Higher-
Order
Termination

Cynthia Kop

June 20, 2012

Overview

1. What is Higher-Order Termination?
2. What is so difficult about this?
3. What do we do about that?
4. How do we do that?
   - Polynomial Interpretations
   - Path Orderings
   - Dependency Pairs
5. Where does that bring us?

Outline

1. What is Higher-Order Termination?
2. What is so difficult about this?
3. What do we do about that?
4. How do we do that?
   - Polynomial Interpretations
   - Path Orderings
   - Dependency Pairs
5. Where does that bring us?

$$\begin{array}{rcl} \mathtt{map}(F, \mathtt{nil}) & \Rightarrow & \mathtt{nil} \\ \mathtt{map}(F, \mathtt{cons}(h, t)) & \Rightarrow & \mathtt{cons}(F \cdot h, \mathtt{map}(F, t)) \end{array}$$

$$\begin{array}{rcl} \mathtt{rec}(0, y, F) & \Rightarrow & y \\ \mathtt{rec}(\mathtt{s}(x), y, F) & \Rightarrow & F \cdot x \cdot \mathtt{rec}(x, y, F) \end{array}$$

$$\begin{array}{rcl} \mathtt{and}(P, \mathtt{forall}(\lambda x.Q(x))) & \Rightarrow & \mathtt{forall}(\lambda x.\mathtt{and}(P, Q(x))) \\ \mathtt{or}(P, \mathtt{forall}(\lambda x.Q(x))) & \Rightarrow & \mathtt{forall}(\lambda x.\mathtt{or}(P, Q(x))) \\ \mathtt{not}(\mathtt{forall}(\lambda x.Q(x))) & \Rightarrow & \mathtt{exists}(\lambda x.\mathtt{not}(Q(x))) \end{array}$$

$$\begin{array}{rcl} \mathtt{map}(F, \mathtt{nil}) & \Rightarrow & \mathtt{nil} \\ \mathtt{map}(F, \mathtt{cons}(h, t)) & \Rightarrow & \mathtt{cons}(F \cdot h, \mathtt{map}(F, t)) \end{array}$$

$$\begin{array}{rcl} \mathtt{rec}(\mathtt{0}, y, F) & \Rightarrow & y \\ \mathtt{rec}(\mathtt{s}(x), y, F) & \Rightarrow & F \cdot x \cdot \mathtt{rec}(x, y, F) \end{array}$$

$$\begin{array}{rcl} \mathtt{and}(P, \mathtt{forall}(\lambda x.Q(x))) & \Rightarrow & \mathtt{forall}(\lambda x.\mathtt{and}(P, Q(x))) \\ \mathtt{or}(P, \mathtt{forall}(\lambda x.Q(x))) & \Rightarrow & \mathtt{forall}(\lambda x.\mathtt{or}(P, Q(x))) \\ \mathtt{not}(\mathtt{forall}(\lambda x.Q(x))) & \Rightarrow & \mathtt{exists}(\lambda x.\mathtt{not}(Q(x))) \end{array}$$

$$
\begin{array}{rcl}
\mathtt{map}(F, \mathtt{nil}) & \Rightarrow & \mathtt{nil} \\
\mathtt{map}(F, \mathtt{cons}(h, t)) & \Rightarrow & \mathtt{cons}(F \cdot h, \mathtt{map}(F, t))
\end{array}
$$

$$
\begin{array}{rcl}
\mathtt{rec}(\mathtt{0}, y, F) & \Rightarrow & y \\
\mathtt{rec}(\mathtt{s}(x), y, F) & \Rightarrow & F \cdot x \cdot \mathtt{rec}(x, y, F)
\end{array}
$$

$$
\begin{array}{rcl}
\mathtt{and}(P, \mathtt{forall}(\lambda x.Q(x))) & \Rightarrow & \mathtt{forall}(\lambda x.\mathtt{and}(P, Q(x))) \\
\mathtt{or}(P, \mathtt{forall}(\lambda x.Q(x))) & \Rightarrow & \mathtt{forall}(\lambda x.\mathtt{or}(P, Q(x))) \\
\mathtt{not}(\mathtt{forall}(\lambda x.Q(x))) & \Rightarrow & \mathtt{exists}(\lambda x.\mathtt{not}(Q(x)))
\end{array}
$$

What is a Higher-Order Term Rewriting System?

Not one clear definition!

In practice: first-order rewriting plus application and/or abstraction

- CS [Aczel, 1978]
- CRS [Klop, 1980]
- ERS [Khasidashvili, 1990]
- HRS [Nipkow, 1991]
- AFS [Jouannaud, Okada, 1991]
- HORS [van Oostrom, 1994]
- IDTS [Blanqui, 2000]
- STRS [Kusakari, 2001]
- STTRS [Yamada, 2001]
- applicative untyped TRSs

- Higher-Order Logic (Isabelle)
- Calculus Of Constructions (Coq)
- Combinatory Reduction Systems with eXtensions
- . . .

What is a Higher-Order Term Rewriting System?

Not one clear definition!

In practice: first-order rewriting plus application and/or abstraction

- CS [Aczel, 1978]
- CRS [Klop, 1980]
- ERS [Khasidashvili, 1990]
- HRS [Nipkow, 1991]
- AFS [Jouannaud, Okada, 1991]
- HORS [van Oostrom, 1994]
- IDTS [Blanqui, 2000]
- STRS [Kusakari, 2001]
- STTRS [Yamada, 2001]
- applicative untyped TRSs

- Higher-Order Logic (Isabelle)
- Calculus Of Constructions (Coq)
- Combinatory Reduction Systems with eXtensions
- . . .

What is a Higher-Order Term Rewriting System?

Not one clear definition!

In practice: first-order rewriting plus application and/or abstraction

- CS [Aczel, 1978]
- CRS [Klop, 1980]
- ERS [Khasidashvili, 1990]
- HRS [Nipkow, 1991]
- AFS [Jouannaud, Okada, 1991]
- HORS [van Oostrom, 1994]
- IDTS [Blanqui, 2000]
- STRS [Kusakari, 2001]
- STTRS [Yamada, 2001]
- applicative untyped TRSs

- Higher-Order Logic (Isabelle)
- Calculus Of Constructions (Coq)
- Combinatory Reduction Systems with eXtensions
- . . .

What is a Higher-Order Term Rewriting System?

Not one clear definition!

In practice: first-order rewriting plus application and/or abstraction

- CS [Aczel, 1978]
- CRS [Klop, 1980]
- ERS [Khasidashvili, 1990]
- HRS [Nipkow, 1991]
- AFS [Jouannaud, Okada, 1991]
- HORS [van Oostrom, 1994]
- IDTS [Blanqui, 2000]
- STRS [Kusakari, 2001]
- STTRS [Yamada, 2001]
- applicative untyped TRSs

- Higher-Order Logic (Isabelle)
- Calculus Of Constructions (Coq)
- Combinatory Reduction Systems with eXtensions
- ...

What systems do I consider?

Systems with λ-abstraction and β-reduction.
(or meta-variables, which can easily encode β-reduction).

Otherwise typically convertible to first-order systems!

Also needed: type system.
(or possibility to derive types)

Otherwise typically non-terminating!

Still many possibilities:

- function symbols with or without arity
- matching with variables or meta-variables
- application present or absent
- simple / polymorphic / dependent types
- terms modulo some equivalence relations (such as β)

What systems do I consider?

Systems with λ-abstraction and β-reduction.
(or meta-variables, which can easily encode β-reduction).

Otherwise typically convertible to first-order systems!

Also needed: type system.
(or possibility to derive types)

Otherwise typically non-terminating!

Still many possibilities:

- function symbols with or without arity
- matching with variables or meta-variables
- application present or absent
- simple / polymorphic / dependent types
- terms modulo some equivalence relations (such as β)

What systems do I consider?

Systems with λ-abstraction and β-reduction.
(or meta-variables, which can easily encode β-reduction).

Otherwise typically convertible to first-order systems!

Also needed: type system.
(or possibility to derive types)

Otherwise typically non-terminating!

Still many possibilities:

- function symbols with or without arity
- matching with variables or meta-variables
- application present or absent
- simple / polymorphic / dependent types
- terms modulo some equivalence relations (such as β)

What systems do I consider?

Systems with λ-abstraction and β-reduction.
(or meta-variables, which can easily encode β-reduction).

Otherwise typically convertible to first-order systems!

Also needed: type system.
(or possibility to derive types)

Otherwise typically non-terminating!

Still many possibilities:

- function symbols with or without arity
- matching with variables or meta-variables
- application present or absent
- simple / polymorphic / dependent types
- terms modulo some equivalence relations (such as β)

What systems do I consider?

Systems with λ-abstraction and β-reduction.
(or meta-variables, which can easily encode β-reduction).

Otherwise typically convertible to first-order systems!

Also needed: type system.
(or possibility to derive types)

Otherwise typically non-terminating!

Still many possibilities:

- function symbols with or without arity
- matching with variables or meta-variables
- application present or absent
- simple / polymorphic / dependent types
- terms modulo some equivalence relations (such as β)

What systems do I consider?

Systems with λ-abstraction and β-reduction.
(or meta-variables, which can easily encode β-reduction).

Otherwise typically convertible to first-order systems!

Also needed: type system.
(or possibility to derive types)

Otherwise typically non-terminating!

Still many possibilities:

- function symbols with or without arity
- matching with variables or meta-variables
- application present or absent
- simple / polymorphic / dependent types
- terms modulo some equivalence relations (such as β)

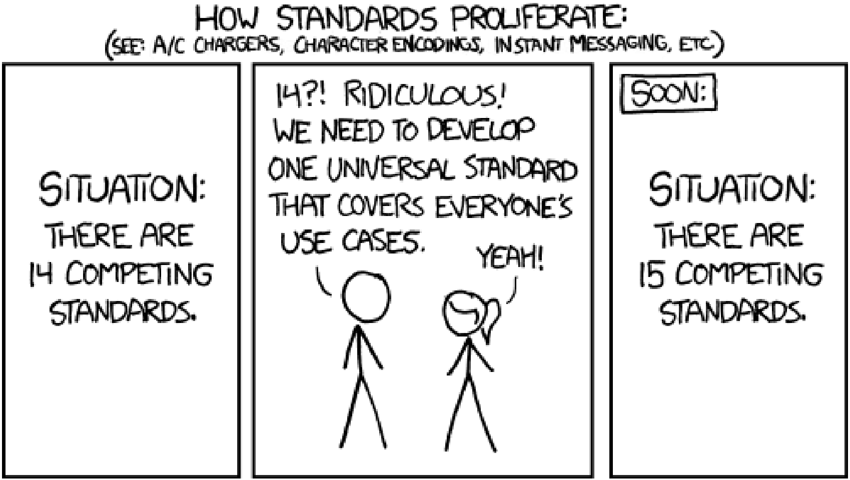
How to choose?

Make my own formalism!

How to choose?

Make my own formalism!

How to choose?



How to choose?

Make my own formalism!

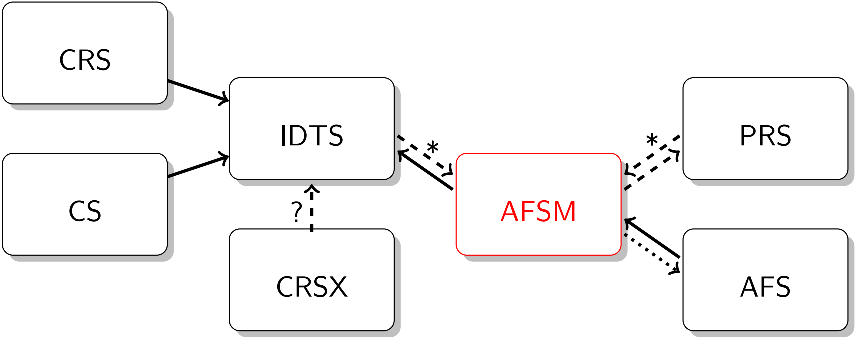
. . . and show how other formalisms relate to it

How to choose?

Make my own formalism!
. . . and show how other formalisms relate to it

How to choose?

