathcal{DP}$ (eval of $\mathcal{DP}$'s args via $\mathcal{R}$)
- Call sequence: infinite minimal chain (all t_i are $\mathcal{R}$-terminating)

$$\nu(s_1) \rhd \nu(t_1) \to_{\mathcal{R}}^* \nu(s_2) \rhd \nu(t_2) \to_{\mathcal{R}}^* \nu(s_3) \rhd \nu(t_3) \to_{\mathcal{R}}^* \quad \cdots \quad \lightning$$

Subterm criterion [Hirokawa, Middeldorp, *IC'07*]:

- Use simple projection ν mapping each $\sharp$-symbol in $\mathcal{DP}$ to an argument

  $\nu(\ \mathsf{append}^\sharp\ ) = 1$
- Extend to terms: $\nu(\ f^\sharp\ t_1\ \ \ldots\ \ t_n\ ) = t_{\nu(f^\sharp)}$

  $\nu(\ \mathsf{append}^\sharp\ (\mathsf{cons}\ x\ xs)\ ys\ ) = \mathsf{cons}\ x\ xs$
- If $s \rhd t$ for all $s \Rrightarrow t \in \mathcal{DP}$: termination proved!

Dependency Pairs and the Subterm Criterion

Example (Dependency Pair for append)

$\mathcal{DP}$: cons x xs $\rhd$ xs

Show termination using Dependency Pairs [Arts, Giesl, *TCS'00*]:

- Show: No ∞ call sequence with $\mathcal{DP}$ (eval of $\mathcal{DP}$'s args via $\mathcal{R}$)
- Call sequence: infinite minimal chain (all t_i are $\mathcal{R}$-terminating)

$$\nu(s_1) \rhd \nu(t_1) \to_{\mathcal{R}}^* \nu(s_2) \rhd \nu(t_2) \to_{\mathcal{R}}^* \nu(s_3) \rhd \nu(t_3) \to_{\mathcal{R}}^* \quad \cdots \quad \lightning$$

Subterm criterion [Hirokawa, Middeldorp, *IC'07*]:

- Use simple projection ν mapping each $\sharp$-symbol in $\mathcal{DP}$ to an argument

  $\nu(\,\mathsf{append}^\sharp\,) = 1$
- Extend to terms: $\nu(\,f^\sharp\ t_1\ \ldots\ t_n\,) = t_{\nu(f^\sharp)}$

  $\nu(\,\mathsf{append}^\sharp\ (\mathsf{cons}\ x\ xs)\ ys\,) = \mathsf{cons}\ x\ xs$
- If $s \rhd t$ for all $s \Rrightarrow t \in \mathcal{DP}$: termination proved!

Termination of Higher-Order Term Rewriting: map

Example (map)

$$\begin{array}{rcl} \mathsf{map}\ F\ \mathsf{nil} & \to & \mathsf{nil} \\ \mathsf{map}\ F\ (\mathsf{cons}\ x\ xs) & \to & \mathsf{cons}\ (F\ x)\ (\mathsf{map}\ F\ xs) \end{array}$$

$$\mathsf{0} : \mathsf{nat}, \quad \mathsf{s} : \mathsf{nat} \Rightarrow \mathsf{nat}, \quad \mathsf{nil} : \mathsf{list}, \quad \mathsf{cons} : \mathsf{nat} \Rightarrow \mathsf{list} \Rightarrow \mathsf{list},$$
$$\mathsf{map} : (\mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{list} \Rightarrow \mathsf{list}$$

Higher-Order Rewriting: Evaluate terms by applying rules from $\mathcal{R}$

Termination of Higher-Order Term Rewriting: map

Example (map)

$$\begin{array}{rcl} \mathsf{map}\ F\ \mathsf{nil} & \to & \mathsf{nil} \\ \mathsf{map}\ F\ (\mathsf{cons}\ x\ xs) & \to & \mathsf{cons}\ (F\ x)\ (\mathsf{map}\ F\ xs) \end{array}$$

$$\mathsf{0} : \mathsf{nat}, \quad \mathsf{s} : \mathsf{nat} \Rightarrow \mathsf{nat}, \quad \mathsf{nil} : \mathsf{list}, \quad \mathsf{cons} : \mathsf{nat} \Rightarrow \mathsf{list} \Rightarrow \mathsf{list},$$
$$\mathsf{map} : (\mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{list} \Rightarrow \mathsf{list}$$

Higher-Order Rewriting: Evaluate terms by applying rules from $\mathcal{R} + \to_\beta$

$$\begin{aligned} \mathsf{map}\ (\lambda x.\mathsf{s}\ x)\ (\mathsf{cons}\ \mathsf{0}\ \mathsf{nil}) &\to_{\mathcal{R}} \mathsf{cons}\ ((\lambda x.\mathsf{s}\ x)\ \mathsf{0})\ (\mathsf{map}\ (\lambda x.\mathsf{s}\ x)\ \mathsf{nil}) \\ &\to_{\mathcal{R}} \mathsf{cons}\ ((\lambda x.\mathsf{s}\ x)\ \mathsf{0})\ \mathsf{nil} \to_\beta \mathsf{cons}\ (\mathsf{s}\ \mathsf{0})\ \mathsf{nil} \end{aligned}$$

Termination of Higher-Order Term Rewriting: map

Example (map)

$$\begin{array}{rcl} \mathsf{map}\ F\ \mathsf{nil} & \to & \mathsf{nil} \\ \mathsf{map}\ F\ (\mathsf{cons}\ x\ xs) & \to & \mathsf{cons}\ (F\ x)\ (\mathsf{map}\ F\ xs) \end{array}$$

$$\mathsf{0} : \mathsf{nat}, \quad \mathsf{s} : \mathsf{nat} \Rightarrow \mathsf{nat}, \quad \mathsf{nil} : \mathsf{list}, \quad \mathsf{cons} : \mathsf{nat} \Rightarrow \mathsf{list} \Rightarrow \mathsf{list},$$
$$\mathsf{map} : (\mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{list} \Rightarrow \mathsf{list}$$

Higher-Order Rewriting: Evaluate terms by applying rules from $\mathcal{R} + \to_\beta$

$$\begin{aligned} \mathsf{map}\ (\lambda x.\mathsf{s}\ x)\ (\mathsf{cons}\ \mathsf{0}\ \mathsf{nil}) &\to_{\mathcal{R}} \mathsf{cons}\ ((\lambda x.\mathsf{s}\ x)\ \mathsf{0})\ (\mathsf{map}\ (\lambda x.\mathsf{s}\ x)\ \mathsf{nil}) \\ &\to_{\mathcal{R}} \mathsf{cons}\ ((\lambda x.\mathsf{s}\ x)\ \mathsf{0})\ \mathsf{nil} \to_\beta \mathsf{cons}\ (\mathsf{s}\ \mathsf{0})\ \mathsf{nil} \end{aligned}$$

Show termination by Static DPs [Sakai, Kusakari, *TIS'05*], [Blanqui, *WST'06*]:

Termination of Higher-Order Term Rewriting: map

Example (map)

$$\begin{array}{rcl} \mathsf{map}\ F\ \mathsf{nil} & \to & \mathsf{nil} \\ \mathsf{map}\ F\ (\mathsf{cons}\ x\ xs) & \to & \mathsf{cons}\ (F\ x)\ (\mathsf{map}\ F\ xs) \end{array}$$

$$\mathsf{0} : \mathsf{nat}, \quad \mathsf{s} : \mathsf{nat} \Rightarrow \mathsf{nat}, \quad \mathsf{nil} : \mathsf{list}, \quad \mathsf{cons} : \mathsf{nat} \Rightarrow \mathsf{list} \Rightarrow \mathsf{list},$$
$$\mathsf{map} : (\mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{list} \Rightarrow \mathsf{list}$$

Higher-Order Rewriting: Evaluate terms by applying rules from $\mathcal{R} + \to_\beta$

$$\begin{aligned} \mathsf{map}\ (\lambda x.\mathsf{s}\ x)\ (\mathsf{cons}\ \mathsf{0}\ \mathsf{nil}) &\to_{\mathcal{R}} \mathsf{cons}\ ((\lambda x.\mathsf{s}\ x)\ \mathsf{0})\ (\mathsf{map}\ (\lambda x.\mathsf{s}\ x)\ \mathsf{nil}) \\ &\to_{\mathcal{R}} \mathsf{cons}\ ((\lambda x.\mathsf{s}\ x)\ \mathsf{0})\ \mathsf{nil} \to_\beta \mathsf{cons}\ (\mathsf{s}\ \mathsf{0})\ \mathsf{nil} \end{aligned}$$

Show termination by Static DPs [Sakai, Kusakari, *TIS'05*], [Blanqui, *WST'06*]:

- For TRS $\mathcal{R}$ build dependency pairs $\mathcal{DP}$ ($\sim$ function calls)

Termination of Higher-Order Term Rewriting: map

Example (map)

$$\begin{array}{rcl} \mathsf{map}\ F\ \mathsf{nil} & \to & \mathsf{nil} \\ \mathsf{map}\ F\ (\mathsf{cons}\ x\ xs) & \to & \mathsf{cons}\ (F\ x)\ (\mathsf{map}\ F\ xs) \end{array}$$

$$\mathsf{0} : \mathsf{nat}, \quad \mathsf{s} : \mathsf{nat} \Rightarrow \mathsf{nat}, \quad \mathsf{nil} : \mathsf{list}, \quad \mathsf{cons} : \mathsf{nat} \Rightarrow \mathsf{list} \Rightarrow \mathsf{list},$$
$$\mathsf{map} : (\mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{list} \Rightarrow \mathsf{list}$$

