Encoding : Booleans

In the untyped setting

true $= \lambda x y . x$

false $= \lambda x y . y$

if b then t else e $= b\ t\ e$

In System F :

$\mathbb{B} = \forall \alpha . \alpha \to \alpha \to \alpha$

true $= \Lambda \alpha . \lambda x : \alpha . \lambda y : \alpha . x$

false $= \Lambda \alpha . \lambda x : \alpha . \lambda y : \alpha . y$

if b then t else e

$= b [\tau]\ t\ e$

where $t : \tau$, $e : \tau$

Note that both branches t, e must have the same type τ

In programming languages, this is an (over-) approximation of the run-time behavior : e.g.

if true
then 5
else false

is ill-typed, but we could let it have type $\mathbb{N}$ without problems.

Note that, since $\mathbb{B} \simeq 1 + 1$ we could use the encoding for + instead. Indeed

$\ulcorner 1 + 1 \urcorner = \forall \alpha . (1 \to \alpha) \to (1 \to \alpha) \to \alpha$

$\simeq \forall \alpha . \alpha \to \alpha \to \alpha$

since $1 \to \alpha \simeq \alpha$ $\quad (A^1 \simeq A)$

Encoding : $\mathbb{N}$

In untyped λ :

zero $= \lambda s z.\ z$

succ $= \lambda n s z.\ s\ (n\ s\ z)$

Idea : $\ulcorner n \urcorner\ (f) = \underbrace{f \circ f \circ \cdots \circ f}_{n \text{ times}}$

In System F :

$\ulcorner \mathbb{N} \urcorner = \forall \alpha.\ (\alpha \to \alpha) \to \alpha \to \alpha$

zero $= \Lambda \alpha.\ \lambda s : (\alpha \to \alpha).\ \lambda z : \alpha.\ z$

succ $= \lambda n : \ulcorner \mathbb{N} \urcorner.$

$\Lambda \alpha.\ \lambda s : (\alpha \to \alpha).\ \lambda z.\ \alpha.$

$s\ (n\ [\alpha]\ s\ z)$

iter $n\ f\ x = f^n\ (x)$

In untyped : iter $n\ f\ x = n\ f\ x$

In System F :

iter $n\ f\ x = n\ [\tau]\ f\ x$

where $n : \ulcorner \mathbb{N} \urcorner$, $f : \tau \to \tau$, $x : \tau$

untyped : Add $= \lambda n m.\ n$ succ m

System F :

Add $= \lambda n : \ulcorner \mathbb{N} \urcorner.\ \lambda m : \ulcorner \mathbb{N} \urcorner.$

$n\ [\ulcorner \mathbb{N} \urcorner]$ succ m

Curry - Howard

System F $\simeq$ propositional intuitionistic second-order logic

$$\frac{\Gamma \vdash q \qquad p \text{ not free in } \Gamma}{\Gamma \vdash \forall p.\, q}\ (\forall I)$$

$$\frac{\Gamma \vdash \forall p.\, q}{\Gamma \vdash q\{r/p\}}\ (\forall E)$$

The C-H iso. still holds

e.g.

$\vdash (\forall p, q.\, ((p \to q) \wedge p) \to q$

hence $\exists t$

$\vdash t : \forall \alpha, \beta.\, ((\alpha \to \beta) \times \alpha) \to \beta$

$\vdash \bot\!\!\!\bot \iff (\forall p.\, p)$

$\vdash \top\!\!\!\top \iff (\forall p.\, p \to p)$

$\vdash (q \wedge r) \iff (\forall p.\, (q \to r \to p) \to p)$

$\vdash (q \vee r) \iff (\forall p.\, (q \to p) \to (r \to p) \to p)$

Ex: prove the above

Models for $\lambda^{\forall}$ (hints)

Interpreting $\forall\alpha.\tau$ is complex.

If types are interpreted as objects in a cat. $\mathcal{C}$, the "naive" interpretation is a family of objects

$$\{ [\![\tau]\!]_{\theta[\alpha \mapsto A]} \}_{A \in |\mathcal{C}|}$$

this must $\in |\mathcal{C}|$!

This easily fails in many categories:
e.g. in Set

$$[\![\forall\alpha.\tau]\!] = (\mathrm{Set} \longrightarrow \mathrm{Set})$$
$$A \longmapsto [\![\tau]\!]_{\theta[\alpha \mapsto A]}$$

but $(\mathrm{Set} \to \mathrm{Set})$ is a proper class !
We can't use "functions" whose domain is Set and claim those form a set.

We need more complex categories to model System F: for instance
- Coherence Spaces
- PERs (Partial Equivalence Relations)

(We will not describe these - possible projects)