Make my own formalism!
. . . and show how other formalisms relate to it



What is Termination?

Goal: study full termination

- no reduction strategy
- arbitrary start terms
  (in particular: start terms may contain λ-abstractions)

Outline

$$\mathtt{A}(\mathtt{L}(x)) \Rightarrow x$$

Non-terminating if $\mathtt{A} : [o] \to o \to o$ and $\mathtt{L} : [o \to o] \to o$

Define:

$$\omega := \mathtt{L}(\lambda x.(\mathtt{A}(x) \cdot x))$$

Then:

$$\begin{aligned} \mathtt{A}(\omega) \cdot \omega &= \mathtt{A}(\mathtt{L}(\lambda x.(\mathtt{A}(x) \cdot x))) \cdot \omega \\ &\Rightarrow (\lambda x.(\mathtt{A}(x) \cdot x)) \cdot \omega \\ &\Rightarrow_\beta \mathtt{A}(\omega) \cdot \omega \end{aligned}$$

Beta-reduction Causes Non-termination

$$\mathtt{A}(\mathtt{L}(x)) \Rightarrow x$$

Non-terminating if $\mathtt{A} : [\mathsf{o}] \to \mathsf{o} \to \mathsf{o}$ and $\mathtt{L} : [\mathsf{o} \to \mathsf{o}] \to \mathsf{o}$

Define:

$$\omega := \mathtt{L}(\lambda x.(\mathtt{A}(x) \cdot x))$$

Then:

$$\begin{aligned} \mathtt{A}(\omega) \cdot \omega &= \mathtt{A}(\mathtt{L}(\lambda x.(\mathtt{A}(x) \cdot x))) \cdot \omega \\ &\Rightarrow (\lambda x.(\mathtt{A}(x) \cdot x)) \cdot \omega \\ &\Rightarrow_\beta \mathtt{A}(\omega) \cdot \omega \end{aligned}$$

Beta-reduction Causes Non-termination

$$\mathtt{A}(\mathtt{L}(x)) \Rightarrow x$$

Non-terminating if $\mathtt{A} : [\mathsf{o}] \to \mathsf{o} \to \mathsf{o}$ and $\mathtt{L} : [\mathsf{o} \to \mathsf{o}] \to \mathsf{o}$

Define:

$$\omega := \mathtt{L}(\lambda x.(\mathtt{A}(x) \cdot x))$$

Then:

$$\begin{aligned} \mathtt{A}(\omega) \cdot \omega &= \mathtt{A}(\mathtt{L}(\lambda x.(\mathtt{A}(x) \cdot x))) \cdot \omega \\ &\Rightarrow (\lambda x.(\mathtt{A}(x) \cdot x)) \cdot \omega \\ &\Rightarrow_\beta \mathtt{A}(\omega) \cdot \omega \end{aligned}$$

$$\mathtt{A}(\mathtt{L}(x)) \Rightarrow x$$

Non-terminating if $\mathtt{A} : [\mathsf{o}] \to \mathsf{o} \to \mathsf{o}$ and $\mathtt{L} : [\mathsf{o} \to \mathsf{o}] \to \mathsf{o}$

Define:

$$\omega := \mathtt{L}(\lambda x.(\mathtt{A}(x) \cdot x))$$

Then:

$$\begin{aligned} \mathtt{A}(\omega) \cdot \omega &= \mathtt{A}(\mathtt{L}(\lambda x.(\mathtt{A}(x) \cdot x))) \cdot \omega \\ &\Rightarrow (\lambda x.(\mathtt{A}(x) \cdot x)) \cdot \omega \\ &\Rightarrow_\beta \mathtt{A}(\omega) \cdot \omega \end{aligned}$$

Beta-reduction Causes Non-termination

$$\mathtt{A}(\mathtt{L}(x)) \Rightarrow x$$

Non-terminating if $\mathtt{A} : [\mathsf{o}] \to \mathsf{o} \to \mathsf{o}$ and $\mathtt{L} : [\mathsf{o} \to \mathsf{o}] \to \mathsf{o}$

Define:

$$\omega := \mathtt{L}(\lambda x.(\mathtt{A}(x) \cdot x))$$

Then:

$$\begin{array}{rcl} \mathtt{A}(\omega) \cdot \omega & = & \mathtt{A}(\mathtt{L}(\lambda x.(\mathtt{A}(x) \cdot x))) \cdot \omega \\ & \Rightarrow & (\lambda x.(\mathtt{A}(x) \cdot x)) \cdot \omega \\ & \Rightarrow_\beta & \mathtt{A}(\omega) \cdot \omega \end{array}$$

Beta-reduction Causes Non-termination

$$\mathtt{A}(\mathtt{L}(x)) \Rightarrow x$$

Non-terminating if $\mathtt{A} : [\mathrm{o}] \to \mathrm{o} \to \mathrm{o}$ and $\mathtt{L} : [\mathrm{o} \to \mathrm{o}] \to \mathrm{o}$

Define:

$$\omega := \mathtt{L}(\lambda x.(\mathtt{A}(x) \cdot x))$$

Then:

$$\begin{array}{rcl} \mathtt{A}(\omega) \cdot \omega & = & \mathtt{A}(\mathtt{L}(\lambda x.(\mathtt{A}(x) \cdot x))) \cdot \omega \\ & \Rightarrow & (\lambda x.(\mathtt{A}(x) \cdot x)) \cdot \omega \\ & \Rightarrow_\beta & \mathtt{A}(\omega) \cdot \omega \end{array}$$

Beta-reduction Causes Non-termination

$$\mathtt{A}(\mathtt{L}(x)) \Rightarrow x$$

Non-terminating if $\mathtt{A} : [\mathrm{o}] \to \mathrm{o} \to \mathrm{o}$ and $\mathtt{L} : [\mathrm{o} \to \mathrm{o}] \to \mathrm{o}$

Define:

$$\omega := \mathtt{L}(\lambda x.(\mathtt{A}(x) \cdot x))$$

Then:

$$\begin{array}{rcl} \mathtt{A}(\omega) \cdot \omega & = & \mathtt{A}(\mathtt{L}(\lambda x.(\mathtt{A}(x) \cdot x))) \cdot \omega \\ & \Rightarrow & (\lambda x.(\mathtt{A}(x) \cdot x)) \cdot \omega \\ & \Rightarrow_\beta & \mathtt{A}(\omega) \cdot \omega \end{array}$$

Beta-reduction Causes Non-termination

$$\mathtt{A}(\mathtt{L}(x)) \Rightarrow x$$

Non-terminating if $\mathtt{A} : [o] \rightarrow o \rightarrow o$ and $\mathtt{L} : [o \rightarrow o] \rightarrow o$

Define:

$$\omega := \mathtt{L}(\lambda x.(\mathtt{A}(x) \cdot x))$$

Then:

$$\begin{array}{rcl} \mathtt{A}(\omega) \cdot \omega & = & \mathtt{A}(\mathtt{L}(\lambda x.(\mathtt{A}(x) \cdot x))) \cdot \omega \\ & \Rightarrow & (\lambda x.(\mathtt{A}(x) \cdot x)) \cdot \omega \\ & \Rightarrow_\beta & \mathtt{A}(\omega) \cdot \omega \end{array}$$

Outline

Solutions

- don't copy first-order methods, generalise the idea
- methods should be type-aware
- use techniques like computability

Solutions

- don't copy first-order methods, generalise the idea
- methods should be type-aware
- use techniques like computability

Solutions

- don't copy first-order methods, generalise the idea
- methods should be type-aware
- use techniques like computability

Outline

$$\begin{array}{rcl} \mathtt{minus}(x, \mathtt{0}) & \Rightarrow & x \\ \mathtt{minus}(\mathtt{0}, x) & \Rightarrow & \mathtt{0} \\ \mathtt{minus}(\mathtt{s}(x), \mathtt{s}(y)) & \Rightarrow & \mathtt{minus}(x, y) \end{array}$$

Use $[\![\_]\!]$ with:

- $[\![\mathtt{0}]\!] = 0$
- $[\![\mathtt{s}]\!] = \lambda n.n + 1$
- $[\![\mathtt{minus}]\!] = \lambda nm.n + m + 1$

$$\begin{array}{rcl} \mathtt{minus}(x, \mathtt{0}) & \Rightarrow & x \\ \mathtt{minus}(\mathtt{0}, x) & \Rightarrow & \mathtt{0} \\ \mathtt{minus}(\mathtt{s}(x), \mathtt{s}(y)) & \Rightarrow & \mathtt{minus}(x, y) \end{array}$$

Use $[\![\_]\!]$ with:

- $[\![\mathtt{0}]\!] = 0$
- $[\![\mathtt{s}]\!] = \boldsymbol{\lambda} n.n + 1$
- $[\![\mathtt{minus}]\!] = \boldsymbol{\lambda} nm.n + m + 1$

$$
\begin{array}{rclcl}
x+1 & = & [\![\texttt{minus}(x,\texttt{0})]\!] & > & [\![x]\!] & = & x \\
x+1 & = & [\![\texttt{minus}(\texttt{0},x)]\!] & > & [\![\texttt{0}]\!] & = & 0 \\
x+y+3 & = & [\![\texttt{minus}(\texttt{s}(x),\texttt{s}(y))]\!] & > & [\![\texttt{minus}(x,y)]\!] & = & x+y+1
\end{array}
$$

Use $[\![\_]\!]$ with:

- $[\![\texttt{0}]\!] = 0$
- $[\![\texttt{s}]\!] = \boldsymbol{\lambda} n.n + 1$
- $[\![\texttt{minus}]\!] = \boldsymbol{\lambda} nm.n + m + 1$

Required:

$$[\![(\lambda x.s) \cdot t]\!] \geq [\![s[x := t]]\!]$$

$$
\begin{array}{rcl}
\mathtt{map}(F, \mathtt{nil}) & \Rightarrow & \mathtt{nil} \\
\mathtt{map}(F, \mathtt{cons}(h, t)) & \Rightarrow & \mathtt{cons}(F \cdot h, \mathtt{map}(F, t))
\end{array}
$$

Use $[\![\_]\!]$ with:

- $[\![\mathtt{nil}]\!] = 1$
- $[\![\mathtt{cons}]\!] = \lambda nm.n + m + 1$
- $[\![\mathtt{map}]\!] = \lambda fn.f(0) + 2 \cdot n + n \cdot f(n)$
- $[\![\cdot^{\mathtt{nat} \to \mathtt{nat}}]\!] = \lambda fn.f(n) + n$

Constraints:

- $F(0) + 2 \cdot 1 + 1 \cdot F(1) > 1$
- $F(h+t+1) + h \cdot f(h+t+1) + t \cdot F(h+t+1) + 2 \cdot h + 1 > F(h) + t \cdot F(t) + 0$

Interpretations in Weakly Monotonic Functionals (Pol '96).

$$\begin{array}{rcl} \mathtt{map}(F, \mathtt{nil}) & \Rightarrow & \mathtt{nil} \\ \mathtt{map}(F, \mathtt{cons}(h, t)) & \Rightarrow & \mathtt{cons}(F \cdot h, \mathtt{map}(F, t)) \end{array}$$

Use $[\![\_]\!]$ with:

- $[\![\mathtt{nil}]\!] = 1$
- $[\![\mathtt{cons}]\!] = \boldsymbol{\lambda} nm.n + m + 1$
- $[\![\mathtt{map}]\!] = \boldsymbol{\lambda} fn.f(0) + 2 \cdot n + n \cdot f(n)$
- $[\![\cdot^{\mathtt{nat} \rightarrow \mathtt{nat}}]\!] = \boldsymbol{\lambda} fn.f(n) + n$

Constraints:

- $F(0) + 2 \cdot 1 + 1 \cdot F(1) > 1$
- $F(h+t+1) + h \cdot f(h+t+1) + t \cdot F(h+t+1) + 2 \cdot h + 1 > F(h) + t \cdot F(t) + 0$

Interpretations in Weakly Monotonic Functionals (Pol '96).