Higher-Order Rewriting: Evaluate terms by applying rules from $\mathcal{R} + \to_\beta$

$$\begin{aligned} \mathsf{map}\ (\lambda x.\mathsf{s}\ x)\ (\mathsf{cons}\ \mathsf{0}\ \mathsf{nil}) &\to_{\mathcal{R}} \mathsf{cons}\ ((\lambda x.\mathsf{s}\ x)\ \mathsf{0})\ (\mathsf{map}\ (\lambda x.\mathsf{s}\ x)\ \mathsf{nil}) \\ &\to_{\mathcal{R}} \mathsf{cons}\ ((\lambda x.\mathsf{s}\ x)\ \mathsf{0})\ \mathsf{nil} \to_\beta \mathsf{cons}\ (\mathsf{s}\ \mathsf{0})\ \mathsf{nil} \end{aligned}$$

Example (Dependency Pair for map)

$$\mathcal{DP}: \quad \mathsf{map}^\sharp\ F\ (\mathsf{cons}\ x\ xs) \quad \Rrightarrow \quad \mathsf{map}^\sharp\ F\ xs$$

Show termination by Static DPs [Sakai, Kusakari, *TIS'05*], [Blanqui, *WST'06*]:

- For TRS $\mathcal{R}$ build dependency pairs $\mathcal{DP}$ ($\sim$ function calls)

Termination of Higher-Order Term Rewriting: map

Example (map)

$$\begin{array}{rcl} \mathsf{map}\ F\ \mathsf{nil} & \to & \mathsf{nil} \\ \mathsf{map}\ F\ (\mathsf{cons}\ x\ xs) & \to & \mathsf{cons}\ (F\ x)\ (\mathsf{map}\ F\ xs) \end{array}$$

$$\mathsf{0} : \mathsf{nat}, \quad \mathsf{s} : \mathsf{nat} \Rightarrow \mathsf{nat}, \quad \mathsf{nil} : \mathsf{list}, \quad \mathsf{cons} : \mathsf{nat} \Rightarrow \mathsf{list} \Rightarrow \mathsf{list},$$
$$\mathsf{map} : (\mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{list} \Rightarrow \mathsf{list}$$

Higher-Order Rewriting: Evaluate terms by applying rules from $\mathcal{R} + \to_\beta$

$$\begin{aligned} \mathsf{map}\ (\lambda x.\mathsf{s}\ x)\ (\mathsf{cons}\ \mathsf{0}\ \mathsf{nil}) &\to_{\mathcal{R}} \mathsf{cons}\ ((\lambda x.\mathsf{s}\ x)\ \mathsf{0})\ (\mathsf{map}\ (\lambda x.\mathsf{s}\ x)\ \mathsf{nil}) \\ &\to_{\mathcal{R}} \mathsf{cons}\ ((\lambda x.\mathsf{s}\ x)\ \mathsf{0})\ \mathsf{nil} \to_\beta \mathsf{cons}\ (\mathsf{s}\ \mathsf{0})\ \mathsf{nil} \end{aligned}$$

Example (Dependency Pair for map)

$$\mathcal{DP}: \quad \mathsf{map}^\sharp\ F\ (\mathsf{cons}\ x\ xs) \quad \Rrightarrow \quad \mathsf{map}^\sharp\ F\ xs$$

Show termination by Static DPs [Sakai, Kusakari, *TIS'05*], [Blanqui, *WST'06*]:

- For TRS $\mathcal{R}$ build dependency pairs $\mathcal{DP}$ ($\sim$ function calls)
- Show: No ∞ call sequence with $\mathcal{DP}$ (eval of $\mathcal{DP}$'s args via $\mathcal{R} + \to_\beta$)

Static Dependency Pairs and the Subterm Criterion

Example (Dependency Pair for map)

$$\mathcal{DP}: \qquad \mathsf{map}^\sharp\ F\ (\mathsf{cons}\ x\ xs) \quad \Rrightarrow \quad \mathsf{map}^\sharp\ F\ xs$$

Show termination by Static DPs [Sakai, Kusakari, *TIS'05*], [Blanqui, *WST'06*]:

- Show: No ∞ call sequence with $\mathcal{DP}$ (eval of $\mathcal{DP}$'s args via $\mathcal{R}$ + $\to_\beta$)
- Call sequence: infinite minimal chain (all t_i are $\mathcal{R}$-terminating)

$$s_1 \Rrightarrow t_1 \to_{\mathcal{R}}^* s_2 \Rrightarrow t_2 \to_{\mathcal{R}}^* s_3 \Rrightarrow t_3 \to_{\mathcal{R}}^* \cdots$$

Static Dependency Pairs and the Subterm Criterion

Example (Dependency Pair for map)

$$\mathcal{DP}: \qquad \mathsf{map}^\sharp\ F\ (\mathsf{cons}\ x\ xs) \quad \Rrightarrow \quad \mathsf{map}^\sharp\ F\ xs$$

Show termination by Static DPs [Sakai, Kusakari, *TIS'05*], [Blanqui, *WST'06*]:

- Show: No ∞ call sequence with $\mathcal{DP}$ (eval of $\mathcal{DP}$'s args via $\mathcal{R}$ + $\to_\beta$)
- Call sequence: infinite minimal chain (all t_i are $\mathcal{R}$-terminating)

$$s_1 \Rrightarrow t_1 \to_{\mathcal{R}}^* s_2 \Rrightarrow t_2 \to_{\mathcal{R}}^* s_3 \Rrightarrow t_3 \to_{\mathcal{R}}^* \cdots$$

Subterm criterion [Kusakari, Isogai, Sakai, Blanqui, *TIS'09*]:

Static Dependency Pairs and the Subterm Criterion

Example (Dependency Pair for map)

$$\mathcal{DP}: \qquad \mathsf{map}^\sharp\ F\ (\mathsf{cons}\ x\ xs) \quad \Rightarrow\!\!\!\!\Rightarrow \quad \mathsf{map}^\sharp\ F\ xs$$

Show termination by Static DPs [Sakai, Kusakari, *TIS'05*], [Blanqui, *WST'06*]:

- Show: No ∞ call sequence with $\mathcal{DP}$ (eval of $\mathcal{DP}$'s args via $\mathcal{R}$ + $\to_\beta$)
- Call sequence: infinite minimal chain (all t_i are $\mathcal{R}$-terminating)

$$s_1 \Rightarrow\!\!\!\!\Rightarrow t_1 \to_{\mathcal{R}}^* s_2 \Rightarrow\!\!\!\!\Rightarrow t_2 \to_{\mathcal{R}}^* s_3 \Rightarrow\!\!\!\!\Rightarrow t_3 \to_{\mathcal{R}}^* \cdots$$

Subterm criterion [Kusakari, Isogai, Sakai, Blanqui, *TIS'09*]:

- Use simple projection ν mapping each $\sharp$-symbol in $\mathcal{DP}$ to an argument

  $\nu(\mathsf{map}^\sharp) = 2$

Static Dependency Pairs and the Subterm Criterion

Example (Dependency Pair for map)

$$\mathcal{DP}: \qquad \mathsf{map}^\sharp\ F\ (\mathsf{cons}\ x\ xs) \quad \Rightarrow \quad \mathsf{map}^\sharp\ F\ xs$$

Show termination by Static DPs [Sakai, Kusakari, *TIS'05*], [Blanqui, *WST'06*]:

- Show: No ∞ call sequence with $\mathcal{DP}$ (eval of $\mathcal{DP}$'s args via $\mathcal{R}$ + $\to_\beta$)
- Call sequence: infinite minimal chain (all t_i are $\mathcal{R}$-terminating)

$$s_1 \ \Rightarrow \ t_1 \ \to_{\mathcal{R}}^* \ s_2 \ \Rightarrow \ t_2 \ \to_{\mathcal{R}}^* \ s_3 \ \Rightarrow \ t_3 \ \to_{\mathcal{R}}^* \ \cdots$$

Subterm criterion [Kusakari, Isogai, Sakai, Blanqui, *TIS'09*]:

- Use simple projection ν mapping each $\sharp$-symbol in $\mathcal{DP}$ to an argument

  $\nu(\ \mathsf{map}^\sharp\ ) = 2$
- Extend to terms: $\nu(\ f^\sharp\ t_1\ \ldots\ t_n\ ) = t_{\nu(f^\sharp)}$

  $\nu(\ \mathsf{map}^\sharp\ F\ (\mathsf{cons}\ x\ xs)\ ) = \mathsf{cons}\ x\ xs$

Static Dependency Pairs and the Subterm Criterion

Example (Dependency Pair for map)

$$\mathcal{DP}: \qquad \mathsf{map}^\sharp\ F\ (\mathsf{cons}\ x\ xs) \quad \Rightarrow\!\!\!\!\!\!\Rightarrow \quad \mathsf{map}^\sharp\ F\ xs$$

Show termination by Static DPs [Sakai, Kusakari, *TIS'05*], [Blanqui, *WST'06*]:

- Show: No ∞ call sequence with $\mathcal{DP}$ (eval of $\mathcal{DP}$'s args via $\mathcal{R}$ + $\to_\beta$)
- Call sequence: infinite minimal chain (all t_i are $\mathcal{R}$-terminating)

$$\nu(s_1) \rhd \nu(t_1) \to_{\mathcal{R}}^* \nu(s_2) \rhd \nu(t_2) \to_{\mathcal{R}}^* \nu(s_3) \rhd \nu(t_3) \to_{\mathcal{R}}^* \quad \cdots \quad \lightning$$

Subterm criterion [Kusakari, Isogai, Sakai, Blanqui, *TIS'09*]:

- Use simple projection ν mapping each $\sharp$-symbol in $\mathcal{DP}$ to an argument

  $\nu(\,\mathsf{map}^\sharp\,) = 2$
- Extend to terms: $\nu(\,f^\sharp\ t_1\ \ldots\ t_n\,) = t_{\nu(f^\sharp)}$

  $\nu(\,\mathsf{map}^\sharp\ F\ (\mathsf{cons}\ x\ xs)\,) = \mathsf{cons}\ x\ xs$
- If $s \rhd t$ for all $s \Rightarrow\!\!\!\!\!\!\Rightarrow t \in \mathcal{DP}$: termination proved!

