5. Automation

22

Implementing automatic HORPO proof search

Needed: status, precedence, which clause to apply when

Strategy: use existing SAT or SMT solvers!

In the past, solvers would implement their own proof search, but with the quality of SAT and SMT solvers, this would not be the most efficient approach – neither in terms of implementation time nor in terms of executation time. Instead, we would implement the problem as a boolean formula, whose satisfiability implies that the HORPO proof succeeds.

Idea:

- for each function symbol: an **integer value** for the precedence
- for each function symbol: an **integer value** for the status
- for each HORPO relation we encounter: a **boolean variable**

23

Example: encoding proof search for `map`

Formula:

- v_1
- $(v_1 \rightarrow v_2 \vee v_3 \vee v_4 \vee v_5 \vee v_6) \wedge$
- $(v_2 \rightarrow v_7 \vee v_8) \wedge$
- $(v_3 \rightarrow (\mathtt{prec}_{\mathtt{map}} > \mathtt{prec}_{\mathtt{cons}} \wedge v_9 \wedge v_{10})) \wedge$
- $\dots$

Variables:

- $v_1 \equiv$ “$\mathtt{map}(F, \mathtt{cons}(x, y)) \succ_{\text{LPO}} \mathtt{cons}(F \cdot x, \mathtt{map}(F, y))$”
- $v_2 \equiv$ “$\mathtt{map}(F, \mathtt{cons}(x, y)) \sqsupset^{\emptyset}_{\text{LPO}} \mathtt{cons}(F \cdot x, \mathtt{map}(F, y))$ by **(sub)**”
- $v_3 \equiv$ “$\mathtt{map}(F, \mathtt{cons}(x, y)) \sqsupset^{\emptyset}_{\text{LPO}} \mathtt{cons}(F \cdot x, \mathtt{map}(F, y))$ by **(copy)**”
- $v_4 \equiv$ “$\mathtt{map}(F, \mathtt{cons}(x, y)) \sqsupset^{\emptyset}_{\text{LPO}} \mathtt{cons}(F \cdot x, \mathtt{map}(F, y))$ by **(lex)**”
- $v_5 \equiv$ “$\mathtt{map}(F, \mathtt{cons}(x, y)) \sqsupset^{\emptyset}_{\text{LPO}} \mathtt{cons}(F \cdot x, \mathtt{map}(F, y))$ by **(mul)**”
- $v_6 \equiv$ “$\mathtt{map}(F, \mathtt{cons}(x, y)) \sqsupset^{\emptyset}_{\text{LPO}} \mathtt{cons}(F \cdot x, \mathtt{map}(F, y))$ by **(app)**”
- $v_7 \equiv$ “$F \succ_{\text{LPO}} \mathtt{cons}(F \cdot x, \mathtt{map}(F, y))$”
- $v_8 \equiv$ “$\mathtt{cons}(x, y) \succ_{\text{LPO}} \mathtt{cons}(F \cdot x, \mathtt{map}(F, y))$”
- $v_9 \equiv$ “$\mathtt{map}(F, \mathtt{cons}(x, y)) \sqsupset^{\emptyset}_{\text{LPO}} F \cdot x$”
- $v_{10} \equiv$ “$\mathtt{map}(F, \mathtt{cons}(x, y)) \sqsupset^{\emptyset}_{\text{LPO}} \mathtt{map}(F, y)$”
- $\dots$

1

Handout for part 3:

Termination: the higher-order recursive path ordering

1. Reduction ordering

How to prove termination?

1. Embed the rewrite relation $\to_{\mathcal{R}}$ in a well-founded ordering.

   (Because then any infinite reduction $s_1 \to s_2 \to s_3 \to \dots$ is an infinite decreasing sequence $s_1 \succ s_2 \succ s_3 \succ \dots$, contradicting well-foundedness.)

2. Pay special attention to function calls
   (Use some form of the *dependency pair framework*.)

To start: we will define a **well-founded ordering**
There are many ways to find such an ordering! In this course, we will consider two of the most popular.

Embedding $\to_{\mathcal{R}}$ in a well-founded ordering

Goal: find a **well-founded ordering** $\succ$ and prove that $s \succ t$ whenever $s \to t$.

Difficulty: how to prove $s \succ t$ whenever $s \to t$? There are infinitely many terms and possible reductions.