$$\begin{array}{rcl} \mathtt{map}(F,\mathtt{nil}) & \Rightarrow & \mathtt{nil} \\ \mathtt{map}(F,\mathtt{cons}(h,t)) & \Rightarrow & \mathtt{cons}(F \cdot h, \mathtt{map}(F,t)) \end{array}$$

Use $[\![\_]\!]$ with:

- $[\![\mathtt{nil}]\!] = 1$
- $[\![\mathtt{cons}]\!] = \boldsymbol{\lambda} nm.n + m + 1$
- $[\![\mathtt{map}]\!] = \boldsymbol{\lambda} fn.f(0) + 2 \cdot n + n \cdot f(n)$
- $[\![\cdot^{\mathtt{nat} \to \mathtt{nat}}]\!] = \boldsymbol{\lambda} fn.f(n) + n$

Constraints:

- $F(0) + 2 \cdot 1 + 1 \cdot F(1) > 1$
- $F(0) + 2 \cdot (h + t + 1) + (h + t + 1) \cdot F(h + t + 1) > F(h) + h + F(0) + 2 \cdot t + t \cdot F(t) + 1$

Interpretations in Weakly Monotonic Functionals (Pol '96).

$$\begin{array}{rcl} \mathtt{map}(F,\mathtt{nil}) & \Rightarrow & \mathtt{nil} \\ \mathtt{map}(F,\mathtt{cons}(h,t)) & \Rightarrow & \mathtt{cons}(F\cdot h,\mathtt{map}(F,t)) \end{array}$$

Use $[\![\_]\!]$ with:

- $[\![\mathtt{nil}]\!] = 1$
- $[\![\mathtt{cons}]\!] = \boldsymbol{\lambda} nm.n + m + 1$
- $[\![\mathtt{map}]\!] = \boldsymbol{\lambda} fn.f(0) + 2 \cdot n + n \cdot f(n)$
- $[\![\cdot^{\mathtt{nat}\to\mathtt{nat}}]\!] = \boldsymbol{\lambda} fn.f(n) + n$

Constraints:

- $F(0) + 2 \cdot 1 + 1 \cdot F(1) > 1$
- $F(h+t+1) + h \cdot f(h+t+1) + t \cdot F(h+t+1) + 2 \cdot h + 1 > F(h) + t \cdot F(t) + 0$

Interpretations in Weakly Monotonic Functionals (Pol '96).

$$\begin{array}{rcl} \mathtt{map}(F, \mathtt{nil}) & \Rightarrow & \mathtt{nil} \\ \mathtt{map}(F, \mathtt{cons}(h, t)) & \Rightarrow & \mathtt{cons}(F \cdot h, \mathtt{map}(F, t)) \end{array}$$

Use $[\![\_]\!]$ with:

- $[\![\mathtt{nil}]\!] = 1$
- $[\![\mathtt{cons}]\!] = \boldsymbol{\lambda} nm.n + m + 1$
- $[\![\mathtt{map}]\!] = \boldsymbol{\lambda} fn.f(0) + 2 \cdot n + n \cdot f(n)$
- $[\![\cdot^{\mathtt{nat} \to \mathtt{nat}}]\!] = \boldsymbol{\lambda} fn.f(n) + n$

Constraints:

- $F(0) + 2 \cdot 1 + 1 \cdot F(1) > 1$
- $F(h + t + 1) + h \cdot f(h + t + 1) + t \cdot F(h + t + 1) + 2 \cdot h + 1 > F(h) + t \cdot F(t) + 0$

Interpretations in Weakly Monotonic Functionals (Pol '96).

Higher-Order Polynomials

Definition

- variables are polynomials
- if p, q polynomials, then $p + q$ is a polynomial
- if p, q polynomials, then $p \cdot q$ is a polynomial
- if F is a variable of type $\tau_1 \rightarrow \ldots \rightarrow \tau_m \rightarrow \iota$ with ι a base type, and $p_1 \in Pol^{\tau_1}, \ldots, p_m \in Pol^{\tau_m}$, then $F(p_1, \ldots, p_m)$ is a higher-order polynomial
  - here, $Pol^{\sigma_1 \rightarrow \ldots \rightarrow \sigma_n}$ contains elements $\boldsymbol{\lambda} y_1 \in WM_{\sigma_1} \ldots y_n \in WM_{\sigma_n}.q$, with q a higher-order polynomial.

In particular, if $p_1, \ldots, p_n$ are higher-order polynomials, and $F : \iota_1 \rightarrow \ldots \rightarrow \iota_n \rightarrow \kappa$ a second-order variable, then $F(p_1, \ldots, p_n)$ is a higher-order polynomial.

Higher-Order Polynomials

Definition

- variables of base type are higher-order polynomials
- if p, q polynomials, then $p + q$ is a higher-order polynomial
- if p, q polynomials, then $p \cdot q$ is a higher-order polynomial
- if F is a variable of type $\tau_1 \to \ldots \to \tau_m \to \iota$ with ι a base type, and $p_1 \in Pol^{\tau_1}, \ldots, p_m \in Pol^{\tau_m}$, then $F(p_1, \ldots, p_m)$ is a higher-order polynomial
  - here, $Pol^{\sigma_1 \to \ldots \to \sigma_n}$ contains elements $\lambda y_1 \in WM_{\sigma_1} \ldots y_n \in WM_{\sigma_n}.q$, with q a higher-order polynomial.

In particular, if $p_1, \ldots, p_n$ are higher-order polynomials, and $F : \iota_1 \to \ldots \to \iota_n \to \kappa$ a second-order variable, then $F(p_1, \ldots, p_n)$ is a higher-order polynomial.

Higher-Order Polynomials

Definition

- variables of base type are higher-order polynomials
- if p, q polynomials, then $p + q$ is a higher-order polynomial
- if p, q polynomials, then $p \cdot q$ is a higher-order polynomial
- if F is a variable of type $\tau_1 \to \ldots \to \tau_m \to \iota$ with ι a base type, and $p_1 \in Pol^{\tau_1}, \ldots, p_m \in Pol^{\tau_m}$, then $F(p_1, \ldots, p_m)$ is a higher-order polynomial
  - here, $Pol^{\sigma_1 \to \ldots \to \sigma_n}$ contains elements $\boldsymbol{\lambda} y_1 \in WM_{\sigma_1} \ldots y_n \in WM_{\sigma_n}.q$, with q a higher-order polynomial.

In particular, if $p_1, \ldots, p_n$ are higher-order polynomials, and $F : \iota_1 \to \ldots \to \iota_n \to \kappa$ a second-order variable, then $F(p_1, \ldots, p_n)$ is a higher-order polynomial.

Higher-Order Polynomials

Definition

- variables of base type are higher-order polynomials
- if p, q polynomials, then $p + q$ is a higher-order polynomial
- if p, q polynomials, then $p \cdot q$ is a higher-order polynomial
- if F is a variable of type $\tau_1 \to \ldots \to \tau_m \to \iota$ with ι a base type, and $p_1 \in Pol^{\tau_1}, \ldots, p_m \in Pol^{\tau_m}$, then $F(p_1, \ldots, p_m)$ is a higher-order polynomial
  - here, $Pol^{\sigma_1 \to \ldots \to \sigma_n}$ contains elements $\boldsymbol{\lambda} y_1 \in WM_{\sigma_1} \ldots y_n \in WM_{\sigma_n}.q$, with q a higher-order polynomial.

In particular, if $p_1, \ldots, p_n$ are higher-order polynomials, and $F : \iota_1 \to \ldots \to \iota_n \to \kappa$ a second-order variable, then $F(p_1, \ldots, p_n)$ is a higher-order polynomial.

Ingredients:

- well-founded precedence $>$
- status function mapping each symbol f to *lex* or *mul*

Definition:

1. $f(s_1, \ldots, s_n) \succ t$ if $s_i \succeq t$ (some i);
2. $f(s_1, \ldots, s_n) \succ g(t_1, \ldots, t_m)$ if $f > g$, $\forall i. f(\vec{s}) \succ t_i$;
3. $f(s_1, \ldots, s_n) \succ f(t_1, \ldots, t_m)$ if $stat(f) = lex$, $[s_1, \ldots, s_n] \succ_{\mathrm{lex}} [t_1, \ldots, t_m]$, $\forall i. f(\vec{s}) \succ t_i$;
4. $f(s_1, \ldots, s_n) \succ f(t_1, \ldots, t_m)$ if $stat(f) = mul$, $\{\{s_1, \ldots, s_n\}\} \succ_{\mathrm{mul}} \{\{t_1, \ldots, t_m\}\}$;
5. $x \succeq x$;
6. $f(s_1, \ldots, s_n) \succeq f(t_1, \ldots, t_n)$ if $stat(f) = lex$ and each $s_i \succeq t_i$;
7. $f(s_1, \ldots, s_n) \succeq f(t_1, \ldots, t_n)$ if $stat(f) = mul$ and $\{\{s_1, \ldots, s_n\}\} \succeq_{\mathrm{mul}} \{\{t_1, \ldots, t_n\}\}$.
8. $f(s_1, \ldots, s_n) \succeq t$ if $f(s_1, \ldots, s_n) \succ t$.

Ingredients:

- well-founded precedence $>$
- status function mapping each symbol f to *lex* or *mul*

Definition:

1. $f(s_1, \ldots, s_n) \succ t$ if $s_i \succeq t$ (some i);
2. $f(s_1, \ldots, s_n) \succ g(t_1, \ldots, t_m)$ if $f > g$, $\forall i. f(\vec{s}) \succ t_i$;
3. $f(s_1, \ldots, s_n) \succ f(t_1, \ldots, t_m)$ if $stat(f) = lex$, $[s_1, \ldots, s_n] \succ_{\mathrm{lex}} [t_1, \ldots, t_m]$, $\forall i. f(\vec{s}) \succ t_i$;
4. $f(s_1, \ldots, s_n) \succ f(t_1, \ldots, t_m)$ if $stat(f) = mul$, $\{\{s_1, \ldots, s_n\}\} \succ_{\mathrm{mul}} \{\{t_1, \ldots, t_m\}\}$;
5. $x \succeq x$;
6. $f(s_1, \ldots, s_n) \succeq f(t_1, \ldots, t_n)$ if $stat(f) = lex$ and each $s_i \succeq t_i$;
7. $f(s_1, \ldots, s_n) \succeq f(t_1, \ldots, t_n)$ if $stat(f) = mul$ and $\{\{s_1, \ldots, s_n\}\} \succeq_{\mathrm{mul}} \{\{t_1, \ldots, t_n\}\}$.
8. $f(s_1, \ldots, s_n) \succeq t$ if $f(s_1, \ldots, s_n) \succ t$.

Ingredients:

- well-founded precedence $>$
- status function mapping each symbol f to *lex* or *mul*

Definition:

1. $f(s_1,\ldots,s_n) \succ t$ if $s_i \succeq t$ (some i);
2. $f(s_1,\ldots,s_n) \succ g(t_1,\ldots,t_m)$ if $f > g$, $\forall i. f(\vec{s}) \succ t_i$;
3. $f(s_1,\ldots,s_n) \succ f(t_1,\ldots,t_m)$ if $stat(f) = lex$, $[s_1,\ldots,s_n] \succ_{\text{lex}} [t_1,\ldots,t_m]$, $\forall i. f(\vec{s}) \succ t_i$;
4. $f(s_1,\ldots,s_n) \succ f(t_1,\ldots,t_m)$ if $stat(f) = mul$, $\{\{s_1,\ldots,s_n\}\} \succ_{\text{mul}} \{\{t_1,\ldots,t_m\}\}$;
5. $x \succeq x$;
6. $f(s_1,\ldots,s_n) \succeq f(t_1,\ldots,t_n)$ if $stat(f) = lex$ and each $s_i \succeq t_i$;
7. $f(s_1,\ldots,s_n) \succeq f(t_1,\ldots,t_n)$ if $stat(f) = mul$ and $\{\{s_1,\ldots,s_n\}\} \succeq_{\text{mul}} \{\{t_1,\ldots,t_n\}\}$.
8. $f(s_1,\ldots,s_n) \succeq t$ if $f(s_1,\ldots,s_n) \succ t$.