Static Dependency Pairs and the Subterm Criterion

Example (Dependency Pair for map)

$\mathcal{DP}$: $\quad \mathsf{cons}\ x\ xs \quad \triangleright \quad xs$

Show termination by Static DPs [Sakai, Kusakari, *TIS'05*], [Blanqui, *WST'06*]:

- Show: No ∞ call sequence with $\mathcal{DP}$ (eval of $\mathcal{DP}$'s args via $\mathcal{R}$ + $\to_\beta$)
- Call sequence: infinite minimal chain (all t_i are $\mathcal{R}$-terminating)

$$\nu(s_1) \triangleright \nu(t_1) \to_{\mathcal{R}}^* \nu(s_2) \triangleright \nu(t_2) \to_{\mathcal{R}}^* \nu(s_3) \triangleright \nu(t_3) \to_{\mathcal{R}}^* \quad \cdots \quad \lightning$$

Subterm criterion [Kusakari, Isogai, Sakai, Blanqui, *TIS'09*]:

- Use simple projection ν mapping each $\sharp$-symbol in $\mathcal{DP}$ to an argument

  $\nu(\ \mathsf{map}^\sharp\ ) = 2$
- Extend to terms: $\nu(\ f^\sharp\ t_1\ \ \ldots\ \ t_n\ ) = t_{\nu(f^\sharp)}$

  $\nu(\ \mathsf{map}^\sharp\ F\ (\mathsf{cons}\ x\ xs)\ ) = \mathsf{cons}\ x\ xs$
- If $s \triangleright t$ for all $s \Rrightarrow t \in \mathcal{DP}$: termination proved!

Termination of Higher-Order Term Rewriting: app-lam

Example ( )

$$\mathcal{R}: \quad \frac{\mathsf{app}\ (\mathsf{lam}\ F)\ x \quad \to \quad F\ x}{\mathsf{app} : \mathsf{o} \Rightarrow \mathsf{o} \Rightarrow \mathsf{o}, \qquad \mathsf{lam} : (\mathsf{o} \Rightarrow \mathsf{o}) \Rightarrow \mathsf{o}}$$

Termination of Higher-Order Term Rewriting: app-lam

Example ( )

$\mathcal{R}$:

$$\frac{\mathsf{app}\ (\mathsf{lam}\ F)\ x \quad \to \quad F\ x}{\mathsf{app} : \mathsf{o} \Rightarrow \mathsf{o} \Rightarrow \mathsf{o}, \qquad \mathsf{lam} : (\mathsf{o} \Rightarrow \mathsf{o}) \Rightarrow \mathsf{o}}$$

No Static DPs for $\mathcal{R}$ at all! So $\mathcal{R}$ is terminating

Termination of Higher-Order Term Rewriting: app-lam

Example ( )

$\mathcal{R}$:

$$\frac{\mathsf{app}\ (\mathsf{lam}\ F)\ x \quad \to \quad F\ x}{\mathsf{app} : \mathsf{o} \Rightarrow \mathsf{o} \Rightarrow \mathsf{o}, \qquad \mathsf{lam} : (\mathsf{o} \Rightarrow \mathsf{o}) \Rightarrow \mathsf{o}}$$

No Static DPs for $\mathcal{R}$ at all! So $\mathcal{R}$ is terminating . . . right?

Termination of Higher-Order Term Rewriting: app-lam

Example ( )

$\mathcal{R}$:

$$\frac{\mathsf{app}\ (\mathsf{lam}\ F)\ x \quad \to \quad F\ x}{\mathsf{app} : \mathsf{o} \Rightarrow \mathsf{o} \Rightarrow \mathsf{o}, \qquad \mathsf{lam} : (\mathsf{o} \Rightarrow \mathsf{o}) \Rightarrow \mathsf{o}}$$

No Static DPs for $\mathcal{R}$ at all! So $\mathcal{R}$ is terminating . . . right?

Alas, for $\omega = \mathsf{lam}\ (\lambda x.\mathsf{app}\ x\ x)$ we get

$$\mathsf{app}\ \omega\ \omega\ \to_{\mathcal{R}}\ (\lambda x.\mathsf{app}\ x\ x)\ \omega\ \to_{\beta}\ \mathsf{app}\ \omega\ \omega\ \to_{\mathcal{R}}\ \cdots$$

Termination of Higher-Order Term Rewriting: app-lam

Example (Encoding the Untyped Lambda-Calculus)

$$\mathcal{R}: \quad \frac{\mathsf{app}\ (\mathsf{lam}\ F)\ x \quad \to \quad F\ x}{\mathsf{app} : \mathsf{o} \Rightarrow \mathsf{o} \Rightarrow \mathsf{o}, \qquad \mathsf{lam} : (\mathsf{o} \Rightarrow \mathsf{o}) \Rightarrow \mathsf{o}}$$

No Static DPs for $\mathcal{R}$ at all! So $\mathcal{R}$ is terminating ... right?

Alas, for $\omega = \mathsf{lam}\ (\lambda x.\mathsf{app}\ x\ x)$ we get

$$\mathsf{app}\ \omega\ \omega \ \to_{\mathcal{R}}\ (\lambda x.\mathsf{app}\ x\ x)\ \omega \ \to_{\beta}\ \mathsf{app}\ \omega\ \omega \ \to_{\mathcal{R}}\ \cdots$$

Termination of Higher-Order Term Rewriting: app-lam

Example (Encoding the Untyped Lambda-Calculus)

$$\mathcal{R}: \quad \frac{\mathsf{app}\ (\mathsf{lam}\ F)\ x \quad \to \quad F\ x}{\mathsf{app} : \mathsf{o} \Rightarrow \mathsf{o} \Rightarrow \mathsf{o}, \qquad \mathsf{lam} : (\mathsf{o} \Rightarrow \mathsf{o}) \Rightarrow \mathsf{o}}$$

No Static DPs for $\mathcal{R}$ at all! So $\mathcal{R}$ is terminating ... right?

Alas, for $\omega = \mathsf{lam}\ (\lambda x.\mathsf{app}\ x\ x)$ we get

$$\mathsf{app}\ \omega\ \omega \ \to_{\mathcal{R}}\ (\lambda x.\mathsf{app}\ x\ x)\ \omega \ \to_{\beta}\ \mathsf{app}\ \omega\ \omega \ \to_{\mathcal{R}}\ \cdots$$

Ouch! What went wrong?

Termination of Higher-Order Term Rewriting: app-lam

Example (Encoding the Untyped Lambda-Calculus)

$$\mathcal{R}: \quad \frac{\mathsf{app}\ (\mathsf{lam}\ F)\ x \ \to\ F\ x}{\mathsf{app} : \mathsf{o} \Rightarrow \mathsf{o} \Rightarrow \mathsf{o}, \qquad \mathsf{lam} : (\mathsf{o} \Rightarrow \mathsf{o}) \Rightarrow \mathsf{o}}$$

No Static DPs for $\mathcal{R}$ at all! So $\mathcal{R}$ is terminating ... right?

Alas, for $\omega = \mathsf{lam}\ (\lambda x.\mathsf{app}\ x\ x)$ we get

$$\mathsf{app}\ \omega\ \omega \ \to_{\mathcal{R}}\ (\lambda x.\mathsf{app}\ x\ x)\ \omega \ \to_{\beta}\ \mathsf{app}\ \omega\ \omega \ \to_{\mathcal{R}}\ \cdots$$

Ouch! What went wrong?

The culprit: the constructor lam in the left-hand side

Termination of Higher-Order Term Rewriting: app-lam

Example (Encoding the Untyped Lambda-Calculus)

$$\mathcal{R}: \quad \frac{\mathsf{app}\ (\mathsf{lam}\ F)\ x \quad \to \quad F\ x}{\mathsf{app} : \mathsf{o} \Rightarrow \mathsf{o} \Rightarrow \mathsf{o}, \qquad \mathsf{lam} : (\mathsf{o} \Rightarrow \mathsf{o}) \Rightarrow \mathsf{o}}$$

No Static DPs for $\mathcal{R}$ at all! So $\mathcal{R}$ is terminating ... right?

Alas, for $\omega = \mathsf{lam}\ (\lambda x.\mathsf{app}\ x\ x)$ we get

$$\mathsf{app}\ \omega\ \omega \ \to_{\mathcal{R}}\ (\lambda x.\mathsf{app}\ x\ x)\ \omega \ \to_{\beta}\ \mathsf{app}\ \omega\ \omega \ \to_{\mathcal{R}}\ \cdots$$

Ouch! What went wrong?

The culprit: the constructor lam in the left-hand side

- outwardly of base type

Termination of Higher-Order Term Rewriting: app-lam

Example (Encoding the Untyped Lambda-Calculus)

$$\mathcal{R}: \quad \frac{\mathsf{app}\ (\mathsf{lam}\ F)\ x \quad \to \quad F\ x}{\mathsf{app} : \mathsf{o} \Rightarrow \mathsf{o} \Rightarrow \mathsf{o}, \qquad \mathsf{lam} : (\mathsf{o} \Rightarrow \mathsf{o}) \Rightarrow \mathsf{o}}$$

No Static DPs for $\mathcal{R}$ at all! So $\mathcal{R}$ is terminating ... right?