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

Needed: $\mathtt{add}(\mathtt{0},\mathtt{0}) \succ \mathtt{0}$, $\mathtt{add}(\mathtt{0},\mathtt{add}(x,y)) \succ \mathtt{add}(x,y), \dots$

Solution: it suffices to orient the **rules** if we have a well-founded ordering $\succ$ with:

- if $s \succ t$ then $s\sigma \succ t\sigma$ for all substitutions σ
  (we say: $\succ$ is **stable**)
- if $s \succ t$ then $\mathtt{f}(\dots,s,\dots) \succ \mathtt{f}(\dots,t,\dots)$ for all $\mathtt{f}$
  (we say: $\succ$ is **monotonic**)

Such an ordering is called a **reduction ordering**.

3. if $s :: \sigma$ is computable and $s \succ_{\text{LPO}} t$ then t is computable

Proof: by **shared** induction on σ (class exercise)

Soundness of HORPO

Main proof ideas:

- if $s[x := t]$ is computable for all computable t, then $\lambda x.s$ computable
- if $s_1, \dots, s_k$ **computable**, and $\mathtt{f}(s_1,\dots,s_k) \sqsupset^{X}_{\text{LPO}} t$,
  then $t[\vec{x} := \vec{u}]$ is computable for all computable $\vec{u}$

  (by induction first on $\mathtt{f}$,
  then on $(s_1,\dots,s_k)$ ordered with *status*$(\mathtt{f})$,
  and finally on the derivation of $\mathtt{f}(s_1,\dots,s_k) \sqsupset^{X}_{\text{LPO}} t$)

4. Computability

18

Challenge: well-foundedness of HORPO

Recall: the well-foundedness proof of RPO was based on the argument:

$$\text{if } (s_1,\dots,s_n) \text{ terminating, and } \mathtt{f}(s_1,\dots,s_n) \succ_{\text{LPO}} t, \text{ then } t \text{ terminating}$$

Problem: termination of, e.g., $\mathtt{map}(F,\mathtt{cons}(x,y))$ depends on the **behaviour** of F.

While F, x and y could all be instantiated by terminating terms, what we really need to know is if the *function* F terminates *when applied to some arbitrary input.*

Example:

$$\begin{array}{rcl} \mathtt{map}(F,[]) & \to & [] \\ \mathtt{map}(F,\mathtt{cons}(x,y)) & \to & \mathtt{cons}(F\cdot x,\mathtt{map}(F,y)) \\ \mathtt{f}\ x & \to & \mathtt{f}\ (\mathtt{s}\ x) \end{array}$$

Although $\mathtt{f}$, $\mathtt{0}$ and $[]$ are all terminating, $\mathtt{map}(\mathtt{f},\mathtt{cons}(\mathtt{0},[]))$ is not.

Of course, this isn't a major problem, because the termination proof will fail regardless on the rule $\mathtt{f}\ x \to \mathtt{f}\ (\mathtt{s}\ x)$. That is, if everything else is terminating, then so is the $\mathtt{map}$ function. This leads to the idea of *computability.*

19

Solution: computability

Definition

- a term s of **base type** is *computable* if s is terminating (under $\succ_{\text{LPO}}$)

- a term s of type $\sigma \Rightarrow \tau$ is computable if
  for all computable t of type σ
  the term $s \cdot t$ (of type τ) is also computable

(This is well-defined by induction on types.)

Computability can be seen as a higher-order version of termination. (Although with disclaimers – there are different definitions of computability, and computability cannot take the place of termination in all proofs.)

20

Properties of computability

Claim: for all types σ:

1. all variables of type σ are computable

2. every computable term of type σ is terminating

2. RPO

4

The (first-order) lexicographic path ordering

A very powerful technique, with many extensions and variations, is the **lexicographic path ordering**.

Let $\rhd$ be a **total, well-founded ordering** on the function symbols.

We define: $\mathtt{f}(s_1,\dots,s_n) \succ_{\text{LPO}} t$ if one of the following holds:

(sub) $s_i \succeq_{\text{LPO}} t$ for some $i \in \{1,\dots,n\}$
(that is, $s_i \succ t$ or $s_i = t$)

(copy) $t = \mathtt{g}(t_1,\dots,t_m)$ and $\mathtt{f} \rhd \mathtt{g}$ and $\mathtt{f}(s_1,\dots,s_n) \succ_{\text{LPO}} t_j$ for all $j \in \{1,\dots,m\}$

