Encoding predicate subtyping in Dedukti

Gabriel Hondet

24th June 2022

Predicate subtyping

In Dedukti

Automation for subtyping

Predicate subtyping

is STT + a few things

$$t ::= x \mid t\,t \mid f \mid \lambda x : t, t \mid \Pi x : t, t$$

Predicate subtyping

is STT + a few things

$$t ::= x \mid t\,t \mid f \mid \lambda x : t, t \mid \Pi x : t, t \mid \{x : t \mid t\}$$

Predicate subtyping

is STT + a few things

$$t ::= x \mid t\,t \mid f \mid \lambda x : t, t \mid \Pi x : t, t \mid \{x : t \mid t\}$$

$$\frac{\Gamma \vdash t : T \qquad \vdash \{t/x\}\,P}{\Gamma \vdash t : \{x : T \mid P\}}$$

Predicate subtyping

is STT + a few things

$$t ::= x \mid t\,t \mid f \mid \lambda x : t, t \mid \Pi x : t, t \mid \{x : t \mid t\}$$

$$\frac{\Gamma \vdash t : T \qquad \vdash \{t/x\}\, P}{\Gamma \vdash t : \{x : T \mid P\}} \qquad \frac{\Gamma \vdash t : \{x : T \mid P\}}{\Gamma \vdash t : T}$$

Predicate subtyping

makes maths easy

$$\frac{\qquad}{\Gamma \vdash \exp\left(\frac{2i\pi}{3}\right) : \{z : \mathbf{C} \mid \exists n.\ z^n = 1\}}$$

Predicate subtyping

makes maths easy

$$\dfrac{\dfrac{}{\Gamma \vdash \exp\left(\frac{2i\pi}{3}\right) : \mathbf{C}}}{\Gamma \vdash \exp\left(\frac{2i\pi}{3}\right) : \{z : \mathbf{C} \mid \exists n.\ z^n = 1\}}$$

Predicate subtyping

makes maths easy

$$\dfrac{\dfrac{\Gamma \vdash \exp : \mathbf{C} \to \mathbf{C} \qquad \Gamma \vdash \frac{2i\pi}{3} : \mathbf{C}}{\Gamma \vdash \exp\left(\frac{2i\pi}{3}\right) : \mathbf{C}}}{\Gamma \vdash \exp\left(\frac{2i\pi}{3}\right) : \{z : \mathbf{C} \mid \exists n.\ z^n = 1\}}$$

Predicate subtyping

makes maths easy

$$\dfrac{\dfrac{\Gamma \vdash \exp : \mathbf{C} \to \mathbf{C} \qquad \Gamma \vdash \frac{2i\pi}{3} : \mathbf{C}}{\Gamma \vdash \exp\left(\frac{2i\pi}{3}\right) : \mathbf{C}} \qquad \Gamma \vdash \exists n.\ \left(\exp\left(\frac{2i\pi}{3}\right)\right)^n = 1}{\Gamma \vdash \exp\left(\frac{2i\pi}{3}\right) : \{z : \mathbf{C} \mid \exists n.\ z^n = 1\}}$$

Predicate subtyping

makes safe programming easy

$\mathtt{incrHd}([3;4]) = [4;4]$

$\mathtt{incrHd}(s) = \mathtt{push}(\mathtt{top}(s) + 1, \mathtt{tail}(s))$

Predicate subtyping

makes safe programming easy

$$\texttt{incrHd}([3;4]) = [4;4]$$

$$\texttt{incrHd}(s) = \texttt{push}(\texttt{top}(s) + 1, \texttt{tail}(s))$$

$$\{\neg\texttt{empty}(s)\}\ \texttt{incrHd}(s)$$

Predicate subtyping

makes safe programming easy

$$\texttt{incrHd}([3;4]) = [4;4] \qquad \texttt{incrHd}(s) = \texttt{push}(\texttt{top}(s)+1, \texttt{tail}(s))$$

$$\{\neg\texttt{empty}(s)\}\ \texttt{incrHd}(s)\ \{\lambda r, \neg\texttt{empty}(r) \wedge \texttt{top}(r) > \texttt{top}(s)\}$$

Predicate subtyping

makes safe programming easy

$\texttt{incrHd}([3;4]) = [4;4]$ $\texttt{incrHd}(s) = \texttt{push}(\texttt{top}(s) + 1, \texttt{tail}(s))$

$$\{\neg\texttt{empty}(s)\}\ \texttt{incrHd}(s)\ \{\lambda r, \neg\texttt{empty}(r) \wedge \texttt{top}(r) > \texttt{top}(s)\}$$

$\vdash \texttt{incrHd} :$

Predicate subtyping

makes safe programming easy

$$\texttt{incrHd}([3;4]) = [4;4] \qquad \texttt{incrHd}(s) = \texttt{push}(\texttt{top}(s) + 1, \texttt{tail}(s))$$

$$\{\neg\texttt{empty}(s)\}\ \texttt{incrHd}(s)\ \{\lambda r, \neg\texttt{empty}(r) \land \texttt{top}(r) > \texttt{top}(s)\}$$

$$\vdash \texttt{incrHd} : \Pi\,(s : \{s : \mathsf{stk} \mid \neg\texttt{empty}(s)\})\,,$$

Predicate subtyping

makes safe programming easy

$$\texttt{incrHd}([3;4]) = [4;4] \qquad \texttt{incrHd}(s) = \texttt{push}(\texttt{top}(s)+1, \texttt{tail}(s))$$

$$\{\neg\texttt{empty}(s)\}\ \texttt{incrHd}(s)\ \{\lambda r, \neg\texttt{empty}(r) \land \texttt{top}(r) > \texttt{top}(s)\}$$

$$\vdash \texttt{incrHd} : \Pi\,(s : \{s : \mathsf{stk} \mid \neg\texttt{empty}(s)\}), \{r : \mathsf{stk} \mid \neg\texttt{empty}(r) \land \texttt{top}(r) > \texttt{top}(s)\}$$

Predicate subtyping

is actually used!

In PVS

```
{ x : reals | x > 0 }
```

In F*

```
x : reals { x > 0 }
```

Predicate subtyping

In Dedukti

Automation for subtyping

Simple type theory

remember Gilles' talk?

$\texttt{El} : \texttt{Set} \to \mathtt{TYPE}; \quad \texttt{Prf} : (\texttt{El}\ \texttt{o}) \to \mathtt{TYPE}; \quad \mathbf{N} : \texttt{Set}$

$[\![-]\!] : \mathsf{STT} \to \mathsf{Lp}[\mathsf{STT}]$

Simple type theory

remember Gilles' talk?