Ingredients:

- well-founded precedence $>$
- status function mapping each symbol f to *lex* or *mul*

Definition:

1. $f(s_1, \ldots, s_n) \succ t$ if $f^\star(s_1, \ldots, s_n) \succeq t$;
2. $f^\star(s_1, \ldots, s_n) \succeq t$ if $s_i \succeq t$ (some i);
3. $f^\star(s_1, \ldots, s_n) \succeq g(t_1, \ldots, t_m)$ if $f > g$, $\forall i. f^\star(\vec{s}) \succeq t_i$;
4. $f^\star(s_1, \ldots, s_n) \succeq f(t_1, \ldots, t_m)$ if $stat(f) = lex$, $[s_1, \ldots, s_n] \succ_{\mathrm{lex}} [t_1, \ldots, t_m]$, $\forall i. f^\star(\vec{s}) \succeq t_i$;
5. $f^\star(s_1, \ldots, s_n) \succeq f(t_1, \ldots, t_m)$ if $stat(f) = mul$, $\{\{s_1, \ldots, s_n\}\} \succ_{\mathrm{mul}} \{\{t_1, \ldots, t_m\}\}$;
6. $x \succeq x$;
7. $f(s_1, \ldots, s_n) \succeq f(t_1, \ldots, t_n)$ if $stat(f) = lex$ and each $s_i \succeq t_i$;
8. $f(s_1, \ldots, s_n) \succeq f(t_1, \ldots, t_n)$ if $stat(f) = mul$ and $\{\{s_1, \ldots, s_n\}\} \succeq_{\mathrm{mul}} \{\{t_1, \ldots, t_n\}\}$.
9. $f(s_1, \ldots, s_n) \succeq t$ if $f^\star(s_1, \ldots, s_n) \succeq t$.

Ingredients:

- application becomes a function symbol
- well-founded precedence $>$
- status function mapping each symbol f to *lex* or *mul*

So

$$\mathtt{map}(F, \mathtt{cons}(h, t)) \Rightarrow \mathtt{cons}(F \cdot h, \mathtt{map}(F, t))$$

becomes

$$\mathtt{map}(F, \mathtt{cons}(h, t)) \Rightarrow \mathtt{cons}(@^{\mathtt{nat} \to \mathtt{nat}}(F, h), \mathtt{map}(F, t))$$

Ingredients:

- application becomes a function symbol
- well-founded precedence $>$
- status function mapping each symbol f to *lex* or *mul*

So

$$\mathtt{map}(F, \mathtt{cons}(h, t)) \Rightarrow \mathtt{cons}(F \cdot h, \mathtt{map}(F, t))$$

becomes

$$\mathtt{map}(F, \mathtt{cons}(h, t)) \Rightarrow \mathtt{cons}(\mathtt{@}^{\mathtt{nat} \to \mathtt{nat}}(F, h), \mathtt{map}(F, t))$$

1. $\lambda\vec{x}.f(s_1,\ldots,s_n) \succ t$ if $\lambda\vec{x}.f^\star(s_1,\ldots,s_n) \succeq t$;
2. $f^\star(s_1,\ldots,s_n) \succeq t$ if $s_i\langle f^\star(\vec{s}),\ldots,f^\star(\vec{s})\rangle \succeq t$ (some i);
3. $f^\star(s_1,\ldots,s_n) \succeq g(t_1,\ldots,t_m)$ if $f > g$, $\forall i$: $f^\star_{\tau_i}(\vec{s}) \succeq t_1,\ldots,t_n$;
4. $f^\star(s_1,\ldots,s_n) \succeq f(t_1,\ldots,t_m)$ if $stat(f) = lex$, $[s_1,\ldots,s_n] \succ_{\mathrm{lex}} [t_1,\ldots,t_m]$, $\forall i.f^\star_{\tau_i}(\vec{s}) \succeq t_1,\ldots,t_m$;
5. $f^\star(s_1,\ldots,s_n) \succeq f(t_1,\ldots,t_m)$ if $stat(f) = mul$, $\{\{s_1,\ldots,s_n\}\} \succ_{\mathrm{mul}} \{\{t_1,\ldots,t_m\}\}$;
6. $f^\star(s_1,\ldots,s_n) \succeq \lambda x.t$ if $f^\star(s_1,\ldots,s_n,x) \succeq t$;
7. $x \succeq x$;
8. $\lambda x.s \succeq \lambda x.t$ if $s \succeq t$;
9. $F(s_1,\ldots,s_n) \succeq F(t_1,\ldots,t_n)$ if $\forall i.s_i \succeq t_i$;
10. $f(s_1,\ldots,s_n) \succeq f(t_1,\ldots,t_n)$ if $stat(f) = lex$ and each $s_i \succeq t_i$;
11. $f(s_1,\ldots,s_n) \succeq f(t_1,\ldots,t_n)$ if $stat(f) = mul$ and $\{\{s_1,\ldots,s_n\}\} \succeq_{\mathrm{mul}} \{\{t_1,\ldots,t_n\}\}$.
12. $f(s_1,\ldots,s_n) \succeq t$ if $f^\star(s_1,\ldots,s_n) \succeq t$.

$$\begin{array}{rcl} \texttt{nil} & : & \mathsf{o} \\ \texttt{cons} & : & [\mathsf{o} \times \mathsf{o}] \to \mathsf{o} \\ \texttt{map} & : & [(\mathsf{o} \to \mathsf{o}) \times \mathsf{o}] \to \mathsf{o} \\ @^{\mathsf{o}\to\mathsf{o}} & : & [(\mathsf{o} \to \mathsf{o}) \times \mathsf{o}] \to \mathsf{o} \end{array}$$

$$\begin{array}{rcl} \texttt{map}(F, \texttt{nil}) & \Rightarrow & \texttt{nil} \\ \texttt{map}(F, \texttt{cons}(h, t)) & \Rightarrow & \texttt{cons}(@^{\mathsf{o}\to\mathsf{o}}(F, h), \texttt{map}(F, t)) \end{array}$$

Choose $\texttt{map} > \texttt{cons}, @^{\mathsf{o}\to\mathsf{o}}$, all statuses *mul*.

$$\begin{array}{rcl} \mathtt{nil} & : & \mathsf{o} \\ \mathtt{cons} & : & [\mathsf{o} \times \mathsf{o}] \to \mathsf{o} \\ \mathtt{map} & : & [(\mathsf{o} \to \mathsf{o}) \times \mathsf{o}] \to \mathsf{o} \\ @^{\mathsf{o} \to \mathsf{o}} & : & [(\mathsf{o} \to \mathsf{o}) \times \mathsf{o}] \to \mathsf{o} \end{array}$$

$$\begin{array}{rcl} \mathtt{map}(F, \mathtt{nil}) & \Rightarrow & \mathtt{nil} \\ \mathtt{map}(F, \mathtt{cons}(h, t)) & \Rightarrow & \mathtt{cons}(@^{\mathsf{o} \to \mathsf{o}}(F, h), \mathtt{map}(F, t)) \end{array}$$

Choose $\mathtt{map} > \mathtt{cons}, @^{\mathsf{o} \to \mathsf{o}}$, all statuses *mul*.

$$\begin{array}{rcl} \mathtt{nil} & : & o \\ \mathtt{cons} & : & [o \times o] \rightarrow o \\ \mathtt{map} & : & [(o \rightarrow o) \times o] \rightarrow o \\ \mathtt{@}^{o \rightarrow o} & : & [(o \rightarrow o) \times o] \rightarrow o \end{array}$$

$$\begin{array}{rcl} \mathtt{map}(F, \mathtt{nil}) & \Rightarrow & \mathtt{nil} \\ \mathtt{map}(F, \mathtt{cons}(h, t)) & \Rightarrow & \mathtt{cons}(\mathtt{@}^{o \rightarrow o}(F, h), \mathtt{map}(F, t)) \end{array}$$

Choose $\mathtt{map} > \mathtt{cons}, \mathtt{@}^{o \rightarrow o}$, all statuses *mul*.

Goal: $\mathtt{map}(F, \mathtt{cons}(h, t)) \succ \mathtt{cons}(\mathtt{@}^{o \rightarrow o}(F, h), \mathtt{map}(F, t))$

$$\begin{array}{rcl} \texttt{nil} & : & \mathsf{o} \\ \texttt{cons} & : & [\mathsf{o} \times \mathsf{o}] \to \mathsf{o} \\ \texttt{map} & : & [(\mathsf{o} \to \mathsf{o}) \times \mathsf{o}] \to \mathsf{o} \\ \texttt{@}^{\mathsf{o}\to\mathsf{o}} & : & [(\mathsf{o} \to \mathsf{o}) \times \mathsf{o}] \to \mathsf{o} \end{array}$$

$$\begin{array}{rcl} \texttt{map}(F, \texttt{nil}) & \Rightarrow & \texttt{nil} \\ \texttt{map}(F, \texttt{cons}(h, t)) & \Rightarrow & \texttt{cons}(\texttt{@}^{\mathsf{o}\to\mathsf{o}}(F, h), \texttt{map}(F, t)) \end{array}$$

Choose $\texttt{map} > \texttt{cons}, \texttt{@}^{\mathsf{o}\to\mathsf{o}}$, all statuses *mul*.

Goal: $\texttt{map}(F, \texttt{cons}(h, t)) \succ \texttt{cons}(\texttt{@}^{\mathsf{o}\to\mathsf{o}}(F, h), \texttt{map}(F, t))$
Because (1)

- $\texttt{map}^{\star}(F, \texttt{cons}(h, t)) \succeq \texttt{cons}(\texttt{@}^{\mathsf{o}\to\mathsf{o}}(F, h), \texttt{map}(F, t))$

$$\begin{array}{rcl} \mathtt{nil} & : & o \\ \mathtt{cons} & : & [o \times o] \to o \\ \mathtt{map} & : & [(o \to o) \times o] \to o \\ \mathtt{@}^{o \to o} & : & [(o \to o) \times o] \to o \end{array}$$

$$\begin{array}{rcl} \mathtt{map}(F, \mathtt{nil}) & \Rightarrow & \mathtt{nil} \\ \mathtt{map}(F, \mathtt{cons}(h, t)) & \Rightarrow & \mathtt{cons}(\mathtt{@}^{o \to o}(F, h), \mathtt{map}(F, t)) \end{array}$$

Choose $\mathtt{map} > \mathtt{cons}, \mathtt{@}^{o \to o}$, all statuses *mul*.

Goal: $\mathtt{map}^{\star}(F, \mathtt{cons}(h, t)) \succeq \mathtt{cons}(\mathtt{@}^{o \to o}(F, h), \mathtt{map}(F, t))$

$$\begin{array}{rcl} \texttt{nil} & : & o \\ \texttt{cons} & : & [o \times o] \rightarrow o \\ \texttt{map} & : & [(o \rightarrow o) \times o] \rightarrow o \\ @^{o \rightarrow o} & : & [(o \rightarrow o) \times o] \rightarrow o \end{array}$$

$$\begin{array}{rcl} \texttt{map}(F, \texttt{nil}) & \Rightarrow & \texttt{nil} \\ \texttt{map}(F, \texttt{cons}(h, t)) & \Rightarrow & \texttt{cons}(@^{o \rightarrow o}(F, h), \texttt{map}(F, t)) \end{array}$$

Choose $\texttt{map} > \texttt{cons}, @^{o \rightarrow o}$, all statuses *mul*.