(lex) $t = \mathtt{f}(t_1,\dots,t_n)$ and $\mathtt{f}(s_1,\dots,s_n) \succ_{\text{LPO}} t_i$ for all $i \in \{1,\dots,n\}$,
and $[s_1,\dots,s_n](\succ_{\text{LPO}})_{\text{lex}}[t_1,\dots,t_n]$;
that is, there is some $i \in \{1,\dots,n\}$ such that:

- $s_j = t_j$ for $j \in \{1,\dots,i-1\}$
- $s_i \succ_{\text{LPO}} t_i$

5

LPO example

$$\begin{array}{rcl} \mathtt{add}(\mathtt{0},y) & \to & y \\ \mathtt{add}(\mathtt{s}(x),y) & \to & \mathtt{s}(\mathtt{add}(x,y)) \\ \mathtt{mul}(\mathtt{0},y) & \to & \mathtt{0} \\ \mathtt{mul}(\mathtt{s}(x),y) & \to & \mathtt{add}(y,\mathtt{mul}(x,y)) \end{array}$$

We choose: $\mathtt{mul} \rhd \mathtt{add} \rhd \mathtt{s} \rhd \mathtt{0}$

We orient the last rule as follows:

A.	$\mathtt{mul}(\mathtt{s}(x),y)$	$\succ_{\text{LPO}}$	$\mathtt{add}(y,\mathtt{mul}(x,y))$	by **(copy)**, B, D
B.	$\mathtt{mul}(\mathtt{s}(x),y)$	$\succ_{\text{LPO}}$	y	by **(sub)**, C
C.	y	$\succeq_{\text{LPO}}$	y	by definition
D.	$\mathtt{mul}(\mathtt{s}(x),y)$	$\succ_{\text{LPO}}$	$\mathtt{mul}(x,y)$	by **(lex)**, E, B, F
E.	$\mathtt{mul}(\mathtt{s}(x),y)$	$\succ_{\text{LPO}}$	x	by **(sub)**, F
F.	$\mathtt{s}(x)$	$\succ_{\text{LPO}}$	x	by **(sub)**, G
G.	x	$\succeq_{\text{LPO}}$	x	by definition

6

Exercise

Use LPO to prove termination of the well-known **Ackermann function**, defined by:

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

Note: it speaks to the power of LPO that we can indeed use it to prove termination of the Ackermann function. After all, while termination theoretically holds, the normal form of $\mathtt{A}(\mathtt{s}(\mathtt{s}(\mathtt{s}(\mathtt{s}(\mathtt{0})))),\mathtt{s}(\mathtt{s}(\mathtt{0})))$ is a term containing N symbols $\mathtt{s}$, for N being a number of 19,729 decimal digits. Hence, this normal form does not fit in all computer memory of the world.

7

Soundness of LPO

Theorem

If $\ell \succ_{\text{LPO}} r$ for all rules in $\mathcal{R}$, then the TRS with rules $\mathcal{R}$ is terminating.

Proof. $\succ_{\text{LPO}}$ is:

- stable: if $s \succ_{\text{LPO}} t$ then $s\sigma \succ_{\text{LPO}} t\sigma$
  (by a simple induction on the definition: if $x \succeq_{\text{LPO}} t$ then $t = x$, so $x\sigma = t\sigma$ too)
- monotonic: if $s \succ_{\text{LPO}} t$ then $\mathtt{f}(\dots,s,\dots) \succ_{\text{LPO}} \mathtt{f}(\dots,t,\dots)$
  (by the **(lex)** rule)
- well-founded: there is no infinite decreasing sequence

8

Well-foundedness of LPO

Define: s is **terminating** if there is no infinite sequence $s \succ_{\text{LPO}} t_1 \succ_{\text{LPO}} t_2 \succ_{\text{LPO}} \dots$ starting in s.

Claim: if $(s_1,\dots,s_n)$ terminating, and $\mathtt{f}(s_1,\dots,s_n) \succ_{\text{LPO}} t$, then t terminating

Proof: by induction on:

- $\mathtt{f}$ first (using $\rhd$)
- $(s_1,\dots,s_n)$ ordered lexicographically by $\succ_{\text{LPO}}$ second;
- the derivation of $\mathtt{f}(s_1,\dots,s_n) \succ_{\text{LPO}} t$ third

Conclude: if there is a smallest non-terminating $\mathtt{f}(s_1,\dots,s_n)$, then by definition of "smallest", all s_i are terminating; therefore, if there is an infinite sequence $\mathtt{f}(s_1,\dots,s_n) \succ_{\text{LPO}} t_1 \succ_{\text{LPO}} \dots$ then t_1 is terminating as we saw above. But this contradicts the existence of the infinite sequence!