$\mathtt{El} : \mathtt{Set} \longrightarrow \mathtt{TYPE}; \quad \mathtt{Prf} : (\mathtt{El}\,\mathtt{o}) \longrightarrow \mathtt{TYPE}; \quad \mathbf{N} : \mathtt{Set}$

$[\![-]\!] : \mathsf{STT} \longrightarrow \mathsf{Lp}[\mathsf{STT}]$

- $[\![\Gamma \vdash \lambda x : \mathbf{N}, x : \mathbf{N} \to \mathbf{N}]\!] = \Delta \vdash \lambda x : (\mathtt{El}\,\mathbf{N}), x : (\mathtt{El}\,(\mathbf{N} \longrightarrow \mathbf{N}))$

Simple type theory

remember Gilles' talk?

$$\texttt{El} : \texttt{Set} \longrightarrow \mathtt{TYPE}; \quad \texttt{Prf} : (\texttt{El}\ \texttt{o}) \longrightarrow \mathtt{TYPE}; \quad \mathbf{N} : \texttt{Set}$$

$$[-] : \mathsf{STT} \longrightarrow \mathsf{Lp}[\mathsf{STT}]$$

- $[\Gamma \vdash \lambda x : \mathbf{N}, x : \mathbf{N} \to \mathbf{N}] = \Delta \vdash \lambda x : (\texttt{El}\ \mathbf{N}), x : (\texttt{El}\ (\mathbf{N} \longrightarrow \mathbf{N}))$
- $[\Gamma \vdash \forall_{\mathbf{N}} x.P : o] = \Delta \vdash (\forall\ \mathbf{N}\ [P]) : (\texttt{El}\ \texttt{o})$

Simple type theory

remember Gilles' talk?

$$\texttt{El} : \texttt{Set} \to \mathtt{TYPE}; \quad \texttt{Prf} : (\texttt{El}\ \texttt{o}) \to \mathtt{TYPE}; \quad \mathbf{N} : \texttt{Set}$$

$$[-] : \mathsf{STT} \to \mathsf{Lp}[\mathsf{STT}]$$

- $[\Gamma \vdash \lambda x : \mathbf{N}, x : \mathbf{N} \to \mathbf{N}] = \Delta \vdash \lambda x : (\texttt{El}\ \mathbf{N}), x : (\texttt{El}\ (\mathbf{N} \to \mathbf{N}))$
- $[\Gamma \vdash \forall_{\mathbf{N}} x.P : o] = \Delta \vdash (\forall\ \mathbf{N}\ [P]) : (\texttt{El}\ \texttt{o})$
- $[\Gamma \vdash \lambda h, h : P \Rightarrow P] = \Delta \vdash \lambda h : (\texttt{Prf}\ [P]), h : (\texttt{Prf}\ ([P] \Rightarrow [P]))$

Predicate subtyping

Dedukti likes predicates, but not subtyping

$\texttt{psub}$:

$(\texttt{psub}$ $)$:

Predicate subtyping

Dedukti likes predicates, but not subtyping

$\texttt{psub} : \Pi\, T : \texttt{Set},$

$(\texttt{psub}\, \mathbf{N} \qquad\qquad\qquad) :$

Predicate subtyping

Dedukti likes predicates, but not subtyping

$\texttt{psub} : \Pi T : \texttt{Set}, (\texttt{El}\,(T \rightarrow \texttt{o})) \rightarrow$
$(\texttt{psub}\,\mathbf{N}\,(\lambda x, x > 0))$:

Predicate subtyping

Dedukti likes predicates, but not subtyping

$$\mathtt{psub} : \Pi T : \mathtt{Set}, (\mathtt{El}\,(T \rightarrow \mathtt{o})) \rightarrow \mathtt{Set}$$

$$(\mathtt{psub}\,\mathbf{N}\,(\lambda x, x > 0)) : \mathtt{Set}$$

Predicate subtyping

Dedukti likes predicates, but not subtyping

$$\texttt{psub} : \Pi T : \texttt{Set}, (\texttt{El}\,(T \rightarrow \texttt{o})) \rightarrow \texttt{Set}$$
$$(\texttt{psub}\,\mathbf{N}\,(\lambda x, x > 0)) \;:\; \texttt{Set}$$

$$\lambda x : \qquad\qquad , x : \texttt{El}\,(\qquad \rightarrow \qquad)$$

Predicate subtyping

Dedukti likes predicates, but not subtyping

$$\texttt{psub} : \Pi\, T : \texttt{Set}, (\texttt{El}\,(T \to \texttt{o})) \to \texttt{Set}$$

$$(\texttt{psub}\,\mathbf{N}\,(\lambda x, x > 0)) \ : \texttt{Set}$$

$$\lambda x : \ (\texttt{El}\ (\texttt{psub}\,\mathbf{N}\,\mathrm{posp}))\,, x : \texttt{El}\,(\,(\texttt{psub}\,\mathbf{N}\,\mathrm{posp}) \to (\texttt{psub}\,\mathbf{N}\,\mathrm{posp})\,)$$

Predicate subtyping

Dedukti likes predicates, but not subtyping

$\mathtt{psub} : \Pi T : \mathtt{Set}, (\mathtt{El}\,(T \to \mathtt{o})) \to \mathtt{Set}$

$(\mathtt{psub}\,\mathbf{N}\,(\lambda x, x > 0)) \; : \mathtt{Set}$

$$\lambda x : (\mathtt{El}\,(\mathtt{psub}\,\mathbf{N}\,\mathrm{posp})), x : \mathtt{El}\,((\mathtt{psub}\,\mathbf{N}\,\mathrm{posp}) \to (\mathtt{psub}\,\mathbf{N}\,\mathrm{posp}))$$

Unicity of types in Dedukti (modulo $\simeq$):

Predicate subtyping

Dedukti likes predicates, but not subtyping

$$\texttt{psub} : \Pi T : \texttt{Set}, (\texttt{El}\,(T \rightarrow \texttt{o})) \rightarrow \texttt{Set}$$
$$(\texttt{psub}\,\mathbf{N}\,(\lambda x, x > 0))\ : \texttt{Set}$$

$$\lambda x : (\texttt{El}\ (\texttt{psub}\,\mathbf{N}\,\texttt{posp}))\,, x : \texttt{El}\,(\,(\texttt{psub}\,\mathbf{N}\,\texttt{posp}) \rightarrow (\texttt{psub}\,\mathbf{N}\,\texttt{posp})\,)$$

Unicity of types in Dedukti (modulo $\simeq$):

- either $\vdash \exp\left(\frac{2i\pi}{3}\right) : \texttt{El}\ (\texttt{psub}\,\mathbf{C}\,(\exists n, \ldots))$

Predicate subtyping

Dedukti likes predicates, but not subtyping

$\texttt{psub} : \Pi T : \texttt{Set}, (\texttt{El}\,(T \to \texttt{o})) \to \texttt{Set}$
$(\texttt{psub}\,\mathbf{N}\,(\lambda x, x > 0)) : \texttt{Set}$

$$\lambda x : (\texttt{El}\,(\texttt{psub}\,\mathbf{N}\,\texttt{posp})), x : \texttt{El}\,((\texttt{psub}\,\mathbf{N}\,\texttt{posp}) \to (\texttt{psub}\,\mathbf{N}\,\texttt{posp}))$$

Unicity of types in Dedukti (modulo $\simeq$):