Goal: $\texttt{map}^{\star}(F, \texttt{cons}(h, t)) \succeq \texttt{cons}(@^{o \rightarrow o}(F, h), \texttt{map}(F, t))$
Because (3)

- $\texttt{map} > \texttt{cons}$
- $\texttt{map}^{\star}(F, \texttt{cons}(h, t)) \succeq @^{o \rightarrow o}(F, h)$
- $\texttt{map}^{\star}(F, \texttt{cons}(h, t)) \succeq \texttt{map}(F, t)$

$$\begin{array}{rcl} \mathtt{nil} & : & o \\ \mathtt{cons} & : & [o \times o] \to o \\ \mathtt{map} & : & [(o \to o) \times o] \to o \\ @^{o\to o} & : & [(o \to o) \times o] \to o \end{array}$$

$$\begin{array}{rcl} \mathtt{map}(F, \mathtt{nil}) & \Rightarrow & \mathtt{nil} \\ \mathtt{map}(F, \mathtt{cons}(h, t)) & \Rightarrow & \mathtt{cons}(@^{o\to o}(F, h), \mathtt{map}(F, t)) \end{array}$$

Choose $\mathtt{map} > \mathtt{cons}, @^{o\to o}$, all statuses *mul*.

Goal 1: $\mathtt{map}^\star(F, \mathtt{cons}(h, t)) \succeq @^{o\to o}(F, h)$
Goal 2: $\mathtt{map}^\star(F, \mathtt{cons}(h, t)) \succeq \mathtt{map}(F, t)$

$$\begin{array}{rcl} \mathtt{nil} & : & o \\ \mathtt{cons} & : & [o \times o] \to o \\ \mathtt{map} & : & [(o \to o) \times o] \to o \\ @^{o \to o} & : & [(o \to o) \times o] \to o \end{array}$$

$$\begin{array}{rcl} \mathtt{map}(F, \mathtt{nil}) & \Rightarrow & \mathtt{nil} \\ \mathtt{map}(F, \mathtt{cons}(h, t)) & \Rightarrow & \mathtt{cons}(@^{o \to o}(F, h), \mathtt{map}(F, t)) \end{array}$$

Choose $\mathtt{map} > \mathtt{cons}, @^{o \to o}$, all statuses *mul*.

Goal 1: $\mathtt{map}^\star(F, \mathtt{cons}(h, t)) \succeq @^{o \to o}(F, h)$
Goal 2: $\mathtt{map}^\star(F, \mathtt{cons}(h, t)) \succeq \mathtt{map}(F, t)$
Because (5)

- $F \succeq F$ (by 9)
- $\mathtt{cons}(h, t) \succ t$

$$\begin{array}{rcl} \texttt{nil} & : & o \\ \texttt{cons} & : & [o \times o] \to o \\ \texttt{map} & : & [(o \to o) \times o] \to o \\ @^{o \to o} & : & [(o \to o) \times o] \to o \end{array}$$

$$\begin{array}{rcl} \texttt{map}(F, \texttt{nil}) & \Rightarrow & \texttt{nil} \\ \texttt{map}(F, \texttt{cons}(h, t)) & \Rightarrow & \texttt{cons}(@^{o \to o}(F, h), \texttt{map}(F, t)) \end{array}$$

Choose $\texttt{map} > \texttt{cons}, @^{o \to o}$, all statuses *mul*.

Goal 1: $\texttt{map}^{\star}(F, \texttt{cons}(h, t)) \succeq @^{o \to o}(F, h)$
Goal 2: $\texttt{cons}(h, t) \succ t$

$$\begin{array}{rcl} \texttt{nil} & : & o \\ \texttt{cons} & : & [o \times o] \to o \\ \texttt{map} & : & [(o \to o) \times o] \to o \\ @^{o \to o} & : & [(o \to o) \times o] \to o \end{array}$$

$$\begin{array}{rcl} \texttt{map}(F, \texttt{nil}) & \Rightarrow & \texttt{nil} \\ \texttt{map}(F, \texttt{cons}(h, t)) & \Rightarrow & \texttt{cons}(@^{o \to o}(F, h), \texttt{map}(F, t)) \end{array}$$

Choose $\texttt{map} > \texttt{cons}, @^{o \to o}$, all statuses *mul*.

Goal 1: $\texttt{map}^{\star}(F, \texttt{cons}(h, t)) \succeq @^{o \to o}(F, h)$
Goal 2: $\texttt{cons}(h, t) \succ t$
Because (1)

- $\texttt{cons}^{\star}(h, t) \succeq t$

$$\begin{array}{rcl} \mathtt{nil} & : & \mathsf{o} \\ \mathtt{cons} & : & [\mathsf{o} \times \mathsf{o}] \to \mathsf{o} \\ \mathtt{map} & : & [(\mathsf{o} \to \mathsf{o}) \times \mathsf{o}] \to \mathsf{o} \\ @^{\mathsf{o} \to \mathsf{o}} & : & [(\mathsf{o} \to \mathsf{o}) \times \mathsf{o}] \to \mathsf{o} \end{array}$$

$$\begin{array}{rcl} \mathtt{map}(F, \mathtt{nil}) & \Rightarrow & \mathtt{nil} \\ \mathtt{map}(F, \mathtt{cons}(h, t)) & \Rightarrow & \mathtt{cons}(@^{\mathsf{o} \to \mathsf{o}}(F, h), \mathtt{map}(F, t)) \end{array}$$

Choose $\mathtt{map} > \mathtt{cons}, @^{\mathsf{o} \to \mathsf{o}}$, all statuses *mul*.

Goal 1: $\mathtt{map}^{\star}(F, \mathtt{cons}(h, t)) \succeq @^{\mathsf{o} \to \mathsf{o}}(F, h)$
Goal 2: $\mathtt{cons}^{\star}(h, t) \succeq t$

$$\begin{array}{rcl} \mathtt{nil} & : & o \\ \mathtt{cons} & : & [o \times o] \to o \\ \mathtt{map} & : & [(o \to o) \times o] \to o \\ @^{o \to o} & : & [(o \to o) \times o] \to o \end{array}$$

$$\begin{array}{rcl} \mathtt{map}(F, \mathtt{nil}) & \Rightarrow & \mathtt{nil} \\ \mathtt{map}(F, \mathtt{cons}(h, t)) & \Rightarrow & \mathtt{cons}(@^{o \to o}(F, h), \mathtt{map}(F, t)) \end{array}$$

Choose $\mathtt{map} > \mathtt{cons}, @^{o \to o}$, all statuses *mul*.

Goal 1: $\mathtt{map}^{\star}(F, \mathtt{cons}(h, t)) \succeq @^{o \to o}(F, h)$
Goal 2: $\mathtt{cons}^{\star}(h, t) \succeq t$
Because (2)

- $t \succeq t$ (by 9)

$$\begin{array}{rcl} \texttt{nil} & : & o \\ \texttt{cons} & : & [o \times o] \to o \\ \texttt{map} & : & [(o \to o) \times o] \to o \\ \texttt{@}^{o \to o} & : & [(o \to o) \times o] \to o \end{array}$$

$$\begin{array}{rcl} \texttt{map}(F, \texttt{nil}) & \Rightarrow & \texttt{nil} \\ \texttt{map}(F, \texttt{cons}(h, t)) & \Rightarrow & \texttt{cons}(\texttt{@}^{o \to o}(F, h), \texttt{map}(F, t)) \end{array}$$

Choose $\texttt{map} > \texttt{cons}, \texttt{@}^{o \to o}$, all statuses *mul*.

Goal 1: $\texttt{map}^{\star}(F, \texttt{cons}(h, t)) \succeq \texttt{@}^{o \to o}(F, h)$

$$\begin{array}{rcl} \mathtt{nil} & : & o \\ \mathtt{cons} & : & [o \times o] \rightarrow o \\ \mathtt{map} & : & [(o \rightarrow o) \times o] \rightarrow o \\ @^{o \rightarrow o} & : & [(o \rightarrow o) \times o] \rightarrow o \end{array}$$

$$\begin{array}{rcl} \mathtt{map}(F, \mathtt{nil}) & \Rightarrow & \mathtt{nil} \\ \mathtt{map}(F, \mathtt{cons}(h, t)) & \Rightarrow & \mathtt{cons}(@^{o \rightarrow o}(F, h), \mathtt{map}(F, t)) \end{array}$$

Choose $\mathtt{map} > \mathtt{cons}, @^{o \rightarrow o}$, all statuses *mul*.

Goal 1: $\mathtt{map}^{\star}(F, \mathtt{cons}(h, t)) \succeq @^{o \rightarrow o}(F, h)$
Because (3)

- $\mathtt{map} > @^{o \rightarrow o}$
- $\mathtt{map}^{\star}_{o \rightarrow o}(F, \mathtt{cons}(h, t)) \succeq F$
- $\mathtt{map}^{\star}(F, \mathtt{cons}(h, t)) \succeq t$

$$\begin{array}{rcl} \mathtt{nil} & : & o \\ \mathtt{cons} & : & [o \times o] \to o \\ \mathtt{map} & : & [(o \to o) \times o] \to o \\ @^{o \to o} & : & [(o \to o) \times o] \to o \end{array}$$

$$\begin{array}{rcl} \mathtt{map}(F, \mathtt{nil}) & \Rightarrow & \mathtt{nil} \\ \mathtt{map}(F, \mathtt{cons}(h, t)) & \Rightarrow & \mathtt{cons}(@^{o \to o}(F, h), \mathtt{map}(F, t)) \end{array}$$

Choose $\mathtt{map} > \mathtt{cons}, @^{o \to o}$, all statuses *mul*.

Goal 1: $\mathtt{map}^{\star}_{o \to o}(F, \mathtt{cons}(h, t)) \succeq F$
Goal 2: $\mathtt{map}^{\star}(F, \mathtt{cons}(h, t)) \succeq h$

$$\begin{array}{rcl} \texttt{nil} & : & \mathsf{o} \\ \texttt{cons} & : & [\mathsf{o} \times \mathsf{o}] \to \mathsf{o} \\ \texttt{map} & : & [(\mathsf{o} \to \mathsf{o}) \times \mathsf{o}] \to \mathsf{o} \\ @^{\mathsf{o} \to \mathsf{o}} & : & [(\mathsf{o} \to \mathsf{o}) \times \mathsf{o}] \to \mathsf{o} \end{array}$$

$$\begin{array}{rcl} \texttt{map}(F, \texttt{nil}) & \Rightarrow & \texttt{nil} \\ \texttt{map}(F, \texttt{cons}(h, t)) & \Rightarrow & \texttt{cons}(@^{\mathsf{o} \to \mathsf{o}}(F, h), \texttt{map}(F, t)) \end{array}$$

Choose $\texttt{map} > \texttt{cons}, @^{\mathsf{o} \to \mathsf{o}}$, all statuses *mul*.

Goal 1: $\texttt{map}^{\star}_{\mathsf{o} \to \mathsf{o}}(F, \texttt{cons}(h, t)) \succeq F$
Goal 2: $\texttt{map}^{\star}(F, \texttt{cons}(h, t)) \succeq h$
Because (2)

- $\texttt{cons}(h, t) \succeq h$

$$\begin{array}{rcl} \texttt{nil} & : & \mathsf{o} \\ \texttt{cons} & : & [\mathsf{o} \times \mathsf{o}] \to \mathsf{o} \\ \texttt{map} & : & [(\mathsf{o} \to \mathsf{o}) \times \mathsf{o}] \to \mathsf{o} \\ \texttt{@}^{\mathsf{o} \to \mathsf{o}} & : & [(\mathsf{o} \to \mathsf{o}) \times \mathsf{o}] \to \mathsf{o} \end{array}$$

$$\begin{array}{rcl} \texttt{map}(F, \texttt{nil}) & \Rightarrow & \texttt{nil} \\ \texttt{map}(F, \texttt{cons}(h, t)) & \Rightarrow & \texttt{cons}(\texttt{@}^{\mathsf{o} \to \mathsf{o}}(F, h), \texttt{map}(F, t)) \end{array}$$

Choose $\texttt{map} > \texttt{cons}, \texttt{@}^{\mathsf{o} \to \mathsf{o}}$, all statuses *mul*.

