Simple Pure Type Systems Examples

Patrik Jansson

Chalmers University of Technology and University of Gothenburg

Chalmers FP workshop 2010

What is the type of the type of the identity function?

$$\boxed{\lambda a : *.\ \lambda x : a.\ x} : \boxed{\forall\ a : *.\ \forall\ x : a.\ a} : \boxed{?}$$

At what levels do *id*, *Id*, *I*, *Endo*, *F* live?

What is a Pure Type System (PTS)?

[Barendregt, 1992]

Definition (abstract syntax)

The abstract syntax for the terms $\mathcal{T}$ of the system is as follows:

$$
\begin{array}{rcll}
\mathcal{T} & = & C & \text{constants (incl. } \textit{sorts}\text{)} \\
 & | & V & \text{variable} \\
 & | & \mathcal{T}\,\mathcal{T} & \text{application} \\
 & | & \lambda V : \mathcal{T} . \mathcal{T} & \text{abstraction} \\
 & | & \forall V : \mathcal{T} . \mathcal{T} & \text{(dep.) function space}
\end{array}
$$

(Syntactic sugar: $A \to B = \forall\ \_ : A.\ B$)

Parametrised over sorts ($s \in \mathcal{S} \subseteq C$, for typing types),
axioms ($c : s \in \mathcal{A}$, for typing constants) and
rules ($(s_1, s_2, s_3) \in \mathcal{R}$, for typing functions).

Examples

$$\mathcal{T} = C \mid V \mid \mathcal{T}\,\mathcal{T} \mid \lambda V : \mathcal{T}.\mathcal{T} \mid \forall V : \mathcal{T}.\mathcal{T}$$

Typical terms (with names):

- $F = \forall\, t : *.\; t$
- $Id = \forall\, a : *.\; a \to a = \forall\, a : *.\; \forall\, x : a.\; a$
- $Endo = \lambda t : *.\; t \to t = \lambda t : *.\; \forall\, x : t.\; t$
- $I = \lambda t : *.\; t$
- $id = \lambda a : *.\; \lambda x : a.\; x$

Examples

$$\mathcal{T} = C \mid V \mid \mathcal{T}\,\mathcal{T} \mid \lambda V : \mathcal{T}.\mathcal{T} \mid \forall V : \mathcal{T}.\mathcal{T}$$

Typical terms (with names):

- $F = \forall\, t : *.\ t$
- $Id = \forall\, a : *.\ a \rightarrow a = \forall\, a : *.\ \forall\, x : a.\ a$
- $Endo = \lambda t : *.\ t \rightarrow t = \lambda t : *.\ \forall\, x : t.\ t$
- $I = \lambda t : *.\ t$
- $id = \lambda a : *.\ \lambda x : a.\ x$

Typical sorts: $*$, $*_1$

Examples

$$\mathcal{T} = \mathcal{C} \mid \mathcal{V} \mid \mathcal{T}\,\mathcal{T} \mid \lambda\mathcal{V} : \mathcal{T}.\mathcal{T} \mid \forall\mathcal{V} : \mathcal{T}.\mathcal{T}$$

Typical terms (with names):

- $F = \forall\, t : *.\ t$
- $Id = \forall\, a : *.\ a \to a = \forall\, a : *.\ \forall\, x : a.\ a$
- $Endo = \lambda t : *.\ t \to t = \lambda t : *.\ \forall\, x : t.\ t$
- $I = \lambda t : *.\ t$
- $id = \lambda a : *.\ \lambda x : a.\ x$

Typical sorts: $*$, $*_1$

Typical axioms:

- $* : *_1$
- $Bool : *$

Some non-axioms:

- $List : * \to *$ because $(* \to *)$ is not a sort
- $Succ\ n : *$ because $(Succ\ n)$ is not a constant

Definition (PTS typing judgements)

$$\frac{\Gamma \vdash A : s}{\Gamma, x : A \vdash x : A}\ \text{start} \qquad \frac{\Gamma \vdash A : B \qquad \Gamma \vdash C : s}{\Gamma, x : C \vdash A : B}\ \text{weakening}$$

$$\frac{}{\vdash c : s}\ \text{axiom} \qquad \frac{\Gamma \vdash F : (\forall x : A.\, B) \qquad \Gamma \vdash a : A}{\Gamma \vdash F\, a : B[x \mapsto a]}\ \text{application}$$

$$\frac{\Gamma \vdash A : s_1 \qquad \Gamma, x : A \vdash B : s_2}{\Gamma \vdash (\forall x : A.\, B) : s_3}\ \text{product}\ (s_1, s_2, s_3)$$

$$\frac{\Gamma, x : A \vdash b : B \qquad \Gamma \vdash (\forall x : A.\, B) : s_3}{\Gamma \vdash (\lambda x : A.\, b) : (\forall x : A.\, B)}\ \text{abstraction}$$

$$\frac{\Gamma \vdash A : B \qquad \Gamma \vdash B' : s \qquad B =_\beta B'}{\Gamma \vdash A : B'}\ \text{conversion}$$

Examples cont.

$\mathcal{T} = C \mid V \mid \mathcal{T}\,\mathcal{T} \mid \lambda V : \mathcal{T} . \mathcal{T} \mid \forall V : \mathcal{T} . \mathcal{T}$

Typical rules:

- Syntactic sugar: $s1 \leadsto s2 = (s1, s2, s2)$
- $* \leadsto * = (*, *, *)$ simply typed lambda calculus
- add $*_1 \leadsto *_1 = (*_1, *_1, *_1)$ types dep. on types
- add $* \leadsto *_1$ for types dep. on values
- add $*_1 \leadsto *$ for (impred.) values dep. on types (parametric polymorphism)
- replace with $(*_1, *, *_1)$ for (simplified) Agda

System F has rules $* \leadsto *$, $*_1 \leadsto *_1$ and $*_1 \leadsto *$

So, what is the type of the type of id?

$$id = \lambda a : *.\ \lambda x : a.\ x$$
$$Id = \forall\ a : *.\ \forall\ x : a.\ a$$
$$\boxed{\ id\ } : \boxed{Id} : \boxed{?}$$

$$\frac{\Gamma, x : A \vdash b : B \qquad \Gamma \vdash (\forall x : A.\ B) : s_3}{\Gamma \vdash (\lambda x : A.\ b) : (\forall x : A.\ B)}\ \text{abstraction}$$

$$\frac{\Gamma \vdash A : s_1 \qquad \Gamma, x : A \vdash B : s_2}{\Gamma \vdash (\forall x : A.\ B) : s_3}\ \text{product}\ (s_1, s_2, s_3)$$

It depends on the rules:

- In simply typed λ-calculus: *id* is not type correct.
- In System F and the "Calculus of Constructions": $Id : *$
- In Agda: $Id : *_1$

Summarising

In Agda *id*, *Endo* and *I* are on the "same level":

```
         F         :  *1
id    :  Id        :  *1
Endo  :  * -> *    :  *1
I     :  * -> *    :  *1
```

In CC the situation is different: Id, Endo and I are on the "same level" (while id is different):

```
        F        : *      : *1
 id   : Id       : *      : *1
        Endo     : * -> * : *1
        I        : * -> * : *1
```