- either $\vdash \exp\left(\frac{2i\pi}{3}\right) : \texttt{El}\,(\texttt{psub}\,\mathbf{C}\,(\exists n, \ldots))$
- or $\vdash \exp\left(\frac{2i\pi}{3}\right) : \texttt{El}\,\mathbf{C}$

Predicate subtyping

Dedukti likes predicates, but not subtyping

$$\mathtt{psub} : \Pi T : \mathtt{Set}, (\mathtt{El}\,(T \to \mathtt{o})) \to \mathtt{Set}$$
$$(\mathtt{psub}\,\mathbf{N}\,(\lambda x, x > 0)) : \mathtt{Set}$$

$$\lambda x : (\mathtt{El}\ (\mathtt{psub}\,\mathbf{N}\,\mathrm{posp})), x : \mathtt{El}\,((\mathtt{psub}\,\mathbf{N}\,\mathrm{posp}) \to (\mathtt{psub}\,\mathbf{N}\,\mathrm{posp}))$$

Unicity of types in Dedukti (modulo $\simeq$):

- either $\vdash \exp\left(\frac{2i\pi}{3}\right) : \mathtt{El}\ (\mathtt{psub}\,\mathbf{C}\,(\exists n, \ldots))$
- or $\vdash \exp\left(\frac{2i\pi}{3}\right) : \mathtt{El}\,\mathbf{C}$

but not both

Predicate subtyping

Dedukti likes predicates, but not subtyping

$$\mathtt{psub} : \Pi T : \mathtt{Set}, (\mathtt{El}\,(T \rightarrow \mathtt{o})) \rightarrow \mathtt{Set}$$
$$(\mathtt{psub}\,\mathbf{N}\,(\lambda x, x > 0)) : \mathtt{Set}$$

$$\lambda x : (\mathtt{El}\,(\mathtt{psub}\,\mathbf{N}\,\mathrm{posp})), x : \mathtt{El}\,((\mathtt{psub}\,\mathbf{N}\,\mathrm{posp}) \rightarrow (\mathtt{psub}\,\mathbf{N}\,\mathrm{posp}))$$

Unicity of types in Dedukti (modulo $\simeq$):

- either $\vdash \exp\left(\frac{2i\pi}{3}\right) : \mathtt{El}\,(\mathtt{psub}\,\mathbf{C}\,(\exists n, \ldots))$
- or $\vdash \exp\left(\frac{2i\pi}{3}\right) : \mathtt{El}\,\mathbf{C}$

but not both

Interpretation of subtyping through coercions

Interpreting subtyping

Dedukti wants everything explained

$$\frac{\vdash e : (\texttt{El}\ (\texttt{psub}\ T\ P))}{\vdash (\qquad e) : (\texttt{El}\ T)}; \qquad \frac{\vdash e : (\texttt{El}\ T) \qquad \vdash \quad : (\texttt{Prf}\ (P\ e))}{\vdash (\qquad e \quad) : (\texttt{El}\ (\texttt{psub}\ T\ P))}$$

Interpreting subtyping

Dedukti wants everything explained

$$\frac{\vdash e : (\mathtt{El}\ (\mathtt{psub}\ T\ P))}{\vdash (\mathtt{fst}\ e) : (\mathtt{El}\ T)}; \qquad \frac{\vdash e : (\mathtt{El}\ T) \qquad \vdash \quad : (\mathtt{Prf}\ (P\ e))}{\vdash (\qquad e \quad) : (\mathtt{El}\ (\mathtt{psub}\ T\ P))}$$

Interpreting subtyping

Dedukti wants everything explained

$$\frac{\vdash e : (\mathtt{El}\ (\mathtt{psub}\ T\ P))}{\vdash (\mathtt{fst}\ e) : (\mathtt{El}\ T)};\qquad \frac{\vdash e : (\mathtt{El}\ T) \qquad \vdash h : (\mathtt{Prf}\ (P\ e))}{\vdash (\mathtt{pair}\ e\ h) : (\mathtt{El}\ (\mathtt{psub}\ T\ P))}$$

Interpreting subtyping

Dedukti wants everything explained

$$\frac{\vdash e : (\texttt{El}\ (\texttt{psub}\ T\ P))}{\vdash (\texttt{fst}\ e)\ :\ (\texttt{El}\ T)}; \qquad \frac{\vdash e : (\texttt{El}\ T) \qquad \vdash h : (\texttt{Prf}\ (P\ e))}{\vdash (\texttt{pair}\ e\ h)\ :\ (\texttt{El}\ (\texttt{psub}\ T\ P))}$$

- $\texttt{pair} : \Pi t : \texttt{Set}, \Pi p : (\texttt{El}\ (t \rightarrow \texttt{o})),$

Interpreting subtyping

Dedukti wants everything explained

$$\frac{\vdash e : (\mathtt{El}\ (\mathtt{psub}\ T\ P))}{\vdash (\mathtt{fst}\ e) : (\mathtt{El}\ T)}; \qquad \frac{\vdash e : (\mathtt{El}\ T) \qquad \vdash h : (\mathtt{Prf}\ (P\ e))}{\vdash (\mathtt{pair}\ e\ h) : (\mathtt{El}\ (\mathtt{psub}\ T\ P))}$$

- $\mathtt{pair} : \Pi t : \mathtt{Set}, \Pi p : (\mathtt{El}\ (t \rightarrow \mathtt{o})), \Pi m : (\mathtt{El}\ t),$

Interpreting subtyping

Dedukti wants everything explained

$$\frac{\vdash e : (\mathtt{El}\ (\mathtt{psub}\ T\ P))}{\vdash (\mathtt{fst}\ e)\ :\ (\mathtt{El}\ T)};\qquad \frac{\vdash e : (\mathtt{El}\ T) \qquad \vdash h : (\mathtt{Prf}\ (P\ e))}{\vdash (\mathtt{pair}\ e\ h)\ :\ (\mathtt{El}\ (\mathtt{psub}\ T\ P))}$$

- $\mathtt{pair} : \Pi t : \mathtt{Set}, \Pi p : (\mathtt{El}\ (t \to \mathtt{o})), \Pi m : (\mathtt{El}\ t), (\mathtt{Prf}\ (p\ m)) \to$

Interpreting subtyping

Dedukti wants everything explained

$$\frac{\vdash e : (\mathtt{El}\ (\mathtt{psub}\ T\ P))}{\vdash (\mathtt{fst}\ e)\ :\ (\mathtt{El}\ T)};\qquad \frac{\vdash e : (\mathtt{El}\ T) \qquad \vdash h : (\mathtt{Prf}\ (P\ e))}{\vdash (\mathtt{pair}\ e\ h)\ :\ (\mathtt{El}\ (\mathtt{psub}\ T\ P))}$$

- $\mathtt{pair} : \Pi t : \mathtt{Set}, \Pi p : (\mathtt{El}\ (t \rightarrow \mathtt{o})), \Pi m : (\mathtt{El}\ t), (\mathtt{Prf}\ (p\ m)) \rightarrow (\mathtt{El}\ (\mathtt{psub}\ t\ p))$