Alas, for $\omega = \mathsf{lam}\ (\lambda x.\mathsf{app}\ x\ x)$ we get

$$\mathsf{app}\ \omega\ \omega \ \to_{\mathcal{R}}\ (\lambda x.\mathsf{app}\ x\ x)\ \omega \ \to_{\beta}\ \mathsf{app}\ \omega\ \omega \ \to_{\mathcal{R}}\ \cdots$$

Ouch! What went wrong?

The culprit: the constructor lam in the left-hand side

- outwardly of base type
- but inwardly carrying the higher-order variable F as an argument

Termination of Higher-Order Term Rewriting: app-lam

Example (Encoding the Untyped Lambda-Calculus)

$$\mathcal{R}: \quad \frac{\mathsf{app}\ (\mathsf{lam}\ F)\ x \quad \to \quad F\ x}{\mathsf{app} : \mathsf{o} \Rightarrow \mathsf{o} \Rightarrow \mathsf{o}, \qquad \mathsf{lam} : (\mathsf{o} \Rightarrow \mathsf{o}) \Rightarrow \mathsf{o}}$$

No Static DPs for $\mathcal{R}$ at all! So $\mathcal{R}$ is terminating ... right?

Alas, for $\omega = \mathsf{lam}\ (\lambda x.\mathsf{app}\ x\ x)$ we get

$$\mathsf{app}\ \omega\ \omega \ \to_{\mathcal{R}}\ (\lambda x.\mathsf{app}\ x\ x)\ \omega \ \to_{\beta}\ \mathsf{app}\ \omega\ \omega \ \to_{\mathcal{R}}\ \cdots$$

Ouch! What went wrong?

The culprit: the constructor lam in the left-hand side

- outwardly of base type
- but inwardly carrying the higher-order variable F as an argument

So can't we just prohibit Static DPs for such malicious inputs?

Plain Function Passing (PFP)

Original soundness criterion for Static DPs [Kusakari, Isogai, Sakai, Blanqui, *TIS'09*] (adapted to our AFSMs):

Definition (Plain Function Passing (PFP))

A rewrite system $\mathcal{R}$ is Plain Function Passing (PFP)
if for all rules $f\ \ell_1\ \ldots\ \ell_n \to r \in \mathcal{R}$ and for all meta-variables F
of functional type in r, we have $F = \ell_i$ for some i.

Plain Function Passing (PFP)

Original soundness criterion for Static DPs [Kusakari, Isogai, Sakai, Blanqui, *TIS'09*] (adapted to our AFSMs):

Definition (Plain Function Passing (PFP))

A rewrite system $\mathcal{R}$ is Plain Function Passing (PFP)
if for all rules $f\ \ell_1\ \ldots\ \ell_n \to r \in \mathcal{R}$ and for all meta-variables F
of functional type in r, we have $F = \ell_i$ for some i.

Theorem (Static DPs sound for Plain Function Passing $\mathcal{R}$)

Let $\mathcal{R}$ be a rewrite system that is PFP. Then Static DPs are sound for proving termination of $\mathcal{R}$.

Plain Function Passing (PFP)

Original soundness criterion for Static DPs [Kusakari, Isogai, Sakai, Blanqui, *TIS'09*] (adapted to our AFSMs):

Definition (Plain Function Passing (PFP))

A rewrite system $\mathcal{R}$ is Plain Function Passing (PFP)
if for all rules $f\ \ell_1\ \ldots\ \ell_n \to r \in \mathcal{R}$ and for all meta-variables F
of functional type in r, we have $F = \ell_i$ for some i.

Theorem (Static DPs sound for Plain Function Passing $\mathcal{R}$)

Let $\mathcal{R}$ be a rewrite system that is PFP. Then Static DPs are sound for proving termination of $\mathcal{R}$.

Which of the following rules is PFP?

Plain Function Passing (PFP)

Original soundness criterion for Static DPs [Kusakari, Isogai, Sakai, Blanqui, *TIS'09*] (adapted to our AFSMs):

Definition (Plain Function Passing (PFP))

A rewrite system $\mathcal{R}$ is Plain Function Passing (PFP)
if for all rules $f\ \ell_1\ \ldots\ \ell_n \to r \in \mathcal{R}$ and for all meta-variables F
of functional type in r, we have $F = \ell_i$ for some i.

Theorem (Static DPs sound for Plain Function Passing $\mathcal{R}$)

Let $\mathcal{R}$ be a rewrite system that is PFP. Then Static DPs are sound for proving termination of $\mathcal{R}$.

Which of the following rules is PFP?

- append $(\text{cons}\ x\ xs)\ ys \to \text{cons}\ x\ (\text{append}\ xs\ ys)$

Plain Function Passing (PFP)

Original soundness criterion for Static DPs [Kusakari, Isogai, Sakai, Blanqui, *TIS'09*] (adapted to our AFSMs):

Definition (Plain Function Passing (PFP))

A rewrite system $\mathcal{R}$ is Plain Function Passing (PFP)
if for all rules $f\ \ell_1\ \ldots\ \ell_n \to r \in \mathcal{R}$ and for all meta-variables F
of functional type in r, we have $F = \ell_i$ for some i.

Theorem (Static DPs sound for Plain Function Passing $\mathcal{R}$)

Let $\mathcal{R}$ be a rewrite system that is PFP. Then Static DPs are sound for proving termination of $\mathcal{R}$.

Which of the following rules is PFP?

- append $(\text{cons}\ x\ xs)\ ys \to \text{cons}\ x\ (\text{append}\ xs\ ys)$ ✓
- map $F\ (\text{cons}\ x\ xs) \to \text{cons}\ (F\ x)\ (\text{map}\ F\ xs)$

Plain Function Passing (PFP)

Original soundness criterion for Static DPs [Kusakari, Isogai, Sakai, Blanqui, *TIS'09*] (adapted to our AFSMs):

Definition (Plain Function Passing (PFP))

A rewrite system $\mathcal{R}$ is Plain Function Passing (PFP)
if for all rules $f\ \ell_1\ \ldots\ \ell_n \to r \in \mathcal{R}$ and for all meta-variables F
of functional type in r, we have $F = \ell_i$ for some i.

Theorem (Static DPs sound for Plain Function Passing $\mathcal{R}$)

Let $\mathcal{R}$ be a rewrite system that is PFP. Then Static DPs are sound for proving termination of $\mathcal{R}$.

Which of the following rules is PFP?

- append $(\text{cons}\ x\ xs)\ ys \to \text{cons}\ x\ (\text{append}\ xs\ ys)$ ✓
- map $F\ (\text{cons}\ x\ xs) \to \text{cons}\ (F\ x)\ (\text{map}\ F\ xs)$ ✓
- app $(\text{lam}\ F)\ x \to F\ x$

Plain Function Passing (PFP)

Original soundness criterion for Static DPs [Kusakari, Isogai, Sakai, Blanqui, *TIS'09*] (adapted to our AFSMs):

Definition (Plain Function Passing (PFP))

A rewrite system $\mathcal{R}$ is Plain Function Passing (PFP)
if for all rules $f\ \ell_1\ \ldots\ \ell_n \to r \in \mathcal{R}$ and for all meta-variables F
of functional type in r, we have $F = \ell_i$ for some i.

Theorem (Static DPs sound for Plain Function Passing $\mathcal{R}$)

Let $\mathcal{R}$ be a rewrite system that is PFP. Then Static DPs are sound for proving termination of $\mathcal{R}$.

Which of the following rules is PFP?

- append $(\text{cons}\ x\ xs)\ ys \to \text{cons}\ x\ (\text{append}\ xs\ ys)$ ✓
- map $F\ (\text{cons}\ x\ xs) \to \text{cons}\ (F\ x)\ (\text{map}\ F\ xs)$ ✓
- app $(\text{lam}\ F)\ x \to F\ x$ ✗

Plain Function Passing (PFP)

Original soundness criterion for Static DPs [Kusakari, Isogai, Sakai, Blanqui, *TIS'09*] (adapted to our AFSMs):

Definition (Plain Function Passing (PFP))

A rewrite system $\mathcal{R}$ is Plain Function Passing (PFP)
if for all rules $f\ \ell_1\ \ldots\ \ell_n \to r \in \mathcal{R}$ and for all meta-variables F
of functional type in r, we have $F = \ell_i$ for some i.

Theorem (Static DPs sound for Plain Function Passing $\mathcal{R}$)

Let $\mathcal{R}$ be a rewrite system that is PFP. Then Static DPs are sound for proving termination of $\mathcal{R}$.

Which of the following rules is PFP?

- append $(\mathsf{cons}\ x\ xs)\ ys \to \mathsf{cons}\ x\ (\mathsf{append}\ xs\ ys)$ ✓
- $\mathsf{map}\ F\ (\mathsf{cons}\ x\ xs) \to \mathsf{cons}\ (F\ x)\ (\mathsf{map}\ F\ xs)$ ✓
- $\mathsf{app}\ (\mathsf{lam}\ F)\ x \to F\ x$ ✗

PFP improved in [Suzuki, Kusakari, Blanqui, *IPSJ TOP'11*].

