Rapport TP1

Valeran MAYTIE

1 Prise en main du logiciel Isabelle

2 Encodage de Church

On cherche à encoder les entiers naturels en λ-calcul pur. Pour cela on utilise l'encodage de Church, il est construit avec une fonction qui prend en paramètre deux variables f une fonction et x une variable. Pour représenter un nombre n on compose n fois la fonction f appliquée à x.

Le λ-term ressemble à :

$$\lambda f.\lambda x.f^n\ x \equiv \lambda f.\lambda x.\ \underbrace{f\ (f\ \ldots\ (f}_{n \text{ fois}}\ x))$$

En Isabelle nous définissons les entiers naturels de 0 à 5 comme ceci :

```
definition ZERO  where "ZERO  ≡ λf x. x "
definition ONE   where "ONE   ≡ λf x. f x"
definition TWO   where "TWO   ≡ λf x. f (f x)"
definition THREE where "THREE ≡ λf x. f (f (f x))"
definition FOUR  where "FOUR  ≡ λf x. f (f (f (f x)))"
definition FIVE  where "FIVE  ≡ λf x. f (f (f (f (f x))))"
```

Figure 1: Entier de Church de 0 à 5

On peut définir des opérations sur cet encodage.

$$\begin{array}{rcl} n+1 & : & \lambda n\ f\ x.\ f(n\ f\ x) \\ n+m & : & \lambda n\ m\ f\ x.\ m\ f\ (n\ f\ x) \\ n \times m & : & \lambda n\ m\ f\ x.\ n\ (m\ f)\ x \\ n^m & : & \lambda n\ m\ f\ x.\ m\ n \end{array}$$

Malheureusement les opérations prédécesseur et soustraction sont plus difficile à écrire.

En Isabelle on défini uniquement l'addition (Figure-2) car ça sera la seule utile pour notre première preuve.

```
definition PLUS where "PLUS ≡ λn m f x. m f (n f x)"
```

Figure 2: Addition avec l'encodage de Church

Notre première démonstration consiste à prouver que $3 + 2 = 5$ en entier de Church. L'énoncé s'écrit comme ceci : `PLUS TWO THREE = FIVE`. Pour commencer, il faut dérouler toutes les définitions. On utilise donc la tactique `unfolding` avec comme argument la définition à déplier suivit de "*_ def*" (par exemple pour la défintion *PLUS* : *PLUS_ def*).

Le dépliage va effectuer tout les calculs possibles on aura donc :

$$\texttt{PLUS TWO THREE} \ \rightarrow^*_\beta\ \lambda f\ x.\ f\ (f\ (f\ (f\ (f\ x))))$$

Nous remarquons que `PLUS TWO THREE` se réduit en `FIVE`, il nous reste donc à monter que `FIVE = FIVE`. Nous avons vue en cours que l'égalité dans Isabelle est réflexiver ($\forall x, x = x$). Il suffit donc d'appliquer le théorème de réflexivité *HOL.refl* que l'on peut trouver à l'aide de la commandel `find_theorems "_ = _"`. On peut appliquer ce théorème en utilisant `apply(rule refl)`. La preuve complète de deux ligne :) se trouve ci dessous (Figure-3).

```
lemma the_first : "PLUS TWO THREE = FIVE"
  unfolding PLUS_def TWO_def THREE_def FIVE_def
  apply(rule refl)
  done
```

Figure 3: Preuve que $2 + 3 = 5$ avec l'encodage de Church