Interpreting subtyping

Dedukti wants everything explained

$$\frac{\vdash e : (\mathtt{El}\ (\mathtt{psub}\ T\ P))}{\vdash (\mathtt{fst}\ e)\ :\ (\mathtt{El}\ T)}; \qquad \frac{\vdash e : (\mathtt{El}\ T) \qquad \vdash h : (\mathtt{Prf}\ (P\ e))}{\vdash (\mathtt{pair}\ e\ h)\ :\ (\mathtt{El}\ (\mathtt{psub}\ T\ P))}$$

- $\mathtt{pair} : \Pi t : \mathtt{Set}, \Pi p : (\mathtt{El}\ (t \rightarrow \mathtt{o})), \Pi m : (\mathtt{El}\ t), (\mathtt{Prf}\ (p\ m)) \rightarrow (\mathtt{El}\ (\mathtt{psub}\ t\ p))$
- $\mathtt{fst} : \Pi t : \mathtt{Set}, \Pi p : (\mathtt{El}\ (t \rightarrow \mathtt{o})),$

Interpreting subtyping

Dedukti wants everything explained

$$\frac{\vdash e : (\mathtt{El}\ (\mathtt{psub}\ T\ P))}{\vdash (\mathtt{fst}\ e)\ :\ (\mathtt{El}\ T)}; \qquad \frac{\vdash e : (\mathtt{El}\ T) \qquad \vdash h : (\mathtt{Prf}\ (P\ e))}{\vdash (\mathtt{pair}\ e\ h)\ :\ (\mathtt{El}\ (\mathtt{psub}\ T\ P))}$$

- $\mathtt{pair} : \Pi t : \mathtt{Set}, \Pi p : (\mathtt{El}\ (t \rightarrow \mathtt{o})), \Pi m : (\mathtt{El}\ t), (\mathtt{Prf}\ (p\ m)) \rightarrow (\mathtt{El}\ (\mathtt{psub}\ t\ p))$
- $\mathtt{fst} : \Pi t : \mathtt{Set}, \Pi p : (\mathtt{El}\ (t \rightarrow \mathtt{o})), (\mathtt{El}\ (\mathtt{psub}\ t\ p)) \rightarrow$

Interpreting subtyping

Dedukti wants everything explained

$$\frac{\vdash e : (\mathtt{El}\ (\mathtt{psub}\ T\ P))}{\vdash (\mathtt{fst}\ e) : (\mathtt{El}\ T)};\qquad \frac{\vdash e : (\mathtt{El}\ T) \qquad \vdash h : (\mathtt{Prf}\ (P\ e))}{\vdash (\mathtt{pair}\ e\ h) : (\mathtt{El}\ (\mathtt{psub}\ T\ P))}$$

- $\mathtt{pair} : \Pi t : \mathtt{Set}, \Pi p : (\mathtt{El}\ (t \rightarrow \mathtt{o})), \Pi m : (\mathtt{El}\ t), (\mathtt{Prf}\ (p\ m)) \rightarrow (\mathtt{El}\ (\mathtt{psub}\ t\ p))$
- $\mathtt{fst} : \Pi t : \mathtt{Set}, \Pi p : (\mathtt{El}\ (t \rightarrow \mathtt{o})), (\mathtt{El}\ (\mathtt{psub}\ t\ p)) \rightarrow (\mathtt{El}\ t)$

Proofs don't matter

Assume

$$\vdash h_1 : (\texttt{Prf}\ (\texttt{posp}\ 2))\,; \qquad \vdash h_2 : (\texttt{Prf}\ (\texttt{posp}\ 2))\,;$$

Proofs don't matter

Assume

$$\vdash h_1 : (\texttt{Prf}\ (\texttt{posp}\,2))\,; \qquad \vdash h_2 : (\texttt{Prf}\ (\texttt{posp}\,2))\,; \qquad h_1 \not\equiv h_2$$

Proofs don't matter

Assume

$$\vdash h_1 : (\texttt{Prf}\ (\mathrm{posp}\ 2))\ ; \qquad \vdash h_2 : (\texttt{Prf}\ (\mathrm{posp}\ 2))\ ; \qquad h_1 \not\equiv h_2$$

$$(\texttt{pair}\ \mathbf{N}\ \mathrm{posp}\ 2\ h_1) \qquad (\texttt{pair}\ \mathbf{N}\ \mathrm{posp}\ 2\ h_2) \qquad : \texttt{El}\ (\texttt{psub}\ \mathbf{N}\ \mathrm{posp})$$

Proofs don't matter

Assume

$$\vdash h_1 : (\texttt{Prf}\ (\mathrm{posp}\, 2))\,; \qquad \vdash h_2 : (\texttt{Prf}\ (\mathrm{posp}\, 2))\,; \qquad h_1 \not\equiv h_2$$

$$(\texttt{pair}\ \mathbf{N}\ \mathrm{posp}\, 2\, h_1) \quad \simeq \quad (\texttt{pair}\ \mathbf{N}\ \mathrm{posp}\, 2\, h_2) \quad : \texttt{El}\ (\texttt{psub}\ \mathbf{N}\ \mathrm{posp})$$

Both reduce to:

$$(\texttt{pair}^{\dagger}\ \mathbf{N}\ \mathrm{posp}\, 2)$$

Proofs don't matter

Assume

$$\vdash h_1 : (\texttt{Prf}\ (\mathrm{posp}\ 2))\ ; \qquad \vdash h_2 : (\texttt{Prf}\ (\mathrm{posp}\ 2))\ ; \qquad h_1 \not\equiv h_2$$

$$\left(\texttt{pair}\ \mathbf{N}\ \mathrm{posp}\ 2\ h_1\right) \quad \simeq \quad \left(\texttt{pair}\ \mathbf{N}\ \mathrm{posp}\ 2\ h_2\right) \quad : \texttt{El}\ \left(\texttt{psub}\ \mathbf{N}\ \mathrm{posp}\right)$$

$$\left(\texttt{pair}^{\dagger}\ \mathbf{N}\ \mathrm{posp}\ 2\right)$$

Proof Irrelevance

Proofs don't matter

or do they?

What about $\left(\texttt{pair}^{\dagger}\ \textbf{N}\ \texttt{posp}\ 0\right)$?

Proofs don't matter

or do they?

What about $\left(\texttt{pair}^{\dagger}\,\mathbf{N}\,\texttt{posp}\,0\right)$?

Symbol protection

Proofs don't matter

or do they?

What about $(\texttt{pair}^{\dagger}\,\mathbf{N}\,\texttt{posp}\,0)$?

Symbol protection

- $\texttt{pair}^{\dagger}$ not typable in foreign modules

Proofs don't matter

or do they?

What about $\left(\texttt{pair}^{\dagger}\,\mathbf{N}\,\texttt{posp}\,0\right)$?