Termination of Higher-Order Term Rewriting: ord-rec

Termination of Higher-Order Term Rewriting: ord-rec

Example (Ordinal recursion; TPDB: `Mixed_HO_10/ordrec.xml`)

$$\begin{array}{rcl} \mathsf{rec}\ 0\ K\ F\ G & \to & K \\ \mathsf{rec}\ (\mathsf{s}\ X)\ K\ F\ G & \to & F\ X\ (\mathsf{rec}\ X\ K\ F\ G) \\ \mathsf{rec}\ (\mathsf{lim}\ H)\ K\ F\ G & \to & G\ H\ (\lambda m.\mathsf{rec}\ (H\ m)\ K\ F\ G) \end{array}$$

Termination of Higher-Order Term Rewriting: ord-rec

Example (Ordinal recursion; TPDB: `Mixed_HO_10/ordrec.xml`)

$$\begin{array}{rcl} \mathsf{rec}\ 0\ K\ F\ G & \to & K \\ \mathsf{rec}\ (\mathsf{s}\ X)\ K\ F\ G & \to & F\ X\ (\mathsf{rec}\ X\ K\ F\ G) \\ \mathsf{rec}\ (\mathsf{lim}\ H)\ K\ F\ G & \to & G\ H\ (\lambda m.\mathsf{rec}\ (H\ m)\ K\ F\ G) \end{array}$$

- Ordinals: $0 : \mathsf{ord}$, $\mathsf{s} : \mathsf{ord} \Rightarrow \mathsf{ord}$, $\mathsf{lim} : (\mathsf{nat} \Rightarrow \mathsf{ord}) \Rightarrow \mathsf{ord}$

Termination of Higher-Order Term Rewriting: ord-rec

Example (Ordinal recursion; TPDB: `Mixed_HO_10/ordrec.xml`)

$$\begin{array}{rcl} \mathsf{rec}\ 0\ K\ F\ G & \to & K \\ \mathsf{rec}\ (\mathsf{s}\ X)\ K\ F\ G & \to & F\ X\ (\mathsf{rec}\ X\ K\ F\ G) \\ \mathsf{rec}\ (\mathsf{lim}\ H)\ K\ F\ G & \to & G\ H\ (\lambda m.\mathsf{rec}\ (H\ m)\ K\ F\ G) \end{array}$$

- Ordinals: $0 : \mathsf{ord}$, $\mathsf{s} : \mathsf{ord} \Rightarrow \mathsf{ord}$, $\mathsf{lim} : (\mathsf{nat} \Rightarrow \mathsf{ord}) \Rightarrow \mathsf{ord}$
- Natural numbers: $\mathsf{zero} : \mathsf{nat}$, $\mathsf{succ} : \mathsf{nat} \Rightarrow \mathsf{nat}$

Termination of Higher-Order Term Rewriting: ord-rec

Example (Ordinal recursion; TPDB: `Mixed_HO_10/ordrec.xml`)

$$\begin{array}{rcl} \mathsf{rec}\ 0\ K\ F\ G & \to & K \\ \mathsf{rec}\ (\mathsf{s}\ X)\ K\ F\ G & \to & F\ X\ (\mathsf{rec}\ X\ K\ F\ G) \\ \mathsf{rec}\ (\mathsf{lim}\ H)\ K\ F\ G & \to & G\ H\ (\lambda m.\mathsf{rec}\ (H\ m)\ K\ F\ G) \end{array}$$

- Ordinals: $0 : \mathsf{ord}$, $\mathsf{s} : \mathsf{ord} \Rightarrow \mathsf{ord}$, $\mathsf{lim} : (\mathsf{nat} \Rightarrow \mathsf{ord}) \Rightarrow \mathsf{ord}$
- Natural numbers: $\mathsf{zero} : \mathsf{nat}$, $\mathsf{succ} : \mathsf{nat} \Rightarrow \mathsf{nat}$
- $\mathsf{rec} : \mathsf{ord} \Rightarrow \mathsf{nat} \Rightarrow (\mathsf{ord} \Rightarrow \mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow ((\mathsf{nat} \Rightarrow \mathsf{ord}) \Rightarrow (\mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{nat}$

Termination of Higher-Order Term Rewriting: ord-rec

Example (Ordinal recursion; TPDB: `Mixed_HO_10/ordrec.xml`)

$$\begin{array}{rcl} \mathsf{rec}\ 0\ K\ F\ G & \to & K \\ \mathsf{rec}\ (\mathsf{s}\ X)\ K\ F\ G & \to & F\ X\ (\mathsf{rec}\ X\ K\ F\ G) \\ \mathsf{rec}\ (\mathsf{lim}\ H)\ K\ F\ G & \to & G\ H\ (\lambda m.\mathsf{rec}\ (H\ m)\ K\ F\ G) \end{array}$$

- Ordinals: $0 : \mathsf{ord}$, $\mathsf{s} : \mathsf{ord} \Rightarrow \mathsf{ord}$, $\mathsf{lim} : (\mathsf{nat} \Rightarrow \mathsf{ord}) \Rightarrow \mathsf{ord}$
- Natural numbers: $\mathsf{zero} : \mathsf{nat}$, $\mathsf{succ} : \mathsf{nat} \Rightarrow \mathsf{nat}$
- $\mathsf{rec} : \mathsf{ord} \Rightarrow \mathsf{nat} \Rightarrow (\mathsf{ord} \Rightarrow \mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow ((\mathsf{nat} \Rightarrow \mathsf{ord}) \Rightarrow (\mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{nat}$

That's not PFP!

Termination of Higher-Order Term Rewriting: ord-rec

Example (Ordinal recursion; TPDB: `Mixed_HO_10/ordrec.xml`)

$$\begin{array}{rcl} \mathsf{rec}\ 0\ K\ F\ G & \to & K \\ \mathsf{rec}\ (\mathsf{s}\ X)\ K\ F\ G & \to & F\ X\ (\mathsf{rec}\ X\ K\ F\ G) \\ \mathsf{rec}\ (\mathsf{lim}\ H)\ K\ F\ G & \to & G\ H\ (\lambda m.\mathsf{rec}\ (H\ m)\ K\ F\ G) \end{array}$$

- Ordinals: $0 : \mathsf{ord}$, $\mathsf{s} : \mathsf{ord} \Rightarrow \mathsf{ord}$, $\mathsf{lim} : (\mathsf{nat} \Rightarrow \mathsf{ord}) \Rightarrow \mathsf{ord}$
- Natural numbers: $\mathsf{zero} : \mathsf{nat}$, $\mathsf{succ} : \mathsf{nat} \Rightarrow \mathsf{nat}$
- $\mathsf{rec} : \mathsf{ord} \Rightarrow \mathsf{nat} \Rightarrow (\mathsf{ord} \Rightarrow \mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow ((\mathsf{nat} \Rightarrow \mathsf{ord}) \Rightarrow (\mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{nat}$

That's not PFP! – But Static DPs may still be applicable . . . (This Work)

Termination of Higher-Order Term Rewriting: ord-rec

Example (Ordinal recursion; TPDB: `Mixed_HO_10/ordrec.xml`)

$$\begin{array}{rcl} \mathsf{rec}\ 0\ K\ F\ G & \to & K \\ \mathsf{rec}\ (\mathsf{s}\ X)\ K\ F\ G & \to & F\ X\ (\mathsf{rec}\ X\ K\ F\ G) \\ \mathsf{rec}\ (\mathsf{lim}\ H)\ K\ F\ G & \to & G\ H\ (\lambda m.\mathsf{rec}\ (H\ m)\ K\ F\ G) \end{array}$$

- Ordinals: $0 : \mathsf{ord}$, $\mathsf{s} : \mathsf{ord} \Rightarrow \mathsf{ord}$, $\mathsf{lim} : (\mathsf{nat} \Rightarrow \mathsf{ord}) \Rightarrow \mathsf{ord}$
- Natural numbers: $\mathsf{zero} : \mathsf{nat}$, $\mathsf{succ} : \mathsf{nat} \Rightarrow \mathsf{nat}$
- $\mathsf{rec} : \mathsf{ord} \Rightarrow \mathsf{nat} \Rightarrow (\mathsf{ord} \Rightarrow \mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow ((\mathsf{nat} \Rightarrow \mathsf{ord}) \Rightarrow (\mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{nat}$

That's not PFP! – But Static DPs may still be applicable . . . (This Work)

Idea: decrease in a **well-founded order** on types when we apply an F taken out of a left-hand side

Termination of Higher-Order Term Rewriting: ord-rec

Example (Ordinal recursion; TPDB: `Mixed_HO_10/ordrec.xml`)

$$\begin{array}{rcl} \mathsf{rec}\ 0\ K\ F\ G & \to & K \\ \mathsf{rec}\ (\mathsf{s}\ X)\ K\ F\ G & \to & F\ X\ (\mathsf{rec}\ X\ K\ F\ G) \\ \mathsf{rec}\ (\mathsf{lim}\ H)\ K\ F\ G & \to & G\ H\ (\lambda m.\mathsf{rec}\ (H\ m)\ K\ F\ G) \end{array}$$

- Ordinals: $0 : \mathsf{ord}$, $\mathsf{s} : \mathsf{ord} \Rightarrow \mathsf{ord}$, $\mathsf{lim} : (\mathsf{nat} \Rightarrow \mathsf{ord}) \Rightarrow \mathsf{ord}$
- Natural numbers: $\mathsf{zero} : \mathsf{nat}$, $\mathsf{succ} : \mathsf{nat} \Rightarrow \mathsf{nat}$
- $\mathsf{rec} : \mathsf{ord} \Rightarrow \mathsf{nat} \Rightarrow (\mathsf{ord} \Rightarrow \mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow ((\mathsf{nat} \Rightarrow \mathsf{ord}) \Rightarrow (\mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{nat}$

That's not PFP! – But Static DPs may still be applicable . . . (This Work)

Idea: decrease in a **well-founded order** on types when we apply an F taken out of a left-hand side