9

Extending LPO

In practice, the lexicographic path ordering is quite minimalistic – and often we can do better.

Polymorphic HOLPO

Idea: be creative with the type collapsing!

Instead of mapping each base type to $\mathtt{o}$, it is actually sound to replace base types by any type, so long as we do it consistently. Thus, we can for instance let:

$$\mathtt{collapse}(\mathsf{list}(\alpha)) := \mathtt{collapse}(\alpha) \quad \text{for all types } \alpha$$

Then to prove termination of all type-instances of the polymorphic $\mathtt{map}$ rule at once, it suffices to orient the following rule using HOLPO:

$$\begin{array}{rcl} \mathtt{cons}_1 & :: & \alpha \Rightarrow \alpha \\ \mathtt{cons}_2 & :: & \beta \Rightarrow \beta \\ \mathtt{map} & :: & (\alpha \Rightarrow \beta) \Rightarrow \alpha \Rightarrow \beta \end{array}$$

$$\mathtt{map}(F_{\alpha\Rightarrow\beta},\mathtt{cons}_1(x_\alpha,y_\alpha)) \to \mathtt{cons}_2(F_{\alpha\Rightarrow\beta}\cdot x_\alpha,\mathtt{map}(F_{\alpha\Rightarrow\beta},y_\alpha))$$

This can be oriented with the same proof as we saw before!

17

HORPO

As in the first-order setting, the higher-order lexicographic path ordering can be extended with status and distinct function symbols being equated. This for example gives rules like:

(mul) $s = \mathtt{f}(s_1,\dots,s_k) \sqsupset^X_{\text{LPO}} \mathtt{g}(t_1,\dots,t_n)$ if

- $\mathtt{f} \approx \mathtt{g}$
- $\mathit{status}(\mathtt{f}) = \mathit{mul}_m$ for some $m \in \mathbf{N}$ with $m \le n$
- $\mathtt{f}(s_1,\dots,s_k) \sqsupset^X_{\text{LPO}} t_i$ for all $i \in \{1,\dots,n\}$
- $\{\{s_1,\dots,s_{\min(k,m)}\}\}\ (\succ_{\text{LPO}})_{\text{mul}}\ \{\{t_1,\dots,t_m\}\}$

1. $\mathtt{map}(F, \mathtt{cons}(x, y)) \succ_{\text{LPO}} \mathtt{cons}(F \cdot x, \mathtt{map}(F, y))$
   by **(greater)**, 2
2. $\mathtt{map}(F, \mathtt{cons}(x, y)) \sqsupset^{\emptyset}_{\text{LPO}} \mathtt{cons}(F \cdot x, \mathtt{map}(F, y))$
   by **(copy)**, $\mathtt{map} \rhd \mathtt{cons}$, 3, 4
3. $\mathtt{map}(F, \mathtt{cons}(x, y)) \sqsupset^{\emptyset}_{\text{LPO}} F \cdot x$
   by **(app)**, 7, 8
4. $\mathtt{map}(F, \mathtt{cons}(x, y)) \sqsupset^{\emptyset}_{\text{LPO}} \mathtt{map}(F, y)$
   by **(lex)**, $F \succeq_{\text{LPO}} F$, 5 (typecheck: o)
5. $\mathtt{cons}(x, y)) \succeq_{\text{LPO}} y$
   by **(greater)**, 6
6. $\mathtt{cons}(x, y)) \sqsupset^{\emptyset}_{\text{LPO}} y$
   by **(sub)**, $y \succeq_{\text{LPO}} y$
7. $\mathtt{map}(F, \mathtt{cons}(x, y)) \sqsupset^{\emptyset}_{\text{LPO}} F$
   by **(sub)**, $F \succeq_{\text{LPO}} F$
8. $\mathtt{map}(F, \mathtt{cons}(x, y)) \sqsupset^{\emptyset}_{\text{LPO}} x$
   by **(sub)**, 9 (typecheck: o)
9. $\mathtt{cons}(x, y)) \succeq x$
   by **(greater)**, 10
10. $\mathtt{cons}(x, y)) \sqsupset^{\emptyset}_{\text{LPO}} x$
    by **(sub)**, $x \succeq_{\text{LPO}} x$

