Retour sur Curry-Howard

On peut prendre System-F comme une logique:

- Correspond (presque) à la logique classique propositionnelle
- Qu'en est-il de la logique des prédicats?

Comment dire (et prouver): $\forall x, y \in \mathbb{N}, x + y = y + x$?

Nombres naturels et addition

```
Nat : Type;
type Nat
   | zero
   | succ Nat;

_+_ : Nat → Nat → Nat;
x + y = case x
   | zero ⇒ y
   | succ n ⇒ succ (n + y);
```

Nombres naturels et addition de Church

```
Nat : Type;
Nat     = (t : Type) → t → (t → t) → t;
zero    = λt (x:t) (f:t→t) → x;
succ n = λt (x:t) (f:t→t) → f (n t x f);

_+_ : Nat → Nat → Nat;
x + y = x Nat y succ;
```

L'égalité, comme un type?

Il faut définir une égalité:

```
_=_ : Nat → Nat → Type;
```

Mais on sait que `Nat : Type` et `Type` : $\square$!

Donc il faut dans R qqch du genre $(\texttt{Type}, \square, ?)$

En clair: des fonctions de valeurs à types!

Aussi connu comme: des *types dépendents*

Le calcul des constructions

CoC = System-F_ω + types dépendents:

$$(\{*, \Box\}, \{* : \Box\}, \{(*, *, *), (\Box, *, *), (\Box, \Box, \Box), (*, \Box, \Box)\})$$

Encore *strongly normalizing* $\Rightarrow$ logique *cohérente*!

Correspond à une sorte de logique des prédicats d'ordre supérieur

Dorénavant, $(x : \tau_1) \to \tau_2$ n'est plus nécessairement $\tau_1 \to \tau_2$

Cependant t est encore *effaçable* dans $\lambda(t : \kappa) \to e$

Égalité de Leibniz

$$x = y \quad \Leftrightarrow \quad \forall P.P(x) \Rightarrow P(y)$$

Pour le cas qui nous intéresse:

```
(x = y) = (P : Nat → Type) → P x → P y;
```

Cas de base: preuve de la réflexivité et commutativité

```
refl : (x : Nat) → (x = x);
refl x = λ(P : Nat → Type) → λ(p : P x) → p;

comm : (x = y) → (y = x);
comm p = p (λ(y′ : Nat) → (y′ = x)) (refl x);
```

Limites de l'encodage imprédicatif

Essayons de prouver: $0 \neq 1$

```
obvious? : (0 = 1) → ⊥;
obvious? = λ(p : (0 = 1)) →
             p (λ(x : Nat)
                → x Type Unit (λ_ → ⊥))
               unit;
```

Le premier argument de x doit être un *type*, pas un *kind*!

D'autres PTS

$\lambda*$ est un PTS extrême; aussi décrit par `Type : Type`

$$\lambda* \quad \equiv \quad (\{*\}, \{* : *\}, \{(*, *, *)\})$$

System-U$^-$: System-F$_\omega$ où STLC est remplacé par System-F!

$$(\{*, \Box, \triangle\}, \{* : \Box, \Box : \triangle\},$$
$$\{(*, *, *), (\Box, *, *), (\Box, \Box, \Box), (\triangle, \Box, \Box)\})$$

Ni l'un ni l'autre ne sont *strongly normalizing*

On peut y trouver un terme de type $\bot$!

Mais c'est toujours un terme qui ne termine pas!

Reality check

Pas *strongly normalizing* $\Rightarrow$ typage non-décidable

- Pas grave en soi

Effets de bord $\Rightarrow$ casse tout (e.g *readNat* "N=" $=$ *readNat* "N=")

Preuves qui ne terminent pas $\Rightarrow$ pas effaçable!

Imprédicativité

Dans System-F la deuxième règle est *imprédicative*:

```
Unit = (t : Type) → t → t;
unit = λ(t : Type) → λ(x : t) → x;
unitUnit = unit Unit;
```

Le quantificateur $\forall$ s'applique à lui-même!

La représentation *extensionnelle* de *id* est très infinie:

- elle associe à chaque valeur, cette même valeur
- elle associe aussi *id* à *id*

$\Rightarrow$ C'est une fonction infinie qui se contient elle-même!

System-F_ω prédicatif

$$(\{*, \Box\}, \{* : \Box\},$$
$$\{(*, *, *), (\Box, *, \Box), (\Box, \Box, \Box)\})$$

Une fonction comme *id* vit alors dans la sorte $\Box$:

- On ne peut plus passer son type à un Λ
- On ne peut plus la passer en argument à une fonction
- Cependant, on peut l'appeler exactement comme avant
- *id* `Int` est une fonction normale (dans la sorte $*$)

E.g. on peut encore faire: `id (Int → Int) (id Int)`

PTS avec univers

Éviter l'imprédicativité excessive de $\lambda*$ par stratification:

$$\begin{aligned}
&(\{\texttt{Type}\ n && \mid n \in \mathbb{N}\},\\
&\ \{\texttt{Type}\ n_1 : \texttt{Type}\ n_2 && \mid n_2 = n_1 + 1\},\\
&\ \{(\texttt{Type}\ n_1, \texttt{Type}\ n_2, \texttt{Type}\ n_3) && \mid n_3 = \mathit{max}\,(n_1, n_2)\})
\end{aligned}$$

Pas d'imprédicativité du tout

- Type d'une fonction toujours "plus haut" que ses arguments

Système *strongly normalizing* et donc logique *cohérente*

Variantes: *universe inclusion*, *universe polymorphism*