Symbol protection

- $\texttt{pair}^{\dagger}$ not typable in foreign modules
- unless in rewrite rule left-hand side

Safety nets for Dedukti users

psub.lp

```
protected constant symbol pair† (T : Set) (P : (El (T → o))) : (El (psub P))
```

Safety nets for Dedukti users

psub.lp

```
protected constant symbol pair† (T : Set) (P : (El (T → o))) : (El (psub P))
```

```
symbol fill : El (stk → nestk) ≔ λs, (pair† s);
```

Safety nets for Dedukti users

psub.lp

protected constant symbol $\texttt{pair}^{\dagger}\,(T : \texttt{Set})\,(P : (\texttt{El}\,(T \rightarrow \texttt{o}))) : (\texttt{El}\,(\texttt{psub}\,P))$

symbol $\texttt{fill} : \texttt{El}\,(\texttt{stk} \rightarrow \texttt{nestk}) \coloneqq \lambda s, \left(\texttt{pair}^{\dagger}\, s\right);$

rule $(\texttt{fill}\, s) \hookrightarrow \left(\texttt{pair}^{\dagger}\, s\right);$

Safety nets for Dedukti users

psub.lp

protected constant symbol $\mathtt{pair}^{\dagger}\,(T : \mathtt{Set})\,(P : (\mathtt{El}\,(T \to \mathtt{o}))) : (\mathtt{El}\,(\mathtt{psub}\,P))$

symbol $\mathtt{fill} : \mathtt{El}\,(\mathtt{stk} \to \mathtt{nestk}) \coloneqq \lambda s, (\mathtt{pair}^{\dagger}\, s)$;

rule $(\mathtt{fill}\, s) \hookrightarrow (\mathtt{pair}^{\dagger}\, s)$;

rule $(\mathtt{concat}\, s\, (\mathtt{pair}^{\dagger}\, s)) \hookrightarrow (\mathtt{push}\, \ldots)$;

Let's try it out!

```
constant symbol stk : Set;
```

Let's try it out!

```
constant symbol stk : Set;
constant symbol empty : El stk;
```

Let's try it out!

```
constant symbol stk : Set;
constant symbol empty : El stk;
constant symbol nestkp (s : El stk) ≔ s ≠ empty;
```

Let's try it out!

constant symbol $\mathtt{stk} : \mathtt{Set}$;
constant symbol $\mathtt{empty} : \mathtt{El}\ \mathtt{stk}$;
constant symbol $\mathtt{nestkp}\,(s : \mathtt{El}\ \mathtt{stk}) \coloneqq s \neq \mathtt{empty}$;
symbol $\mathtt{nestk} \coloneqq \mathtt{psub}\ \mathtt{stk}\ \mathtt{nestkp}$;

Let's try it out!

constant symbol $\texttt{stk} : \texttt{Set}$;
constant symbol $\texttt{empty} : \texttt{El}\,\texttt{stk}$;
constant symbol $\texttt{nestkp}\,(s : \texttt{El}\,\texttt{stk}) \coloneqq s \neq \texttt{empty}$;
symbol $\texttt{nestk} \coloneqq \texttt{psub}\,\texttt{stk}\,\texttt{nestkp}$;
constant symbol $\texttt{push} : (\texttt{El}\,(\mathbf{N} \longrightarrow \texttt{stk} \longrightarrow \texttt{nestk}))$;

Let's try it out!

constant symbol $\texttt{stk} : \texttt{Set}$;
constant symbol $\texttt{empty} : \texttt{El}\ \texttt{stk}$;
constant symbol $\texttt{nestkp}\,(s : \texttt{El}\ \texttt{stk}) \coloneqq s \neq \texttt{empty}$;
symbol $\texttt{nestk} \coloneqq \texttt{psub}\ \texttt{stk}\ \texttt{nestkp}$;
constant symbol $\texttt{push} : (\texttt{El}\,(\mathbf{N} \rightarrow \texttt{stk} \rightarrow \texttt{nestk}))$;
constant symbol $\texttt{pop} : (\texttt{El}\,(\texttt{nestk} \rightarrow \texttt{stk}))$;

Let's try it out!

constant symbol $\mathtt{stk} : \mathtt{Set}$;
constant symbol $\mathtt{empty} : \mathtt{El}\,\mathtt{stk}$;
constant symbol $\mathtt{nestkp}\,(s : \mathtt{El}\,\mathtt{stk}) \coloneqq s \neq \mathtt{empty}$;
symbol $\mathtt{nestk} \coloneqq \mathtt{psub}\,\mathtt{stk}\,\mathtt{nestkp}$;
constant symbol $\mathtt{push} : (\mathtt{El}\,(\mathbf{N} \to \mathtt{stk} \to \mathtt{nestk}))$;
constant symbol $\mathtt{pop} : (\mathtt{El}\,(\mathtt{nestk} \to \mathtt{stk}))$;
constant symbol $\mathtt{top} : (\mathtt{El}\,(\mathtt{nestk} \to \mathbf{N}))$;

Let's try it out!

```
constant symbol stk : Set;
constant symbol empty : El stk;
constant symbol nestkp (s : El stk) ≔ s ≠ empty;
symbol nestk ≔ psub stk nestkp;
constant symbol push : (El (ℕ → stk → nestk)) ;
constant symbol pop : (El (nestk → stk)) ;
constant symbol top : (El (nestk → ℕ)) ;
constant symbol pushTopPop : Prf
```

Let's try it out!

```
constant symbol stk : Set;
constant symbol empty : El stk;
constant symbol nestkp (s : El stk) ≔ s ≠ empty;
symbol nestk ≔ psub stk nestkp;
constant symbol push : (El (ℕ → stk → nestk)) ;
constant symbol pop : (El (nestk → stk)) ;
constant symbol top : (El (nestk → ℕ)) ;
constant symbol pushTopPop : Prf
  ∀ stk (λs,
```

Let's try it out!

```
constant symbol stk : Set;
constant symbol empty : El stk;
constant symbol nestkp (s : El stk) ≔ s ≠ empty;
symbol nestk ≔ psub stk nestkp;
constant symbol push : (El (ℕ ⟶ stk ⟶ nestk)) ;
constant symbol pop : (El (nestk ⟶ stk)) ;
constant symbol top : (El (nestk ⟶ ℕ)) ;
constant symbol pushTopPop : Prf
  ∀ stk (λ s,
  (nestkp s) ⇒
```

Let's try it out!

constant symbol $\texttt{stk} : \texttt{Set}$;
constant symbol $\texttt{empty} : \texttt{El}\,\texttt{stk}$;
constant symbol $\texttt{nestkp}\,(s : \texttt{El}\,\texttt{stk}) \coloneqq s \neq \texttt{empty}$;
symbol $\texttt{nestk} \coloneqq \texttt{psub}\,\texttt{stk}\,\texttt{nestkp}$;
constant symbol $\texttt{push} : (\texttt{El}\,(\mathbf{N} \to \texttt{stk} \to \texttt{nestk}))$;
constant symbol $\texttt{pop} : (\texttt{El}\,(\texttt{nestk} \to \texttt{stk}))$;
constant symbol $\texttt{top} : (\texttt{El}\,(\texttt{nestk} \to \mathbf{N}))$;
constant symbol $\texttt{pushTopPop} : \texttt{Prf}$
$\forall\,\texttt{stk}\,(\lambda s,$
$(\texttt{nestkp}\,s) \Rightarrow$
$(\quad (\texttt{push}\,(\texttt{top}\,(\quad s \quad))\,(\texttt{pop}\,(\quad s \quad)))) = s)$