→ Accessibility, see General Schema [Blanqui, Jouannaud, Okada, *TCS'02*], Computability Path Ordering [Blanqui, Jouannaud, Rubio, *LMCS'15*], and Computability Closure [Blanqui, *TCS'16*]

PFP with Accessibility: Accessible Function Passing

Example (Ordinal recursion; TPDB: `Mixed_HO_10/ordrec.xml`)

$$\begin{array}{rcl} \mathsf{rec}\ 0\ K\ F\ G & \to & K \\ \mathsf{rec}\ (\mathsf{s}\ X)\ K\ F\ G & \to & F\ X\ (\mathsf{rec}\ X\ K\ F\ G) \\ \mathsf{rec}\ (\mathsf{lim}\ H)\ K\ F\ G & \to & G\ H\ (\lambda m.\mathsf{rec}\ (H\ m)\ K\ F\ G) \end{array}$$

- Ordinals: $0 : \mathsf{ord}$, $\mathsf{s} : \mathsf{ord} \Rightarrow \mathsf{ord}$, $\mathsf{lim} : (\mathsf{nat} \Rightarrow \mathsf{ord}) \Rightarrow \mathsf{ord}$
- Natural numbers: $\mathsf{zero} : \mathsf{nat}$, $\mathsf{succ} : \mathsf{nat} \Rightarrow \mathsf{nat}$
- $\mathsf{rec} : \mathsf{ord} \Rightarrow \mathsf{nat} \Rightarrow (\mathsf{ord} \Rightarrow \mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow ((\mathsf{nat} \Rightarrow \mathsf{ord}) \Rightarrow (\mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{nat}$

PFP with Accessibility: Accessible Function Passing

Example (Ordinal recursion; TPDB: `Mixed_HO_10/ordrec.xml`)

$$\begin{array}{rcl} \mathsf{rec}\ 0\ K\ F\ G & \to & K \\ \mathsf{rec}\ (\mathsf{s}\ X)\ K\ F\ G & \to & F\ X\ (\mathsf{rec}\ X\ K\ F\ G) \\ \mathsf{rec}\ (\mathsf{lim}\ H)\ K\ F\ G & \to & G\ H\ (\lambda m.\mathsf{rec}\ (H\ m)\ K\ F\ G) \end{array}$$

- Ordinals: $0 : \mathsf{ord}$, $\mathsf{s} : \mathsf{ord} \Rightarrow \mathsf{ord}$, $\mathsf{lim} : (\mathsf{nat} \Rightarrow \mathsf{ord}) \Rightarrow \mathsf{ord}$
- Natural numbers: $\mathsf{zero} : \mathsf{nat}$, $\mathsf{succ} : \mathsf{nat} \Rightarrow \mathsf{nat}$
- $\mathsf{rec} : \mathsf{ord} \Rightarrow \mathsf{nat} \Rightarrow (\mathsf{ord} \Rightarrow \mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow ((\mathsf{nat} \Rightarrow \mathsf{ord}) \Rightarrow (\mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{nat}$

Issue: Is it ok to take $H : \mathsf{nat} \Rightarrow \mathsf{ord}$ out from below $\mathsf{lim}\ H : \mathsf{ord}$?

PFP with Accessibility: Accessible Function Passing

Example (Ordinal recursion; TPDB: `Mixed_HO_10/ordrec.xml`)

$$\begin{array}{rcl} \mathsf{rec}\ 0\ K\ F\ G & \to & K \\ \mathsf{rec}\ (\mathsf{s}\ X)\ K\ F\ G & \to & F\ X\ (\mathsf{rec}\ X\ K\ F\ G) \\ \mathsf{rec}\ (\mathsf{lim}\ H)\ K\ F\ G & \to & G\ H\ (\lambda m.\mathsf{rec}\ (H\ m)\ K\ F\ G) \end{array}$$

- Ordinals: $0 : \mathsf{ord}$, $\mathsf{s} : \mathsf{ord} \Rightarrow \mathsf{ord}$, $\mathsf{lim} : (\mathsf{nat} \Rightarrow \mathsf{ord}) \Rightarrow \mathsf{ord}$
- Natural numbers: $\mathsf{zero} : \mathsf{nat}$, $\mathsf{succ} : \mathsf{nat} \Rightarrow \mathsf{nat}$
- $\mathsf{rec} : \mathsf{ord} \Rightarrow \mathsf{nat} \Rightarrow (\mathsf{ord} \Rightarrow \mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow ((\mathsf{nat} \Rightarrow \mathsf{ord}) \Rightarrow (\mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{nat}$

Issue: Is it ok to take $H : \mathsf{nat} \Rightarrow \mathsf{ord}$ out from below $\mathsf{lim}\ H : \mathsf{ord}$?

Use $\mathsf{ord} \succ^{\mathcal{S}} \mathsf{nat}$.

PFP with Accessibility: Accessible Function Passing

Example (Ordinal recursion; TPDB: `Mixed_HO_10/ordrec.xml`)

$$\begin{array}{rcl} \mathsf{rec}\ 0\ K\ F\ G & \to & K \\ \mathsf{rec}\ (\mathsf{s}\ X)\ K\ F\ G & \to & F\ X\ (\mathsf{rec}\ X\ K\ F\ G) \\ \mathsf{rec}\ (\mathsf{lim}\ H)\ K\ F\ G & \to & G\ H\ (\lambda m.\mathsf{rec}\ (H\ m)\ K\ F\ G) \end{array}$$

- Ordinals: $0 : \mathsf{ord}$, $\mathsf{s} : \mathsf{ord} \Rightarrow \mathsf{ord}$, $\mathsf{lim} : (\mathsf{nat} \Rightarrow \mathsf{ord}) \Rightarrow \mathsf{ord}$
- Natural numbers: $\mathsf{zero} : \mathsf{nat}$, $\mathsf{succ} : \mathsf{nat} \Rightarrow \mathsf{nat}$
- $\mathsf{rec} : \mathsf{ord} \Rightarrow \mathsf{nat} \Rightarrow (\mathsf{ord} \Rightarrow \mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow ((\mathsf{nat} \Rightarrow \mathsf{ord}) \Rightarrow (\mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{nat}$

Issue: Is it ok to take $H : \mathsf{nat} \Rightarrow \mathsf{ord}$ out from below $\mathsf{lim}\ H : \mathsf{ord}$?

Use $\mathsf{ord} \succ^{\mathcal{S}} \mathsf{nat}$. Then $\mathsf{ord} \succeq^{\mathcal{S}}_{+} \mathsf{nat} \Rightarrow \mathsf{ord}$ because

PFP with Accessibility: Accessible Function Passing

Example (Ordinal recursion; TPDB: `Mixed_HO_10/ordrec.xml`)

$$\begin{array}{rcl} \mathsf{rec}\ 0\ K\ F\ G & \to & K \\ \mathsf{rec}\ (\mathsf{s}\ X)\ K\ F\ G & \to & F\ X\ (\mathsf{rec}\ X\ K\ F\ G) \\ \mathsf{rec}\ (\mathsf{lim}\ H)\ K\ F\ G & \to & G\ H\ (\lambda m.\mathsf{rec}\ (H\ m)\ K\ F\ G) \end{array}$$

- Ordinals: $0 : \mathsf{ord}$, $\mathsf{s} : \mathsf{ord} \Rightarrow \mathsf{ord}$, $\mathsf{lim} : (\mathsf{nat} \Rightarrow \mathsf{ord}) \Rightarrow \mathsf{ord}$
- Natural numbers: $\mathsf{zero} : \mathsf{nat}$, $\mathsf{succ} : \mathsf{nat} \Rightarrow \mathsf{nat}$
- $\mathsf{rec} : \mathsf{ord} \Rightarrow \mathsf{nat} \Rightarrow (\mathsf{ord} \Rightarrow \mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow ((\mathsf{nat} \Rightarrow \mathsf{ord}) \Rightarrow (\mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{nat}$

Issue: Is it ok to take $H : \mathsf{nat} \Rightarrow \mathsf{ord}$ out from below $\mathsf{lim}\ H : \mathsf{ord}$?

Use $\mathsf{ord} \succ^{\mathcal{S}} \mathsf{nat}$. Then $\mathsf{ord} \succeq^{\mathcal{S}}_{+} \mathsf{nat} \Rightarrow \mathsf{ord}$ because

- $\mathsf{ord} \succeq^{\mathcal{S}} \mathsf{ord}$

PFP with Accessibility: Accessible Function Passing

Example (Ordinal recursion; TPDB: `Mixed_HO_10/ordrec.xml`)

$$\begin{array}{rcl} \mathsf{rec}\ 0\ K\ F\ G & \to & K \\ \mathsf{rec}\ (\mathsf{s}\ X)\ K\ F\ G & \to & F\ X\ (\mathsf{rec}\ X\ K\ F\ G) \\ \mathsf{rec}\ (\mathsf{lim}\ H)\ K\ F\ G & \to & G\ H\ (\lambda m.\mathsf{rec}\ (H\ m)\ K\ F\ G) \end{array}$$

- Ordinals: $0 : \mathsf{ord}$, $\mathsf{s} : \mathsf{ord} \Rightarrow \mathsf{ord}$, $\mathsf{lim} : (\mathsf{nat} \Rightarrow \mathsf{ord}) \Rightarrow \mathsf{ord}$
- Natural numbers: $\mathsf{zero} : \mathsf{nat}$, $\mathsf{succ} : \mathsf{nat} \Rightarrow \mathsf{nat}$
- $\mathsf{rec} : \mathsf{ord} \Rightarrow \mathsf{nat} \Rightarrow (\mathsf{ord} \Rightarrow \mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow ((\mathsf{nat} \Rightarrow \mathsf{ord}) \Rightarrow (\mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{nat}$

Issue: Is it ok to take $H : \mathsf{nat} \Rightarrow \mathsf{ord}$ out from below $\mathsf{lim}\ H : \mathsf{ord}$?