Exercise

Orient the following rules using HOLPO:

$$\begin{array}{rcl} \mathtt{start} & :: & \mathsf{o} \Rightarrow \mathsf{o} \\ \mathtt{add} & :: & \mathsf{o} \Rightarrow \mathsf{o} \Rightarrow \mathsf{o} \\ \mathtt{map} & :: & (\mathsf{o} \Rightarrow \mathsf{o}) \Rightarrow \mathsf{o} \Rightarrow \mathsf{o} \end{array}$$

$$\mathtt{start}(y) \to \mathtt{map}(\lambda x_{\mathsf{o}}.\mathtt{add}(x_{\mathsf{o}}, x_{\mathsf{o}}), y)$$

$$\begin{array}{rcl} \mathtt{a} & :: & \mathsf{o} \\ \mathtt{b} & :: & \mathsf{o} \\ \mathtt{f} & :: & ((\mathsf{o} \Rightarrow \mathsf{o}) \Rightarrow \mathsf{o}) \Rightarrow \mathsf{o} \end{array}$$

$$\mathtt{f}(\lambda x_{\mathsf{o}\Rightarrow\mathsf{o}}.x_{\mathsf{o}\Rightarrow\mathsf{o}} \cdot \mathtt{a}) \to \mathtt{f}(\lambda y_{\mathsf{o}\Rightarrow\mathsf{o}}.y_{\mathsf{o}\Rightarrow\mathsf{o}} \cdot \mathtt{b})$$

Challenge: mutual recursion

$$\begin{array}{rcl} \mathtt{f}(\mathtt{s}(x)) & \to & \mathtt{g}(x) \\ \mathtt{g}(\mathtt{s}(x)) & \to & \mathtt{f}(x) \end{array} \qquad \begin{array}{rcl} \mathtt{f}(x) & \to & \mathtt{s}(x) \\ \mathtt{f}(x) & \to & \mathtt{s}(x) \end{array}$$

Solution: allow an **equivalence relation** $\approx$ compatible with $\rhd$, and set $\mathtt{f} \approx \mathtt{g}$

Challenge: argument permutations

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

Solution: allow some function symbols to order arguments using the **multiset ordering**

This yields the **recursive path ordering** (RPO).

3. A higher-order RPO

10

Applying RPO to higher-order systems

The recursive path ordering is a powerful method, and we would like to apply it in higher-order rewriting as well. However, we run into a few challenges…

Challenge: $\mathtt{f}(\mathtt{g}(x)) \succ_{\mathrm{LPO}} x$

Why is this a problem? Well…

Recall: if

$$\mathtt{f} :: \mathsf{o} \Rightarrow \mathsf{o} \Rightarrow \mathsf{o} \text{ and}$$

$$\mathtt{g} :: (\mathsf{o} \Rightarrow \mathsf{o}) \Rightarrow \mathsf{o},$$

this is non-terminating as it encodes the untyped λ-calculus (we can see $\mathtt{f}$ as the application symbol, and $\mathtt{g}$ as a wrapper for abstractions)!

Nor is the problem with the **(sub)** rule the only one. The recursive path ordering has no functionality to deal with **applications**.

Challenge: how to derive $s \succ_{\mathrm{LPO}} F \cdot x$?

We could of course try encoding application as a function symbol – but this comes with all the problems we had before. Besides, this still does not solve all our problems.

Challenge: do we have $\mathtt{f}(s,t) \succ_{\mathrm{LPO}} @(\mathtt{f}(s),t)$ since $(s,t)(\succ_{\mathrm{LPO}})_{\mathrm{lex}}(s)$?

Conclusion:

- A dedicated higher-order definition is needed. Such a definition could take head-variables, lambda-abstraction and also partial application into account.
- Types are important for termination! Hence, we should take them into account in the definition of higher-order RPO.

11

HOLPO

So now, let us present a higher-order extension of the basic lexicographic path ordering.

- $s \succ_{\mathrm{LPO}} t$ if s and t **have the same type** and:

(greater) $s \sqsupset^X_{\mathrm{LPO}} t$

(@) $s = s_1 \cdot s_2$, $t = t_1 \cdot t_2$, each $s_i \succeq_{\mathrm{LPO}} t_i$, some $s_i \succ_{\mathrm{LPO}} t_i$
This could be used for instance to derive $x \cdot s \cdot t \cdot u \succ_{\mathrm{LPO}} x \cdot s \cdot t' \cdot u$ if $t \succ_{\mathrm{LPO}} t'$. We would typically *not* use it for applications with a function symbol at the head, since there it is more powerful to just use $\sqsupset_{\mathrm{LPO}}$ instead.