Let's try it out!

```
constant symbol stk : Set;
constant symbol empty : El stk;
constant symbol nestkp (s : El stk) ≔ s ≠ empty;
symbol nestk ≔ psub stk nestkp;
constant symbol push : (El (ℕ → stk → nestk)) ;
constant symbol pop : (El (nestk → stk)) ;
constant symbol top : (El (nestk → ℕ)) ;
constant symbol pushTopPop : Prf
  ∀ stk (λ s,
  (nestkp s) ⇒
  (      (push (top (pair s ?)) (pop (pair s ?)))) = s)
```

Let's try it out!

```
constant symbol stk : Set;
constant symbol empty : El stk;
constant symbol nestkp (s : El stk) ≔ s ≠ empty;
symbol nestk ≔ psub stk nestkp;
constant symbol push : (El (ℕ → stk → nestk)) ;
constant symbol pop : (El (nestk → stk)) ;
constant symbol top : (El (nestk → ℕ)) ;
constant symbol pushTopPop : Prf
  ∀ stk (λ s,
  (nestkp s) ⇒
  (fst (push (top (pair s ?)) (pop (pair s ?)))) = s)
```

Let's debug this out

$$(\forall\, \texttt{stk}\, (\lambda s, (\texttt{nestkp}\, s) \Rightarrow (\texttt{fst}\ (\texttt{push}\ (\texttt{top}\ (\texttt{pair}\, s\, ?t))\ (\texttt{pop}\ (\texttt{pair}\, s\, ?p)))) = s))$$

Let's debug this out

$$(\forall\, \mathtt{stk}\, (\lambda s, (\mathtt{nestkp}\, s) \Rightarrow (\mathtt{fst}\ (\mathtt{push}\ (\mathtt{top}\ (\mathtt{pair}\, s\, ?t))\ (\mathtt{pop}\ (\mathtt{pair}\, s\, ?p)))) = s))$$

- $\mathtt{top} : \mathtt{El}\, ((\mathtt{psub}\, \mathtt{stk}\, (\lambda x, x \neq \mathtt{empty})) \rightarrow \mathbf{N})$
- $s : (\mathtt{El}\, \mathtt{stk}) \vdash ?t : (\mathtt{Prf}\, (s \neq \mathtt{empty}))$

Let's debug this out

$$(\forall\, \mathtt{stk}\, (\lambda s, (\mathtt{nestkp}\, s) \Rightarrow (\mathtt{fst}\ (\mathtt{push}\ (\mathtt{top}\ (\mathtt{pair}\, s\, ?t))\ (\mathtt{pop}\ (\mathtt{pair}\, s\, ?p)))) = s))$$

- $\mathtt{top} : \mathtt{El}\, ((\mathtt{psub}\, \mathtt{stk}\, (\lambda x, x \neq \mathtt{empty})) \rightarrow \mathbf{N})$
- $s : (\mathtt{El}\, \mathtt{stk}) \vdash ?t : (\mathtt{Prf}\, (s \neq \mathtt{empty}))$

- $\mathtt{pop} : \mathtt{El}\, ((\mathtt{psub}\, \mathtt{stk}\, (\lambda x, x \neq \mathtt{empty})) \rightarrow \mathtt{stk})$
- $s : (\mathtt{El}\, \mathtt{stk}) \vdash ?p : (\mathtt{Prf}\, (s \neq \mathtt{empty}))$

Let's debug this out

$$(\forall\, \mathtt{stk}\, (\lambda s, (\mathtt{nestkp}\, s) \Rightarrow (\mathtt{fst}\ (\mathtt{push}\ (\mathtt{top}\ (\mathtt{pair}\, s\, ?t))\ (\mathtt{pop}\ (\mathtt{pair}\, s\, ?p)))) = s))$$

- $\mathtt{top} : \mathtt{El}\,((\mathtt{psub}\,\mathtt{stk}\,(\lambda x, x \neq \mathtt{empty})) \rightarrow \mathbf{N})$
- $s : (\mathtt{El}\,\mathtt{stk}) \vdash ?t : (\mathtt{Prf}\,(s \neq \mathtt{empty}))$

- $\mathtt{pop} : \mathtt{El}\,((\mathtt{psub}\,\mathtt{stk}\,(\lambda x, x \neq \mathtt{empty})) \rightarrow \mathtt{stk})$
- $s : (\mathtt{El}\,\mathtt{stk}) \vdash ?p : (\mathtt{Prf}\,(s \neq \mathtt{empty}))$

$$\frac{s : (\mathtt{El}\,\mathtt{stk})}{?x : (\mathtt{Prf}\,(s \neq \mathtt{empty}))}$$

Predicate subtypes inhabitants contain proofs

Let's recap

$$s : (\mathtt{El}\,\mathtt{stk}) \vdash (\mathtt{nestkp}\,s) \Rightarrow (\mathtt{fst}\;(\mathtt{push}\;(\mathtt{top}\;(\mathtt{pair}\,s\,?t))\;(\mathtt{pop}\;(\mathtt{pair}\,s\,?p)))) = s$$

with $\Rightarrow : \mathtt{El}\,(\mathtt{o} \to \mathtt{o} \to \mathtt{o})$

Predicate subtypes inhabitants contain proofs

Let's recap

$$s : (\texttt{El}\,\texttt{stk}) \vdash (\texttt{nestkp}\,s) \Rightarrow (\texttt{fst}\ (\texttt{push}\ (\texttt{top}\ (\texttt{pair}\,s\,?t))\ (\texttt{pop}\ (\texttt{pair}\,s\,?p)))) = s$$

with $\Rightarrow : \texttt{El}\,(\texttt{o} \to \texttt{o} \to \texttt{o})$

Dependent implication $\Rightarrow : \Pi p : (\texttt{El}\,\texttt{o}), ((\texttt{Prf}\ p) \to (\texttt{El}\,\texttt{o})) \to (\texttt{El}\,\texttt{o})$

Predicate subtypes inhabitants contain proofs

Let's recap

$$s : (\texttt{El}\ \texttt{stk}) \vdash (\texttt{nestkp}\ s) \Rightarrow (\texttt{fst}\ (\texttt{push}\ (\texttt{top}\ (\texttt{pair}\ s\ ?t))\ (\texttt{pop}\ (\texttt{pair}\ s\ ?p)))) = s$$

with $\Rightarrow : \texttt{El}\,(\texttt{o} \to \texttt{o} \to \texttt{o})$

Dependent implication $\Rightarrow : \Pi p : (\texttt{El}\ \texttt{o}), ((\texttt{Prf}\ p) \to (\texttt{El}\ \texttt{o})) \to (\texttt{El}\ \texttt{o})$

$$(\texttt{nestkp}\ s) \Rightarrow \lambda h : \texttt{Prf}\ (\texttt{nestkp}\ s),\\ (\texttt{fst}\ (\texttt{push}\ (\texttt{top}\ (\texttt{pair}\ s\ ?t))\ (\texttt{pop}\ (\texttt{pair}\ s\ ?p)))) = s$$