Goal 1: $\texttt{map}^{\star}_{\mathsf{o} \to \mathsf{o}}(F, \texttt{cons}(h, t)) \succeq F$
Goal 2: $\texttt{cons}(h, t) \succeq h$

$$\begin{array}{rcl} \texttt{nil} & : & \mathsf{o} \\ \texttt{cons} & : & [\mathsf{o} \times \mathsf{o}] \to \mathsf{o} \\ \texttt{map} & : & [(\mathsf{o} \to \mathsf{o}) \times \mathsf{o}] \to \mathsf{o} \\ @^{\mathsf{o} \to \mathsf{o}} & : & [(\mathsf{o} \to \mathsf{o}) \times \mathsf{o}] \to \mathsf{o} \end{array}$$

$$\begin{array}{rcl} \texttt{map}(F, \texttt{nil}) & \Rightarrow & \texttt{nil} \\ \texttt{map}(F, \texttt{cons}(h, t)) & \Rightarrow & \texttt{cons}(@^{\mathsf{o} \to \mathsf{o}}(F, h), \texttt{map}(F, t)) \end{array}$$

Choose $\texttt{map} > \texttt{cons}, @^{\mathsf{o} \to \mathsf{o}}$, all statuses *mul*.

Goal 1: $\texttt{map}^{\star}_{\mathsf{o} \to \mathsf{o}}(F, \texttt{cons}(h, t)) \succeq F$
Goal 2: $\texttt{cons}(h, t) \succeq h$
Because (12)

- $\texttt{cons}^{\star}(h, t) \succeq h$

$$\begin{array}{rcl} \mathtt{nil} & : & \mathsf{o} \\ \mathtt{cons} & : & [\mathsf{o} \times \mathsf{o}] \to \mathsf{o} \\ \mathtt{map} & : & [(\mathsf{o} \to \mathsf{o}) \times \mathsf{o}] \to \mathsf{o} \\ @^{\mathsf{o} \to \mathsf{o}} & : & [(\mathsf{o} \to \mathsf{o}) \times \mathsf{o}] \to \mathsf{o} \end{array}$$

$$\begin{array}{rcl} \mathtt{map}(F, \mathtt{nil}) & \Rightarrow & \mathtt{nil} \\ \mathtt{map}(F, \mathtt{cons}(h, t)) & \Rightarrow & \mathtt{cons}(@^{\mathsf{o} \to \mathsf{o}}(F, h), \mathtt{map}(F, t)) \end{array}$$

Choose $\mathtt{map} > \mathtt{cons}, @^{\mathsf{o} \to \mathsf{o}}$, all statuses *mul*.

Goal 1: $\mathtt{map}^{\star}_{\mathsf{o} \to \mathsf{o}}(F, \mathtt{cons}(h, t)) \succeq F$
Goal 2: $\mathtt{cons}^{\star}(h, t) \succeq h$

$$
\begin{array}{rcl}
\mathtt{nil} & : & \mathsf{o} \\
\mathtt{cons} & : & [\mathsf{o} \times \mathsf{o}] \to \mathsf{o} \\
\mathtt{map} & : & [(\mathsf{o} \to \mathsf{o}) \times \mathsf{o}] \to \mathsf{o} \\
@^{\mathsf{o} \to \mathsf{o}} & : & [(\mathsf{o} \to \mathsf{o}) \times \mathsf{o}] \to \mathsf{o}
\end{array}
$$

$$
\begin{array}{rcl}
\mathtt{map}(F, \mathtt{nil}) & \Rightarrow & \mathtt{nil} \\
\mathtt{map}(F, \mathtt{cons}(h, t)) & \Rightarrow & \mathtt{cons}(@^{\mathsf{o} \to \mathsf{o}}(F, h), \mathtt{map}(F, t))
\end{array}
$$

Choose $\mathtt{map} > \mathtt{cons}, @^{\mathsf{o} \to \mathsf{o}}$, all statuses *mul*.

Goal 1: $\mathtt{map}^{\star}_{\mathsf{o} \to \mathsf{o}}(F, \mathtt{cons}(h, t)) \succeq F$
Goal 2: $\mathtt{cons}^{\star}(h, t) \succeq h$
Because (2)

- $h \succeq h$ (by 9)

$$\begin{array}{rcl} \mathtt{nil} & : & o \\ \mathtt{cons} & : & [o \times o] \to o \\ \mathtt{map} & : & [(o \to o) \times o] \to o \\ @^{o\to o} & : & [(o \to o) \times o] \to o \end{array}$$

$$\begin{array}{rcl} \mathtt{map}(F, \mathtt{nil}) & \Rightarrow & \mathtt{nil} \\ \mathtt{map}(F, \mathtt{cons}(h, t)) & \Rightarrow & \mathtt{cons}(@^{o\to o}(F, h), \mathtt{map}(F, t)) \end{array}$$

Choose $\mathtt{map} > \mathtt{cons}, @^{o\to o}$, all statuses *mul*.

Goal 1: $\mathtt{map}^{\star}_{o\to o}(F, \mathtt{cons}(h, t)) \succeq F$

$$\begin{array}{rcl} \texttt{nil} & : & \mathsf{o} \\ \texttt{cons} & : & [\mathsf{o} \times \mathsf{o}] \to \mathsf{o} \\ \texttt{map} & : & [(\mathsf{o} \to \mathsf{o}) \times \mathsf{o}] \to \mathsf{o} \\ @^{\mathsf{o}\to\mathsf{o}} & : & [(\mathsf{o} \to \mathsf{o}) \times \mathsf{o}] \to \mathsf{o} \end{array}$$

$$\begin{array}{rcl} \texttt{map}(F, \texttt{nil}) & \Rightarrow & \texttt{nil} \\ \texttt{map}(F, \texttt{cons}(h, t)) & \Rightarrow & \texttt{cons}(@^{\mathsf{o}\to\mathsf{o}}(F, h), \texttt{map}(F, t)) \end{array}$$

Choose $\texttt{map} > \texttt{cons}, @^{\mathsf{o}\to\mathsf{o}}$, all statuses *mul*.

Goal 1: $\texttt{map}^{\star}_{\mathsf{o}\to\mathsf{o}}(F, \texttt{cons}(h, t)) \succeq F$
Because (2)

- $F \succeq F$ (by 9).

$$\begin{array}{rcl} \texttt{nil} & : & \mathsf{o} \\ \texttt{cons} & : & [\mathsf{o} \times \mathsf{o}] \to \mathsf{o} \\ \texttt{map} & : & [(\mathsf{o} \to \mathsf{o}) \times \mathsf{o}] \to \mathsf{o} \\ @^{\mathsf{o}\to\mathsf{o}} & : & [(\mathsf{o} \to \mathsf{o}) \times \mathsf{o}] \to \mathsf{o} \end{array}$$

$$\begin{array}{rcl} \texttt{map}(F, \texttt{nil}) & \Rightarrow & \texttt{nil} \\ \texttt{map}(F, \texttt{cons}(h, t)) & \Rightarrow & \texttt{cons}(@^{\mathsf{o}\to\mathsf{o}}(F, h), \texttt{map}(F, t)) \end{array}$$

Choose $\texttt{map} > \texttt{cons}, @^{\mathsf{o}\to\mathsf{o}}$, all statuses *mul*.

Termination proved!

$$\begin{array}{lcl} \mathtt{A}^{\sigma,\tau} & : & [(\sigma \to \tau) \times \sigma] \to \tau \\ \mathtt{L}^{\sigma,\tau} & : & [\sigma \to \tau] \to \sigma \to \tau \end{array}$$

$$\mathtt{A}^{\sigma,\tau}(\mathtt{L}^{\sigma,\tau}(\lambda x.F(x)), Y) \Rightarrow F(Y)$$

$$\begin{array}{lcl} \mathtt{A}^{\sigma,\tau} & : & [(\sigma \rightarrow \tau) \times \sigma] \rightarrow \tau \\ \mathtt{L}^{\sigma,\tau} & : & [\sigma \rightarrow \tau] \rightarrow \sigma \rightarrow \tau \end{array}$$

$$\mathtt{A}^{\sigma,\tau}(\mathtt{L}^{\sigma,\tau}(\lambda x.F(x)), Y) \Rightarrow F(Y)$$

Goal: $\mathtt{A}(\mathtt{L}(\lambda x.F(x)), Y) \succ F(Y)$

$$\begin{array}{lcl} \mathtt{A}^{\sigma,\tau} & : & [(\sigma \to \tau) \times \sigma] \to \tau \\ \mathtt{L}^{\sigma,\tau} & : & [\sigma \to \tau] \to \sigma \to \tau \end{array}$$

$$\mathtt{A}^{\sigma,\tau}(\mathtt{L}^{\sigma,\tau}(\lambda x.F(x)), Y) \Rightarrow F(Y)$$

Goal: $\mathtt{A}(\mathtt{L}(\lambda x.F(x)), Y) \succ F(Y)$
Because: (1)

- $\mathtt{A}^{\star}(\mathtt{L}(\lambda x.F(x)), Y) \succeq F(Y)$

$$\begin{array}{lcl} \mathtt{A}^{\sigma,\tau} & : & [(\sigma \rightarrow \tau) \times \sigma] \rightarrow \tau \\ \mathtt{L}^{\sigma,\tau} & : & [\sigma \rightarrow \tau] \rightarrow \sigma \rightarrow \tau \end{array}$$

$$\mathtt{A}^{\sigma,\tau}(\mathtt{L}^{\sigma,\tau}(\lambda x.F(x)), Y) \Rightarrow F(Y)$$

Goal: $\mathtt{A}^{\star}(\mathtt{L}(\lambda x.F(x)), Y) \succeq F(Y)$

$$\begin{array}{lcl} \mathtt{A}^{\sigma,\tau} & : & [(\sigma \to \tau) \times \sigma] \to \tau \\ \mathtt{L}^{\sigma,\tau} & : & [\sigma \to \tau] \to \sigma \to \tau \end{array}$$

$$\mathtt{A}^{\sigma,\tau}(\mathtt{L}^{\sigma,\tau}(\lambda x.F(x)), Y) \Rightarrow F(Y)$$

Goal: $\mathtt{A}^{\star}(\mathtt{L}(\lambda x.F(x)), Y) \succeq F(Y)$
Because: (2)

- $\mathtt{L}(\lambda x.F(x)) : \sigma \to \tau$ and $F(Y) : \tau$
- $\mathtt{L}^{\star}(\lambda x.F(x), \mathtt{A}^{\star}_{\sigma}(\mathtt{L}(\lambda x.F(x)), Y)) \succeq F(Y)$

$$\begin{array}{lcl} \mathtt{A}^{\sigma,\tau} & : & [(\sigma \to \tau) \times \sigma] \to \tau \\ \mathtt{L}^{\sigma,\tau} & : & [\sigma \to \tau] \to \sigma \to \tau \end{array}$$

$$\mathtt{A}^{\sigma,\tau}(\mathtt{L}^{\sigma,\tau}(\lambda x.F(x)), Y) \Rightarrow F(Y)$$

Goal: $\mathtt{L}^{\star}(\lambda x.F(x), \mathtt{A}^{\star}_{\sigma}(\mathtt{L}(\lambda x.F(x)), Y)) \succeq F(Y)$

$$\begin{array}{lcl} \mathtt{A}^{\sigma,\tau} & : & [(\sigma \to \tau) \times \sigma] \to \tau \\ \mathtt{L}^{\sigma,\tau} & : & [\sigma \to \tau] \to \sigma \to \tau \end{array}$$

$$\mathtt{A}^{\sigma,\tau}(\mathtt{L}^{\sigma,\tau}(\lambda x.F(x)), Y) \Rightarrow F(Y)$$

Goal: $\mathtt{L}^{\star}(\lambda x.F(x), \mathtt{A}^{\star}_{\sigma}(\mathtt{L}(\lambda x.F(x)), Y)) \succeq F(Y)$
Because: (2)

- $\lambda x.F(x) : \sigma \to \tau$ and $F(Y) : \tau$
- $F(\mathtt{L}^{\star}_{\sigma}(\lambda x.F(x), \mathtt{A}^{\star}_{\sigma}(\mathtt{L}(\lambda x.F(x)), Y))) \succeq F(Y)$

$$\begin{array}{lcl} \mathtt{A}^{\sigma,\tau} & : & [(\sigma \to \tau) \times \sigma] \to \tau \\ \mathtt{L}^{\sigma,\tau} & : & [\sigma \to \tau] \to \sigma \to \tau \end{array}$$

$$\mathtt{A}^{\sigma,\tau}(\mathtt{L}^{\sigma,\tau}(\lambda x.F(x)), Y) \Rightarrow F(Y)$$