Use $\mathsf{ord} \succ^{\mathcal{S}} \mathsf{nat}$. Then $\mathsf{ord} \succeq^{\mathcal{S}}_{+} \mathsf{nat} \Rightarrow \mathsf{ord}$ because

- $\mathsf{ord} \succeq^{\mathcal{S}} \mathsf{ord}$
- $\mathsf{ord} \succ^{\mathcal{S}}_{-} \mathsf{nat}$

PFP with Accessibility: Accessible Function Passing

Example (Ordinal recursion; TPDB: `Mixed_HO_10/ordrec.xml`)

$$\begin{array}{rcl} \mathsf{rec}\ 0\ K\ F\ G & \to & K \\ \mathsf{rec}\ (\mathsf{s}\ X)\ K\ F\ G & \to & F\ X\ (\mathsf{rec}\ X\ K\ F\ G) \\ \mathsf{rec}\ (\mathsf{lim}\ H)\ K\ F\ G & \to & G\ H\ (\lambda m.\mathsf{rec}\ (H\ m)\ K\ F\ G) \end{array}$$

- Ordinals: $0 : \mathsf{ord}$, $\mathsf{s} : \mathsf{ord} \Rightarrow \mathsf{ord}$, $\mathsf{lim} : (\mathsf{nat} \Rightarrow \mathsf{ord}) \Rightarrow \mathsf{ord}$
- Natural numbers: $\mathsf{zero} : \mathsf{nat}$, $\mathsf{succ} : \mathsf{nat} \Rightarrow \mathsf{nat}$
- $\mathsf{rec} : \mathsf{ord} \Rightarrow \mathsf{nat} \Rightarrow (\mathsf{ord} \Rightarrow \mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow ((\mathsf{nat} \Rightarrow \mathsf{ord}) \Rightarrow (\mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{nat}$

Issue: Is it ok to take $H : \mathsf{nat} \Rightarrow \mathsf{ord}$ out from below $\mathsf{lim}\ H : \mathsf{ord}$?

Use $\mathsf{ord} \succ^{\mathcal{S}} \mathsf{nat}$. Then $\mathsf{ord} \succeq^{\mathcal{S}}_{+} \mathsf{nat} \Rightarrow \mathsf{ord}$ because

- $\mathsf{ord} \succeq^{\mathcal{S}} \mathsf{ord}$
- $\mathsf{ord} \succ^{\mathcal{S}}_{-} \mathsf{nat}$

$\Rightarrow 1 \in Acc(\mathsf{lim})$

PFP with Accessibility: Accessible Function Passing

Example (Ordinal recursion; TPDB: `Mixed_HO_10/ordrec.xml`)

$$\begin{array}{rcl} \mathsf{rec}\ 0\ K\ F\ G & \to & K \\ \mathsf{rec}\ (\mathsf{s}\ X)\ K\ F\ G & \to & F\ X\ (\mathsf{rec}\ X\ K\ F\ G) \\ \mathsf{rec}\ (\mathsf{lim}\ H)\ K\ F\ G & \to & G\ H\ (\lambda m.\mathsf{rec}\ (H\ m)\ K\ F\ G) \end{array}$$

- Ordinals: $0 : \mathsf{ord}$, $\mathsf{s} : \mathsf{ord} \Rightarrow \mathsf{ord}$, $\mathsf{lim} : (\mathsf{nat} \Rightarrow \mathsf{ord}) \Rightarrow \mathsf{ord}$
- Natural numbers: $\mathsf{zero} : \mathsf{nat}$, $\mathsf{succ} : \mathsf{nat} \Rightarrow \mathsf{nat}$
- $\mathsf{rec} : \mathsf{ord} \Rightarrow \mathsf{nat} \Rightarrow (\mathsf{ord} \Rightarrow \mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow ((\mathsf{nat} \Rightarrow \mathsf{ord}) \Rightarrow (\mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{nat}$

Issue: Is it ok to take $H : \mathsf{nat} \Rightarrow \mathsf{ord}$ out from below $\mathsf{lim}\ H : \mathsf{ord}$?

Use $\mathsf{ord} \succ^{\mathcal{S}} \mathsf{nat}$. Then $\mathsf{ord} \succeq^{\mathcal{S}}_{+} \mathsf{nat} \Rightarrow \mathsf{ord}$ because

- $\mathsf{ord} \succeq^{\mathcal{S}} \mathsf{ord}$
- $\mathsf{ord} \succ^{\mathcal{S}}_{-} \mathsf{nat}$

$\Rightarrow 1 \in Acc(\mathsf{lim})$ $\Rightarrow$ Static DPs sound here!

PFP with Accessibility: Accessible Function Passing

Example (Ordinal recursion; TPDB: `Mixed_HO_10/ordrec.xml`)

$$\begin{array}{rcl} \mathsf{rec}\ 0\ K\ F\ G & \to & K \\ \mathsf{rec}\ (\mathsf{s}\ X)\ K\ F\ G & \to & F\ X\ (\mathsf{rec}\ X\ K\ F\ G) \\ \mathsf{rec}\ (\mathsf{lim}\ H)\ K\ F\ G & \to & G\ H\ (\lambda m.\mathsf{rec}\ (H\ m)\ K\ F\ G) \end{array}$$

- Ordinals: $0 : \mathsf{ord}$, $\mathsf{s} : \mathsf{ord} \Rightarrow \mathsf{ord}$, $\mathsf{lim} : (\mathsf{nat} \Rightarrow \mathsf{ord}) \Rightarrow \mathsf{ord}$
- Natural numbers: $\mathsf{zero} : \mathsf{nat}$, $\mathsf{succ} : \mathsf{nat} \Rightarrow \mathsf{nat}$
- $\mathsf{rec} : \mathsf{ord} \Rightarrow \mathsf{nat} \Rightarrow (\mathsf{ord} \Rightarrow \mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow ((\mathsf{nat} \Rightarrow \mathsf{ord}) \Rightarrow (\mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{nat}$

Issue: Is it ok to take $H : \mathsf{nat} \Rightarrow \mathsf{ord}$ out from below $\mathsf{lim}\ H : \mathsf{ord}$?

Use $\mathsf{ord} \succ^{\mathcal{S}} \mathsf{nat}$. Then $\mathsf{ord} \succeq^{\mathcal{S}}_{+} \mathsf{nat} \Rightarrow \mathsf{ord}$ because

- $\mathsf{ord} \succeq^{\mathcal{S}} \mathsf{ord}$
- $\mathsf{ord} \succ^{\mathcal{S}}_{-} \mathsf{nat}$

$\Rightarrow 1 \in Acc(\mathsf{lim})$ $\Rightarrow$ Static DPs sound here!

Related criterion (incomparable): [Blanqui, *WST'06*]

Static DPs and the Computable Subterm Criterion

Example (Static DPs for ord-rec)

$$\begin{aligned} \mathsf{rec}^\sharp\ (\mathsf{s}\ X)\ K\ F\ G &\Rightarrow \mathsf{rec}^\sharp\ X\ K\ F\ G \\ \mathsf{rec}^\sharp\ (\mathsf{lim}\ H)\ K\ F\ G &\Rightarrow \mathsf{rec}^\sharp\ (H\ M)\ K\ F\ G \end{aligned}$$

Static DPs and the Computable Subterm Criterion

Example (Static DPs for ord-rec)

$$\begin{array}{rcl} \mathsf{rec}^\sharp\ (\mathsf{s}\ X)\ K\ F\ G & \Rightarrow & \mathsf{rec}^\sharp\ X\ K\ F\ G \\ \mathsf{rec}^\sharp\ (\mathsf{lim}\ H)\ K\ F\ G & \Rightarrow & \mathsf{rec}^\sharp\ (H\ M)\ K\ F\ G \end{array}$$

Candidate for Subterm Criterion: $\nu(\mathsf{rec}^\sharp) = 1$.

.

Static DPs and the Computable Subterm Criterion

Example (Static DPs for ord-rec)

$$\begin{array}{rcl} \mathsf{s}\ X & \rhd & X \\ \mathsf{lim}\ H & \rhd & H\ M \end{array}$$

Candidate for Subterm Criterion: $\nu(\mathsf{rec}^\sharp) = 1$.

- $\mathsf{s}\ X \quad \rhd \quad X$

Static DPs and the Computable Subterm Criterion

Example (Static DPs for ord-rec)

$$\begin{array}{rcl} \mathsf{s}\ X & \rhd & X \\ \mathsf{lim}\ H & \rhd & H\ M \end{array}$$

Candidate for Subterm Criterion: $\nu(\mathsf{rec}^\sharp) = 1$.

- $\mathsf{s}\ X \quad \rhd \quad X$ ✓
- $\mathsf{lim}\ H \quad \rhd \quad H\ M$

.

Static DPs and the Computable Subterm Criterion

Example (Static DPs for ord-rec)

$$\begin{array}{rcl} \mathsf{s}\ X & \rhd & X \\ \lim\ H & \rhd & H\ M \end{array}$$

Candidate for Subterm Criterion: $\nu(\mathsf{rec}^\sharp) = 1$.

- $\mathsf{s}\ X \quad \rhd \quad X$ ✓
- $\lim\ H \quad \rhd \quad H\ M$ ✗

Static DPs and the Computable Subterm Criterion

Example (Static DPs for ord-rec)

$$\begin{array}{rcl} \mathsf{s}\ X & \rhd & X \\ \mathsf{lim}\ H & \rhd & H\ M \end{array}$$

Candidate for Subterm Criterion: $\nu(\mathsf{rec}^\sharp) = 1$.