Predicate subtypes inhabitants contain proofs

Let's recap

$$s : (\mathtt{El}\ \mathrm{stk}) \vdash (\mathrm{nestkp}\ s) \Rightarrow (\mathtt{fst}\ (\mathrm{push}\ (\mathrm{top}\ (\mathtt{pair}\ s\ ?t))\ (\mathrm{pop}\ (\mathtt{pair}\ s\ ?p)))) = s$$

with $\Rightarrow : \mathtt{El}\ (\mathtt{o} \to \mathtt{o} \to \mathtt{o})$

Dependent implication $\Rightarrow : \Pi p : (\mathtt{El}\ \mathtt{o}), ((\mathtt{Prf}\ p) \to (\mathtt{El}\ \mathtt{o})) \to (\mathtt{El}\ \mathtt{o})$

$$(\mathrm{nestkp}\ s) \Rightarrow \lambda h : \mathtt{Prf}\ (\mathrm{nestkp}\ s), \\ (\mathtt{fst}\ (\mathrm{push}\ (\mathrm{top}\ (\mathtt{pair}\ s\ ?t))\ (\mathrm{pop}\ (\mathtt{pair}\ s\ ?p)))) = s$$

$$\frac{s : (\mathtt{El}\ \mathrm{stk}) \qquad h : (\mathtt{Prf}\ (s \neq \mathrm{empty}))}{?x \qquad : (\mathtt{Prf}\ (s \neq \mathrm{empty}))}$$

Predicate subtypes inhabitants contain proofs

Let's recap

$$s : (\mathtt{El}\ \mathtt{stk}) \vdash (\mathtt{nestkp}\ s) \Rightarrow (\mathtt{fst}\ (\mathtt{push}\ (\mathtt{top}\ (\mathtt{pair}\ s\ ?t))\ (\mathtt{pop}\ (\mathtt{pair}\ s\ ?p)))) = s$$

with $\Rightarrow : \mathtt{El}\ (\mathtt{o} \to \mathtt{o} \to \mathtt{o})$

Dependent implication $\Rightarrow : \Pi p : (\mathtt{El}\ \mathtt{o}), ((\mathtt{Prf}\ p) \to (\mathtt{El}\ \mathtt{o})) \to (\mathtt{El}\ \mathtt{o})$

$$(\mathtt{nestkp}\ s) \Rightarrow \lambda h : \mathtt{Prf}\ (\mathtt{nestkp}\ s),$$
$$(\mathtt{fst}\ (\mathtt{push}\ (\mathtt{top}\ (\mathtt{pair}\ s\ ?t))\ (\mathtt{pop}\ (\mathtt{pair}\ s\ ?p)))) = s$$

$$\frac{s : (\mathtt{El}\ \mathtt{stk}) \qquad h : (\mathtt{Prf}\ (s \neq \mathtt{empty}))}{?x := h : (\mathtt{Prf}\ (s \neq \mathtt{empty}))}$$

Predicate subtyping

In Dedukti

Automation for subtyping

Dedukti is too rigid

$$\dfrac{\dfrac{}{}}{s : \texttt{El}\,\texttt{stk} \vdash (\texttt{top}\,s) : \texttt{El}\,\mathbf{N}}$$

Dedukti is too rigid

$$\frac{s : \mathtt{El}\,\mathtt{stk} \vdash \mathtt{top} : \mathtt{El}\,\mathtt{nestk} \to \mathtt{El}\,\mathbf{N} \qquad \overline{}}{s : \mathtt{El}\,\mathtt{stk} \vdash (\mathtt{top}\,s) : \mathtt{El}\,\mathbf{N}}$$

Dedukti is too rigid

$$\dfrac{s : \mathtt{El}\,\mathtt{stk} \vdash \mathtt{top} : \mathtt{El}\,\mathtt{nestk} \to \mathtt{El}\,\mathbf{N} \qquad \dfrac{}{s : \mathtt{El}\,\mathtt{stk} \vdash s : \mathtt{El}\,\mathtt{nestk}}}{s : \mathtt{El}\,\mathtt{stk} \vdash (\mathtt{top}\,s) : \mathtt{El}\,\mathbf{N}}$$

Dedukti is too rigid

$$\dfrac{s : \mathtt{El}\,\mathtt{stk} \vdash \mathtt{top} : \mathtt{El}\,\mathtt{nestk} \to \mathtt{El}\,\mathbf{N} \qquad \dfrac{(\mathtt{El}\,\mathtt{stk}) \simeq (\mathtt{El}\,\mathtt{nestk})}{s : \mathtt{El}\,\mathtt{stk} \vdash s : \mathtt{El}\,\mathtt{nestk}}}{s : \mathtt{El}\,\mathtt{stk} \vdash (\mathtt{top}\,s) : \mathtt{El}\,\mathbf{N}}$$

Dedukti is too rigid

$$\dfrac{s : \mathtt{El}\,\mathtt{stk} \vdash \mathtt{top} : \mathtt{El}\,\mathtt{nestk} \to \mathtt{El}\,\mathbf{N} \qquad \dfrac{(\mathtt{El}\,\mathtt{stk}) \simeq (\mathtt{El}\,\mathtt{nestk})}{s : \mathtt{El}\,\mathtt{stk} \vdash s : \mathtt{El}\,\mathtt{nestk}}}{s : \mathtt{El}\,\mathtt{stk} \vdash (\mathtt{top}\,s) : \mathtt{El}\,\mathbf{N}}$$

Dedukti is too rigid

$$\frac{s : \mathtt{El}\,\mathtt{stk} \vdash \mathtt{top} : \mathtt{El}\,\mathtt{nestk} \to \mathtt{El}\,\mathbf{N} \qquad \dfrac{(\mathtt{El}\,\mathtt{stk}) \simeq (\mathtt{El}\,\mathtt{nestk})}{s : \mathtt{El}\,\mathtt{stk} \vdash s : \mathtt{El}\,\mathtt{nestk}}}{s : \mathtt{El}\,\mathtt{stk} \vdash (\mathtt{top}\,s) : \mathtt{El}\,\mathbf{N}}$$

$$\frac{\vdash t : T \qquad \vdash U : s \qquad T \simeq U}{\vdash t : U}$$

Dedukti is too rigid

$$\dfrac{s : \mathtt{El}\,\mathtt{stk} \vdash \mathtt{top} : \mathtt{El}\,\mathtt{nestk} \to \mathtt{El}\,\mathbf{N} \qquad \dfrac{(\mathtt{El}\,\mathtt{stk}) \simeq (\mathtt{El}\,\mathtt{nestk})}{s : \mathtt{El}\,\mathtt{stk} \vdash s : \mathtt{El}\,\mathtt{nestk}}}{s : \mathtt{El}\,\mathtt{stk} \vdash (\mathtt{top}\, s) : \mathtt{El}\,\mathbf{N}}$$