Goal: $F(\mathtt{L}^{\star}_{\sigma}(\lambda x.F(x), \mathtt{A}^{\star}_{\sigma}(\mathtt{L}(\lambda x.F(x)), Y))) \succeq F(Y)$

$$\begin{array}{lcl} \mathtt{A}^{\sigma,\tau} & : & [(\sigma \rightarrow \tau) \times \sigma] \rightarrow \tau \\ \mathtt{L}^{\sigma,\tau} & : & [\sigma \rightarrow \tau] \rightarrow \sigma \rightarrow \tau \end{array}$$

$$\mathtt{A}^{\sigma,\tau}(\mathtt{L}^{\sigma,\tau}(\lambda x.F(x)), Y) \Rightarrow F(Y)$$

Goal: $F(\mathtt{L}^{\star}_{\sigma}(\lambda x.F(x), \mathtt{A}^{\star}_{\sigma}(\mathtt{L}(\lambda x.F(x)), Y))) \succeq F(Y)$
Because: (9)

- $\mathtt{L}^{\star}_{\sigma}(\lambda x.F(x), \mathtt{A}^{\star}_{\sigma}(\mathtt{L}(\lambda x.F(x)), Y)) \succeq Y$

$$\mathtt{A}^{\sigma,\tau} \quad : \quad [(\sigma \to \tau) \times \sigma] \to \tau$$
$$\mathtt{L}^{\sigma,\tau} \quad : \quad [\sigma \to \tau] \to \sigma \to \tau$$

$$\mathtt{A}^{\sigma,\tau}(\mathtt{L}^{\sigma,\tau}(\lambda x.F(x)), Y) \Rightarrow F(Y)$$

Goal: $\mathtt{L}^{\star}_{\sigma}(\lambda x.F(x), \mathtt{A}^{\star}_{\sigma}(\mathtt{L}(\lambda x.F(x)), Y)) \succeq Y$

$$\begin{array}{lcl} \mathtt{A}^{\sigma,\tau} & : & [(\sigma \to \tau) \times \sigma] \to \tau \\ \mathtt{L}^{\sigma,\tau} & : & [\sigma \to \tau] \to \sigma \to \tau \end{array}$$

$$\mathtt{A}^{\sigma,\tau}(\mathtt{L}^{\sigma,\tau}(\lambda x.F(x)), Y) \Rightarrow F(Y)$$

Goal: $\mathtt{L}^{\star}_{\sigma}(\lambda x.F(x), \mathtt{A}^{\star}_{\sigma}(\mathtt{L}(\lambda x.F(x)), Y)) \succeq Y$
Because: (2)

- $\mathtt{A}^{\star}_{\sigma}(\mathtt{L}(\lambda x.F(x)), Y) \succeq Y$

$$\begin{array}{lcl} \mathtt{A}^{\sigma,\tau} & : & [(\sigma \rightarrow \tau) \times \sigma] \rightarrow \tau \\ \mathtt{L}^{\sigma,\tau} & : & [\sigma \rightarrow \tau] \rightarrow \sigma \rightarrow \tau \end{array}$$

$$\mathtt{A}^{\sigma,\tau}(\mathtt{L}^{\sigma,\tau}(\lambda x.F(x)), Y) \Rightarrow F(Y)$$

Goal: $\mathtt{A}^{\star}_{\sigma}(\mathtt{L}(\lambda x.F(x)), Y) \succeq Y$

$$\begin{array}{lcl} \mathtt{A}^{\sigma,\tau} & : & [(\sigma \to \tau) \times \sigma] \to \tau \\ \mathtt{L}^{\sigma,\tau} & : & [\sigma \to \tau] \to \sigma \to \tau \end{array}$$

$$\mathtt{A}^{\sigma,\tau}(\mathtt{L}^{\sigma,\tau}(\lambda x.F(x)), Y) \Rightarrow F(Y)$$

Goal: $\mathtt{A}^{\star}_{\sigma}(\mathtt{L}(\lambda x.F(x)), Y) \succeq Y$
Because: (2)

- $Y \succeq Y$ (by 9)

$$\begin{aligned} \mathtt{A}^{\sigma,\tau} &: [(\sigma \to \tau) \times \sigma] \to \tau \\ \mathtt{L}^{\sigma,\tau} &: [\sigma \to \tau] \to \sigma \to \tau \end{aligned}$$

$$\mathtt{A}^{\sigma,\tau}(\mathtt{L}^{\sigma,\tau}(\lambda x.F(x)), Y) \Rightarrow F(Y)$$

Termination proved!

First-order Dependency Pairs

[Arts and Giesl 1997]
rewrite rules:

$$\begin{array}{rcl} \mathtt{A}(\mathtt{0}, n) & \Rightarrow & \mathtt{s}(n) \\ \mathtt{A}(\mathtt{s}(m), \mathtt{0}) & \Rightarrow & \mathtt{A}(m, \mathtt{s}(\mathtt{0})) \\ \mathtt{A}(\mathtt{s}(m), \mathtt{s}(n)) & \Rightarrow & \mathtt{A}(m, \mathtt{A}(\mathtt{s}(m), n)) \end{array}$$

dependency pairs:

$$\begin{array}{lrcl} 1. & \mathtt{A}^\sharp(\mathtt{s}(m), \mathtt{0}) & \Rightarrow & \mathtt{A}^\sharp(m, \mathtt{s}(\mathtt{0})) \\ 2. & \mathtt{A}^\sharp(\mathtt{s}(m), \mathtt{s}(n)) & \Rightarrow & \mathtt{A}^\sharp(m, \mathtt{A}(\mathtt{s}(m), n)) \\ 3. & \mathtt{A}^\sharp(\mathtt{s}(m), \mathtt{s}(n)) & \Rightarrow & \mathtt{A}^\sharp(\mathtt{s}(m), n) \end{array}$$

infinite reduction $\Leftrightarrow$ infinite dependency chain

First-order Dependency Pairs

[Arts and Giesl 1997]
rewrite rules:

$$
\begin{array}{rcl}
\mathtt{A}(0, n) & \Rightarrow & \mathtt{s}(n) \\
\mathtt{A}(\mathtt{s}(m), 0) & \Rightarrow & \mathtt{A}(m, \mathtt{s}(0)) \\
\mathtt{A}(\mathtt{s}(m), \mathtt{s}(n)) & \Rightarrow & \mathtt{A}(m, \mathtt{A}(\mathtt{s}(m), n))
\end{array}
$$

dependency pairs:

$$
\begin{array}{lrcl}
1. & \mathtt{A}^\sharp(\mathtt{s}(m), 0) & \Rightarrow & \mathtt{A}^\sharp(m, \mathtt{s}(0)) \\
2. & \mathtt{A}^\sharp(\mathtt{s}(m), \mathtt{s}(n)) & \Rightarrow & \mathtt{A}^\sharp(m, \mathtt{A}(\mathtt{s}(m), n)) \\
3. & \mathtt{A}^\sharp(\mathtt{s}(m), \mathtt{s}(n)) & \Rightarrow & \mathtt{A}^\sharp(\mathtt{s}(m), n)
\end{array}
$$

infinite reduction $\Leftrightarrow$ infinite dependency chain

First-order Dependency Pairs

[Arts and Giesl 1997]

rewrite rules:

$$\begin{array}{rcl} \mathtt{A}(\mathtt{0}, n) & \Rightarrow & \mathtt{s}(n) \\ \mathtt{A}(\mathtt{s}(m), \mathtt{0}) & \Rightarrow & \mathtt{A}(m, \mathtt{s}(\mathtt{0})) \\ \mathtt{A}(\mathtt{s}(m), \mathtt{s}(n)) & \Rightarrow & \mathtt{A}(m, \mathtt{A}(\mathtt{s}(m), n)) \end{array}$$

dependency pairs:

$$\begin{array}{lrcl} 1. & \mathtt{A}^\sharp(\mathtt{s}(m), \mathtt{0}) & \Rightarrow & \mathtt{A}^\sharp(m, \mathtt{s}(\mathtt{0})) \\ 2. & \mathtt{A}^\sharp(\mathtt{s}(m), \mathtt{s}(n)) & \Rightarrow & \mathtt{A}^\sharp(m, \mathtt{A}(\mathtt{s}(m), n)) \\ 3. & \mathtt{A}^\sharp(\mathtt{s}(m), \mathtt{s}(n)) & \Rightarrow & \mathtt{A}^\sharp(\mathtt{s}(m), n) \end{array}$$

infinite reduction $\Leftrightarrow$ infinite dependency chain

First-order Dependency Pairs

[Arts and Giesl 1997]
rewrite rules:

$$
\begin{array}{rcl}
\mathtt{A}(\mathtt{0}, n) & \Rightarrow & \mathtt{s}(n) \\
\mathtt{A}(\mathtt{s}(m), \mathtt{0}) & \Rightarrow & \mathtt{A}(m, \mathtt{s}(\mathtt{0})) \\
\mathtt{A}(\mathtt{s}(m), \mathtt{s}(n)) & \Rightarrow & \mathtt{A}(m, \mathtt{A}(\mathtt{s}(m), n))
\end{array}
$$

dependency pairs:

$$
\begin{array}{lrcl}
1. & \mathtt{A}^\sharp(\mathtt{s}(m), \mathtt{0}) & \Rrightarrow & \mathtt{A}^\sharp(m, \mathtt{s}(\mathtt{0})) \\
2. & \mathtt{A}^\sharp(\mathtt{s}(m), \mathtt{s}(n)) & \Rrightarrow & \mathtt{A}^\sharp(m, \mathtt{A}(\mathtt{s}(m), n)) \\
3. & \mathtt{A}^\sharp(\mathtt{s}(m), \mathtt{s}(n)) & \Rrightarrow & \mathtt{A}^\sharp(\mathtt{s}(m), n)
\end{array}
$$

infinite reduction $\Leftrightarrow$ infinite dependency chain

Higher-order Dependency Pairs

$$
\begin{array}{rcl}
\mathtt{I}(\mathtt{0}) & \Rightarrow & \mathtt{0} \\
\mathtt{I}(\mathtt{s}(n)) & \Rightarrow & \mathtt{s}(\mathtt{twice}(\lambda x.\mathtt{I}(x), n)) \\
\mathtt{twice}(F, n) & \Rightarrow & F \cdot (F \cdot n)
\end{array}
$$

Two styles of Dependency Pairs!

Dynamic Dependency Pairs:

$$
\begin{array}{rcl}
\mathtt{I}^\sharp(\mathtt{s}(n)) & \Rightarrow & \mathtt{twice}^\sharp(\lambda x.\mathtt{I}(x), n) \\
\mathtt{I}^\sharp(\mathtt{s}(n)) & \Rightarrow & \mathtt{I}^\sharp(\mathtt{c}_x) \\
\mathtt{twice}^\sharp(F, n) & \Rightarrow & F \cdot (F \cdot n) \\
\mathtt{twice}^\sharp(F, n) & \Rightarrow & F \cdot n
\end{array}
$$

Static Dependency Pairs:

$$
\begin{array}{rcl}
\mathtt{I}^\sharp(\mathtt{s}(n)) & \Rightarrow & \mathtt{twice}^\sharp(\lambda x.\mathtt{I}(x), n) \\
\mathtt{I}^\sharp(\mathtt{s}(n)) & \Rightarrow & \mathtt{I}^\sharp(x)
\end{array}
$$

Higher-order Dependency Pairs

$$
\begin{aligned}
\mathtt{I}(\mathtt{0}) &\Rightarrow \mathtt{0} \\
\mathtt{I}(\mathtt{s}(n)) &\Rightarrow \mathtt{s}(\mathtt{twice}(\lambda x.\mathtt{I}(x), n)) \\
\mathtt{twice}(F, n) &\Rightarrow F \cdot (F \cdot n)
\end{aligned}
$$

Two styles of Dependency Pairs!