(lam) $s = \lambda x.s'$, $t = \lambda x.t'$ and $s' \succ_{\mathrm{LPO}} t'$
We can use α-renaming to make sure both variables are the same.

(beta) $s = (\lambda x.s') \cdot u_0 \cdots u_n$ and $s'[x := u_0] \cdot u_1 \cdots u_n \succeq_{\mathrm{LPO}} t$ (where $n \geq 0$)

- $\mathtt{f}(s_1,\dots,s_n) \sqsupset^X_{\mathrm{LPO}} t$ if:

(sub) $s_i \succeq_{\mathrm{LPO}} t$ for some $i \in \{1,\dots,n\}$ or $t \in X$
Note that $\succeq_{\mathrm{LPO}}$ is the reflexive closure of the type-conscious relation $\succ_{\mathrm{LPO}}$, so this implicitly requires that s_i and t have the same type.

(copy) $t = \mathtt{g}(t_1,\dots,t_m)$ and $\mathtt{f} \rhd \mathtt{g}$ and $\mathtt{f}(s_1,\dots,s_n) \sqsupset^X_{\mathrm{LPO}} t_j$ for all $j \in \{1,\dots,m\}$

(lex) $t = \mathtt{f}(t_1,\dots,t_n)$ and $\mathtt{f}(s_1,\dots,s_n) \sqsupset^X_{\mathrm{LPO}} t_i$ for all $i \in \{1,\dots,n\}$,
and $[s_1,\dots,s_n]\ (\succ_{\mathrm{LPO}})_{\mathrm{lex}}[t_1,\dots,t_n]$;
Note that this means that there is some $i \in \{1,\dots,n\}$ such that:

- $s_j = t_j$ for $j \in \{1,\dots,i-1\}$
- $s_i \succ_{\mathrm{LPO}} t_i$

We explicitly do **not** have $[s_1,\dots,s_{n+1}]\ (\succ_{\mathrm{LPO}})_{\mathrm{lex}}\ [s_1,\dots,s_n]$. Note also that for the arguments we use the type-sensitive comparison.

(app) $t = t_0 \cdot t_1 \cdots t_n$ and $\mathtt{f}(s_1,\dots,s_n) \sqsupset^X_{\mathrm{LPO}} t_i$ for all $i \in \{0,\dots,n\}$

(abs) $t = \lambda x.t'$ and $\mathtt{f}(s_1,\dots,s_n) \sqsupset^{X\cup\{x\}}_{\mathrm{LPO}} t'$

12

Collapsing types in HOLPO

$$\begin{aligned}
[] &: \mathsf{natlist}\\
\mathtt{cons} &: \mathsf{nat} \Rightarrow \mathsf{natlist} \Rightarrow \mathsf{natlist}\\
\mathtt{map} &: (\mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{natlist} \Rightarrow \mathsf{natlist}
\end{aligned}$$

$$\begin{aligned}
\mathtt{map}(F,[]) &\to []\\
\mathtt{map}(F,\mathtt{cons}(x,y)) &\to \mathtt{cons}(F \cdot x, \mathtt{map}(F,y))
\end{aligned}$$

Sometimes problematic: Not $\mathtt{cons}(x,y) \succ y$ due to types!

Solution:

$$\begin{aligned}
[] &: \mathsf{o}\\
\mathtt{cons} &: \mathsf{o} \Rightarrow \mathsf{o} \Rightarrow \mathsf{o}\\
\mathtt{map} &: (\mathsf{o} \Rightarrow \mathsf{o}) \Rightarrow \mathsf{o} \Rightarrow \mathsf{o}
\end{aligned}$$

13

Example

$$\begin{aligned}
[] &: \mathsf{o}\\
\mathtt{cons} &: \mathsf{o} \Rightarrow \mathsf{o} \Rightarrow \mathsf{o}\\
\mathtt{map} &: (\mathsf{o} \Rightarrow \mathsf{o}) \Rightarrow \mathsf{o} \Rightarrow \mathsf{o}
\end{aligned}$$

$$\begin{aligned}
\mathtt{map}(F,[]) &\to []\\
\mathtt{map}(F,\mathtt{cons}(x,y)) &\to \mathtt{cons}(F \cdot x, \mathtt{map}(F,y))
\end{aligned}$$

Choose $\mathtt{map} \rhd \mathtt{cons}, []$.