$$\dfrac{\vdash t : T \qquad \vdash U : s \qquad T \simeq U}{\vdash t : U}$$

$$\dfrac{\vdash t : T \qquad \vdash U : s \qquad T <: U}{\vdash t : U}$$

Dedukti is too rigid

$$\dfrac{s : \mathtt{El}\,\mathtt{stk} \vdash \mathtt{top} : \mathtt{El}\,\mathtt{nestk} \to \mathtt{El}\,\mathbf{N} \qquad \dfrac{(\mathtt{El}\,\mathtt{stk}) \simeq (\mathtt{El}\,\mathtt{nestk})}{s : \mathtt{El}\,\mathtt{stk} \vdash s : \mathtt{El}\,\mathtt{nestk}}}{s : \mathtt{El}\,\mathtt{stk} \vdash (\mathtt{top}\,s) : \mathtt{El}\,\mathbf{N}}$$

$$\dfrac{\vdash t : T \qquad \vdash U : s \qquad T \simeq U}{\vdash t : U}$$

$$\dfrac{\vdash t : T \qquad \vdash U : s \qquad T <: U}{\vdash t : U}$$

$(\mathtt{El}\,\mathtt{stk}) <: (\mathtt{El}\ (\mathtt{psub}\,\mathtt{stk}\,(\lambda s, s \neq \mathtt{empty})))?$

Lambdapi can be softened

with 'implicit' coercions

$$\Gamma \vdash t : T \rhd t_0$$

Lambdapi can be softened

with 'implicit' coercions

$$\Gamma \vdash t : T \rhd t_0$$

$$\frac{}{\Gamma \vdash t : U \rhd}$$

Lambdapi can be softened

with ‘implicit’ coercions

$$\Gamma \vdash t : T \rhd t_0$$

$$\frac{\Gamma \vdash t : T \rhd t_0 \qquad \Gamma \vdash U : s \rhd U_0}{\Gamma \vdash t : U \rhd}$$

Lambdapi can be softened

with 'implicit' coercions

$$\Gamma \vdash t : T \rhd t_0$$

$$\frac{\Gamma \vdash t : T \rhd t_0 \qquad \Gamma \vdash U : s \rhd U_0 \qquad \mathcal{D} :: U_0 <: T}{\Gamma \vdash t : U \rhd}$$

Lambdapi can be softened

with ‘implicit’ coercions

$$\Gamma \vdash t : T \rhd t_0$$

$$\frac{\Gamma \vdash t : T \rhd t_0 \qquad \Gamma \vdash U : s \rhd U_0 \qquad \mathcal{D} :: U_0 <: T}{\Gamma \vdash t : U \rhd (\llbracket \mathcal{D} \rrbracket \; t_0)}$$

Lambdapi can be softened

with 'implicit' coercions

$$\Gamma \vdash t : T \rhd t_0$$

$$\frac{\Gamma \vdash t : T \rhd t_0 \qquad \Gamma \vdash U : s \rhd U_0 \qquad \mathcal{D} :: U_0 <: T}{\Gamma \vdash t : U \rhd (\llbracket \mathcal{D} \rrbracket\ t_0)}$$

- $\llbracket T <: (\texttt{psub}\ T\ P) \rrbracket = \lambda x : \texttt{El}\ T, (\texttt{pair}\ T\ P\ x\ ?x)$

Lambdapi can be softened

with ‘implicit’ coercions

$$\Gamma \vdash t : T \rhd t_0$$

$$\frac{\Gamma \vdash t : T \rhd t_0 \qquad \Gamma \vdash U : s \rhd U_0 \qquad \mathcal{D} :: U_0 <: T}{\Gamma \vdash t : U \rhd (\llbracket \mathcal{D} \rrbracket\ t_0)}$$

- $\llbracket T <: (\texttt{psub}\ T\ P) \rrbracket = \lambda x : \texttt{El}\ T, (\texttt{pair}\ T\ P\ x\ ?x)$
- $\llbracket (\texttt{psub}\ T\ P) <: T \rrbracket = \lambda x : \texttt{El}\ (\texttt{psub}\ T\ P), (\texttt{fst}\ T\ P\ x)$

How to compute the subtyping derivation?

with rewrite rules, of course!

$([\![\mathcal{D} :: U <: T]\!]\ x)$ replaced by

How to compute the subtyping derivation?

with rewrite rules, of course!

$(\llbracket \mathcal{D} :: U <: T \rrbracket \; x)$ replaced by $(\kappa \; U \; T \; x)$

How to compute the subtyping derivation?

with rewrite rules, of course!

$([\![\mathcal{D} :: U <: T]\!]\ x)$ replaced by $(\kappa\ U\ T\ x)$

- $(\kappa\ (\texttt{psub}\ T\ P)\ T\ X) \hookrightarrow (\texttt{fst}\ T\ P\ X)$

How to compute the subtyping derivation?

with rewrite rules, of course!

$([\![\mathcal{D} :: U <: T]\!]\ x)$ replaced by $(\kappa\ U\ T\ x)$

- $(\kappa\ (\texttt{psub}\ T\ P)\ T\ X) \hookrightarrow (\texttt{fst}\ T\ P\ X)$
- $(\kappa\ T\ (\texttt{psub}\ T\ P)\ X) \hookrightarrow (\texttt{pair}\ T\ P\ X\ ?X)$

User-friendly stacks

```
symbol pushTopPop : Prf
  ∀ stk (λ s,
  (nestkp s) ⇒
  λ h, (fst (push (top (pair s h)) (pop (pair s h)))) = s);
```

User-friendly stacks

symbol $\texttt{pushTopPop} : \texttt{Prf}$
$\forall\, \texttt{stk}\, (\lambda s,$
$(\texttt{nestkp}\, s) \Rightarrow$
$\lambda h, (\texttt{fst}\ (\texttt{push}\ (\texttt{top}\ (\texttt{pair}\, s\, h))\ (\texttt{pop}\ (\texttt{pair}\, s\, h)))) = s);$

symbol
$\texttt{pushTopPop} : (\texttt{Prf}\ (\forall\, \texttt{stk}\, \lambda s, ((\texttt{nestkp}\, s) \Rightarrow (\lambda h, (\texttt{push}\ (\texttt{top}\, s)\ (\texttt{pop}\, s))) = s)))$

User-friendly stacks

```
symbol pushTopPop : Prf
  ∀ stk (λs,
  (nestkp s) ⇒
  λh, (fst (push (top (pair s h)) (pop (pair s h)))) = s);
```

```
symbol
pushTopPop : (Prf (∀ stk λs, ((nestkp s) ⇒ (λh, (push (top s) (pop s))) = s)))
begin
  assume h;
  refine h;
  refine h;
end;
```

Wrapping up

Prototypical!