Dynamic Dependency Pairs:

$$
\begin{aligned}
\mathtt{I}^\sharp(\mathtt{s}(n)) &\Rightarrow \mathtt{twice}^\sharp(\lambda x.\mathtt{I}(x), n) \\
\mathtt{I}^\sharp(\mathtt{s}(n)) &\Rightarrow \mathtt{I}^\sharp(\mathtt{c}_x) \\
\mathtt{twice}^\sharp(F, n) &\Rightarrow F \cdot (F \cdot n) \\
\mathtt{twice}^\sharp(F, n) &\Rightarrow F \cdot n
\end{aligned}
$$

Static Dependency Pairs:

$$
\begin{aligned}
\mathtt{I}^\sharp(\mathtt{s}(n)) &\Rightarrow \mathtt{twice}^\sharp(\lambda x.\mathtt{I}(x), n) \\
\mathtt{I}^\sharp(\mathtt{s}(n)) &\Rightarrow \mathtt{I}^\sharp(x)
\end{aligned}
$$

Higher-order Dependency Pairs

$$
\begin{array}{rcl}
\mathtt{I}(\mathtt{0}) & \Rightarrow & \mathtt{0} \\
\mathtt{I}(\mathtt{s}(n)) & \Rightarrow & \mathtt{s}(\mathtt{twice}(\lambda x.\mathtt{I}(x), n)) \\
\mathtt{twice}(F, n) & \Rightarrow & F \cdot (F \cdot n)
\end{array}
$$

Two styles of Dependency Pairs!

Dynamic Dependency Pairs:

$$
\begin{array}{rcl}
\mathtt{I}^\sharp(\mathtt{s}(n)) & \Rightarrow & \mathtt{twice}^\sharp(\lambda x.\mathtt{I}(x), n) \\
\mathtt{I}^\sharp(\mathtt{s}(n)) & \Rightarrow & \mathtt{I}^\sharp(\mathtt{c}_x) \\
\mathtt{twice}^\sharp(F, n) & \Rightarrow & F \cdot (F \cdot n) \\
\mathtt{twice}^\sharp(F, n) & \Rightarrow & F \cdot n
\end{array}
$$

Static Dependency Pairs:

$$
\begin{array}{rcl}
\mathtt{I}^\sharp(\mathtt{s}(n)) & \Rightarrow & \mathtt{twice}^\sharp(\lambda x.\mathtt{I}(x), n) \\
\mathtt{I}^\sharp(\mathtt{s}(n)) & \Rightarrow & \mathtt{I}^\sharp(x)
\end{array}
$$

Higher-order Dependency Pairs

$$\begin{array}{rcl} \mathtt{I}(\mathtt{0}) & \Rightarrow & \mathtt{0} \\ \mathtt{I}(\mathtt{s}(n)) & \Rightarrow & \mathtt{s}(\mathtt{twice}(\lambda x.\mathtt{I}(x), n)) \\ \mathtt{twice}(F, n) & \Rightarrow & F \cdot (F \cdot n) \end{array}$$

Two styles of Dependency Pairs!

Dynamic Dependency Pairs:

$$\begin{array}{rcl} \mathtt{I}^\sharp(\mathtt{s}(n)) & \Rightarrow & \mathtt{twice}^\sharp(\lambda x.\mathtt{I}(x), n) \\ \mathtt{I}^\sharp(\mathtt{s}(n)) & \Rightarrow & \mathtt{I}^\sharp(\mathtt{c}_x) \\ \mathtt{twice}^\sharp(F, n) & \Rightarrow & F \cdot (F \cdot n) \\ \mathtt{twice}^\sharp(F, n) & \Rightarrow & F \cdot n \end{array}$$

Static Dependency Pairs:

$$\begin{array}{rcl} \mathtt{I}^\sharp(\mathtt{s}(n)) & \Rightarrow & \mathtt{twice}^\sharp(\lambda x.\mathtt{I}(x), n) \\ \mathtt{I}^\sharp(\mathtt{s}(n)) & \Rightarrow & \mathtt{I}^\sharp(x) \end{array}$$

Higher-order Dependency Pairs

$$\begin{array}{rcl} \mathtt{I}(\mathtt{0}) & \Rightarrow & \mathtt{0} \\ \mathtt{I}(\mathtt{s}(n)) & \Rightarrow & \mathtt{s}(\mathtt{twice}(\lambda x.\mathtt{I}(x), n)) \\ \mathtt{twice}(F, n) & \Rightarrow & F \cdot (F \cdot n) \end{array}$$

Two styles of Dependency Pairs!

Dynamic Dependency Pairs:

$$\begin{array}{rcl} \mathtt{I}^\sharp(\mathtt{s}(n)) & \Rrightarrow & \mathtt{twice}^\sharp(\lambda x.\mathtt{I}(x), n) \\ \mathtt{I}^\sharp(\mathtt{s}(n)) & \Rrightarrow & \mathtt{I}^\sharp(\mathtt{c}_x) \\ \mathtt{twice}^\sharp(F, n) & \Rrightarrow & F \cdot (F \cdot n) \\ \mathtt{twice}^\sharp(F, n) & \Rrightarrow & F \cdot n \end{array}$$

Static Dependency Pairs:

$$\begin{array}{rcl} \mathtt{I}^\sharp(\mathtt{s}(n)) & \Rrightarrow & \mathtt{twice}^\sharp(\lambda x.\mathtt{I}(x), n) \\ \mathtt{I}^\sharp(\mathtt{s}(n)) & \Rrightarrow & \mathtt{I}^\sharp(x) \end{array}$$

Higher-order Dependency Pairs

$$\begin{array}{rcl} \mathtt{I}(\mathtt{0}) & \Rightarrow & \mathtt{0} \\ \mathtt{I}(\mathtt{s}(n)) & \Rightarrow & \mathtt{s}(\mathtt{twice}(\lambda x.\mathtt{I}(x), n)) \\ \mathtt{twice}(F, n) & \Rightarrow & F \cdot (F \cdot n) \end{array}$$

Two styles of Dependency Pairs!

Dynamic Dependency Pairs:

$$\begin{array}{rcl} \mathtt{I}^\sharp(\mathtt{s}(n)) & \Rrightarrow & \mathtt{twice}^\sharp(\lambda x.\mathtt{I}(x), n) \\ \mathtt{I}^\sharp(\mathtt{s}(n)) & \Rrightarrow & \mathtt{I}^\sharp(\mathtt{c}_x) \\ \mathtt{twice}^\sharp(F, n) & \Rrightarrow & F \cdot (F \cdot n) \\ \mathtt{twice}^\sharp(F, n) & \Rrightarrow & F \cdot n \end{array}$$

Static Dependency Pairs:

$$\begin{array}{rcl} \mathtt{I}^\sharp(\mathtt{s}(n)) & \Rrightarrow & \mathtt{twice}^\sharp(\lambda x.\mathtt{I}(x), n) \\ \mathtt{I}^\sharp(\mathtt{s}(n)) & \Rrightarrow & \mathtt{I}^\sharp(x) \end{array}$$

Higher-order Dependency Pairs

argument filterings remain powerful (assuming some restrictions)

dependency graph relatively weak in the presence of collapsing DPs (but can still be used, and extended with bound-variable occurrence checks)

usable rules relatively weak in the presence of collapsing rules (but can be useful in particular with static dependency pairs, and extended with type information)

formative rules symmetric to usable rules, rely on type information

first-order removal if the first-order part is terminating, its dependency pairs need not be considered (given certain constraints)

Higher-order Dependency Pairs

argument filterings remain powerful (assuming some restrictions)

dependency graph relatively weak in the presence of collapsing DPs (but can still be used, and extended with bound-variable occurrence checks)

usable rules relatively weak in the presence of collapsing rules (but can be useful in particular with static dependency pairs, and extended with type information)

formative rules symmetric to usable rules, rely on type information

first-order removal if the first-order part is terminating, its dependency pairs need not be considered (given certain constraints)

Higher-order Dependency Pairs

argument filterings remain powerful (assuming some restrictions)

dependency graph relatively weak in the presence of collapsing DPs (but can still be used, and extended with bound-variable occurrence checks)

usable rules relatively weak in the presence of collapsing rules (but can be useful in particular with static dependency pairs, and extended with type information)

formative rules symmetric to usable rules, rely on type information

first-order removal if the first-order part is terminating, its dependency pairs need not be considered (given certain constraints)

Higher-order Dependency Pairs

argument filterings remain powerful (assuming some restrictions)

dependency graph relatively weak in the presence of collapsing DPs (but can still be used, and extended with bound-variable occurrence checks)

usable rules relatively weak in the presence of collapsing rules (but can be useful in particular with static dependency pairs, and extended with type information)

formative rules symmetric to usable rules, rely on type information

first-order removal if the first-order part is terminating, its dependency pairs need not be considered (given certain constraints)

Higher-order Dependency Pairs

argument filterings remain powerful (assuming some restrictions)

dependency graph relatively weak in the presence of collapsing DPs (but can still be used, and extended with bound-variable occurrence checks)

usable rules relatively weak in the presence of collapsing rules (but can be useful in particular with static dependency pairs, and extended with type information)

formative rules symmetric to usable rules, rely on type information

first-order removal if the first-order part is terminating, its dependency pairs need not be considered (given certain constraints)

Outline

Conclusions

- first-order termination techniques can often be generalised
  often even in more than one way!
- specific methods for higher-order rewriting can also be designed
  in particular: transformations
- automatability: similar to the first-order case
  although higher risk of exponential blow-up

Conclusions

- first-order termination techniques can often be generalised
  often even in more than one way!
- specific methods for higher-order rewriting can also be designed
  in particular: transformations
- automatability: similar to the first-order case
  although higher risk of exponential blow-up

Conclusions

- first-order termination techniques can often be generalised
  often even in more than one way!
- specific methods for higher-order rewriting can also be designed
  in particular: transformations
- automatability: similar to the first-order case
  although higher risk of exponential blow-up

Conclusions

- first-order termination techniques can often be generalised
  often even in more than one way!
- specific methods for higher-order rewriting can also be designed
  in particular: transformations
- automatability: similar to the first-order case
  although higher risk of exponential blow-up

Conclusions

- first-order termination techniques can often be generalised
  often even in more than one way!
- specific methods for higher-order rewriting can also be designed
  in particular: transformations
- automatability: similar to the first-order case
  although higher risk of exponential blow-up

Conclusions

- first-order termination techniques can often be generalised
  often even in more than one way!
- specific methods for higher-order rewriting can also be designed
  in particular: transformations
- automatability: similar to the first-order case
  although higher risk of exponential blow-up

Future Work

- further extension of first-order techniques
  - narrowing (non-termination)
  - usable and formative rules wrt an argument filtering
  - max-interpretations
- extension to more complicated type systems
  - polymorphism
  - dependent types
- using WANDA for real applications

Questions?

Future Work

- further extension of first-order techniques
  - narrowing (non-termination)
  - usable and formative rules wrt an argument filtering
  - max-interpretations
- extension to more complicated type systems
  - polymorphism
  - dependent types
- using WANDA for real applications

Questions?

Future Work

- further extension of first-order techniques
  - narrowing (non-termination)
  - usable and formative rules wrt an argument filtering
  - max-interpretations
- extension to more complicated type systems
  - polymorphism
  - dependent types
- using WANDA for real applications

Questions?

Future Work

- further extension of first-order techniques
  - narrowing (non-termination)
  - usable and formative rules wrt an argument filtering
  - max-interpretations
- extension to more complicated type systems
  - polymorphism
  - dependent types
- using WANDA for real applications

Questions?

Future Work

- further extension of first-order techniques
    - narrowing (non-termination)
    - usable and formative rules wrt an argument filtering
    - max-interpretations
- extension to more complicated type systems
    - polymorphism
    - dependent types
- using WANDA for real applications

Questions?

Future Work

- further extension of first-order techniques
  - narrowing (non-termination)
  - usable and formative rules wrt an argument filtering
  - max-interpretations
- extension to more complicated type systems
  - polymorphism
  - dependent types
- using WANDA for real applications

Questions?