- $\mathsf{s}\ X \quad \rhd \quad X$ ✓
- $\mathsf{lim}\ H \quad \rhd \quad H\ M$ ✗

Recall minimal chains:

$$s_1 \Rrightarrow t_1 \to_{\mathcal{R}}^* s_2 \Rrightarrow t_2 \to_{\mathcal{R}}^* s_3 \Rrightarrow t_3 \to_{\mathcal{R}}^* \cdots$$

All t_i are $\mathcal{R}$-**terminating**.

.

Static DPs and the Computable Subterm Criterion

Example (Static DPs for ord-rec)

$$\begin{array}{rcl} \mathsf{s}\ X & \rhd & X \\ \lim\ H & \rhd & H\ M \end{array}$$

Candidate for Subterm Criterion: $\nu(\mathsf{rec}^\sharp) = 1$.

- $\mathsf{s}\ X \quad \rhd \quad X$ ✓
- $\lim\ H \quad \rhd \quad H\ M$ ✗

Recall minimal chains:

$$s_1 \Rrightarrow t_1 \to_{\mathcal{R}}^* s_2 \Rrightarrow t_2 \to_{\mathcal{R}}^* s_3 \Rrightarrow t_3 \to_{\mathcal{R}}^* \cdots$$

All t_i are $\mathcal{R}$-**terminating**.

Soundness proof of Static DPs (when applicable) gives stronger property:
All t_i are $\mathcal{R}$-**computable** [Tait, *JSL'67*].

Static DPs and the Computable Subterm Criterion

Example (Static DPs for ord-rec)

$$\begin{array}{rcl} \mathsf{s}\ X & \rhd & X \\ \lim\ H & \rhd & H\ M \end{array}$$

Candidate for Subterm Criterion: $\nu(\mathsf{rec}^\sharp) = 1$.

- $\mathsf{s}\ X \quad \rhd \quad X$ ✓
- $\lim\ H \quad \rhd \quad H\ M$ ✗

Recall minimal chains:

$$s_1 \Rrightarrow t_1 \to_{\mathcal{R}}^* s_2 \Rrightarrow t_2 \to_{\mathcal{R}}^* s_3 \Rrightarrow t_3 \to_{\mathcal{R}}^* \cdots$$

All t_i are $\mathcal{R}$-**terminating**.

Soundness proof of Static DPs (when applicable) gives stronger property:
All t_i are $\mathcal{R}$-**computable** [Tait, *JSL'67*].

$\Rightarrow$ may ignore ($\mathcal{R}$-computable) argument M of H here:

Static DPs and the Computable Subterm Criterion

Example (Static DPs for ord-rec)

$$\begin{array}{rcl} \mathsf{s}\ X & \rhd & X \\ \mathsf{lim}\ H & \rhd & H\ M \end{array}$$

Candidate for Subterm Criterion: $\nu(\mathsf{rec}^\sharp) = 1$.

- $\mathsf{s}\ X \quad \rhd \quad X$ ✓
- $\mathsf{lim}\ H \quad \rhd \quad H\ M$ ✗

Recall minimal chains:

$$s_1 \Rrightarrow t_1 \to_{\mathcal{R}}^* s_2 \Rrightarrow t_2 \to_{\mathcal{R}}^* s_3 \Rrightarrow t_3 \to_{\mathcal{R}}^* \cdots$$

All t_i are $\mathcal{R}$-**terminating**.

Soundness proof of Static DPs (when applicable) gives stronger property:
All t_i are $\mathcal{R}$-**computable** [Tait, *JSL'67*].

$\Rightarrow$ may ignore ($\mathcal{R}$-computable) argument M of H here:

- $\mathsf{lim}\ H \quad \rhd \quad H$

Static DPs and the Computable Subterm Criterion

Example (Static DPs for ord-rec)

$$\begin{array}{rcl} \mathsf{s}\ X & \rhd & X \\ \mathsf{lim}\ H & \rhd & H\ M \end{array}$$

Candidate for Subterm Criterion: $\nu(\mathsf{rec}^\sharp) = 1$.

- $\mathsf{s}\ X \quad \rhd \quad X$ ✓
- $\mathsf{lim}\ H \quad \rhd \quad H\ M$ ✗

Recall minimal chains:

$$s_1 \Rrightarrow t_1 \to_{\mathcal{R}}^* s_2 \Rrightarrow t_2 \to_{\mathcal{R}}^* s_3 \Rrightarrow t_3 \to_{\mathcal{R}}^* \cdots$$

All t_i are $\mathcal{R}$-**terminating**.

Soundness proof of Static DPs (when applicable) gives stronger property:
All t_i are $\mathcal{R}$-**computable** [Tait, *JSL'67*].

$\Rightarrow$ may ignore ($\mathcal{R}$-computable) argument M of H here:

- $\mathsf{lim}\ H \quad \rhd \quad H$ ✓

Static DPs and the Computable Subterm Criterion

Example (Static DPs for ord-rec)

$$\begin{array}{rcl} \mathsf{s}\ X & \triangleright & X \\ \mathsf{lim}\ H & \triangleright & H\ M \end{array}$$

Candidate for Subterm Criterion: $\nu(\mathsf{rec}^\sharp) = 1$.

- $\mathsf{s}\ X \quad \triangleright \quad X$ ✓
- $\mathsf{lim}\ H \quad \triangleright \quad H\ M$ ✗

Recall minimal chains:

$$s_1 \Rrightarrow t_1 \to_{\mathcal{R}}^* s_2 \Rrightarrow t_2 \to_{\mathcal{R}}^* s_3 \Rrightarrow t_3 \to_{\mathcal{R}}^* \cdots$$

All t_i are $\mathcal{R}$-**terminating**.

Soundness proof of Static DPs (when applicable) gives stronger property:
All t_i are $\mathcal{R}$-**computable** [Tait, *JSL'67*].

$\Rightarrow$ may ignore ($\mathcal{R}$-computable) argument M of H here:

- $\mathsf{lim}\ H \quad \triangleright \quad H$ ✓

$\Rightarrow$ ord-rec is terminating by **Computable Subterm Criterion**

Conclusion

Here:

- Make Static DPs applicable to larger class of higher-order systems
- New class of infinite dependency chains to analyse: $\mathcal{R}$-computable chains
- New technique for Static DPs: Computable Subterm Criterion

Conclusion

Here:

- Make Static DPs applicable to larger class of higher-order systems
- New class of infinite dependency chains to analyse: $\mathcal{R}$-computable chains
- New technique for Static DPs: Computable Subterm Criterion

Further extensions available:

https://arxiv.org/abs/1805.09390

- Unified framework integrating static and dynamic DPs (dynamic DPs include DPs for calls to meta-variables of functional type)
- Many additional termination proving techniques as DP processors
- Completeness results wrt non-termination for static and dynamic DPs

Bonus slide: Plain Function Passing, v2

PFP: Must the F really be direct arguments on left-hand side?

Bonus slide: Plain Function Passing, v2

PFP: Must the F really be direct arguments on left-hand side?

Example (deriv)

$$\frac{\text{deriv } (\lambda x.\text{sin } F[x])\ Y \to \text{times } (\text{deriv } (\lambda x.F[x])\ Y)\ (\text{cos } F[Y])}{\begin{array}{c}\text{sin}, \text{cos} : \text{real} \Rightarrow \text{real}, \qquad \text{times} : \text{real} \Rightarrow \text{real} \Rightarrow \text{real}, \\ \text{deriv} : (\text{real} \Rightarrow \text{real}) \Rightarrow \text{real} \Rightarrow \text{real}\end{array}}$$

is safe, too [Suzuki, Kusakari, Blanqui, *IPSJ TOP'11*]

Bonus slide: Plain Function Passing, v2

PFP: Must the F really be direct arguments on left-hand side?

Example (deriv)

$$\frac{\text{deriv } (\lambda x.\text{sin } F[x])\ Y \to \text{times } (\text{deriv } (\lambda x.F[x])\ Y)\ (\text{cos } F[Y])}{\begin{array}{c}\text{sin}, \text{cos} : \text{real} \Rightarrow \text{real}, \qquad \text{times} : \text{real} \Rightarrow \text{real} \Rightarrow \text{real}, \\ \text{deriv} : (\text{real} \Rightarrow \text{real}) \Rightarrow \text{real} \Rightarrow \text{real}\end{array}}$$

is safe, too [Suzuki, Kusakari, Blanqui, *IPSJ TOP'11*]

But F of higher-order type in lower-order context on left-hand side can still be problematic:

Bonus slide: Plain Function Passing, v2

PFP: Must the F really be direct arguments on left-hand side?

Example (deriv)

$$\text{deriv}\ (\lambda x.\text{sin}\ F[x])\ Y \to \text{times}\ (\text{deriv}\ (\lambda x.F[x])\ Y)\ (\text{cos}\ F[Y])$$

$$\text{sin}, \text{cos} : \text{real} \Rightarrow \text{real}, \qquad \text{times} : \text{real} \Rightarrow \text{real} \Rightarrow \text{real},$$
$$\text{deriv} : (\text{real} \Rightarrow \text{real}) \Rightarrow \text{real} \Rightarrow \text{real}$$

is safe, too [Suzuki, Kusakari, Blanqui, *IPSJ TOP'11*]

But F of higher-order type in lower-order context on left-hand side can still be problematic:

$$\text{app}\ (\text{lam}\ F)\ x \to F\ x \quad \text{✗}$$