NOLLE

Damien

L3 - Informatique

MF - Devoir 3 :

Note :

/20

Exercice 1)

Q1) Un λ terme de type : $(\alpha \to (\beta \to \gamma)) \to ((\alpha \to \beta) \to (\alpha \to \gamma))$ est une fonction d'arité 1, donc :

$\lambda x_1 . e_1$

avec :

$x_1 : \alpha \to (\beta \to \gamma)$,

$e_1 : (\alpha \to \beta) \to (\alpha \to \gamma)$.

e_1 est aussi une fonction d'arité 1, donc :

$e_1 = \lambda x_2 . e_2$

avec :

$x_2 : \alpha \to \beta$,

$e_2 : \alpha \to \gamma$.

e_2 est aussi une fonction d'arité 1, donc :

$e_2 = \lambda x_3 . e_3$

avec :

$x_3 : \alpha$

$e_3 : \gamma$

Donc : $\lambda x_1 . \lambda x_2 . \lambda x_3 . e_3$

NOLLE
Damien

Pour des raisons de lisibilité :

$x_1 = f$,

$x_2 = g$,

et $x_3 = x$.

L'expression e_3 doit permettre d'obtenir un résultat de type τ, en utilisant f, g et x.

Donc :

$e_3 = f(x)(g(x)) : \tau$

et

$\lambda f . \lambda g . \lambda x . f(x)(g(x))$.

Q2)

$[f : \alpha \to \beta \leftarrow g : \alpha \to \beta,\ e : \beta \leftarrow x : \alpha]$

$[f : \alpha \to \beta \leftarrow f : \alpha \to (\beta \to \tau),\ e : \alpha \leftarrow x : \alpha]$ (app.)

$$\dfrac{\dfrac{\dfrac{\dfrac{[f : \alpha \to (\beta \to \tau), x : \alpha]\ [g : \alpha \to \beta], [x : \alpha]}{f(x) : \beta \to \tau,\ g(x) : \beta}\ \text{(app.)}}{x : \alpha \vdash f(x)(g(x)) : \tau}}{g : \alpha \to \beta \vdash \lambda x . f(x)(g(x)) : \alpha \to \tau}\ \text{(abs.)}}{f : \alpha \to (\beta \to \tau) \vdash \lambda g . \lambda x . f(x)(g(x)) : (\alpha \to \beta) \to (\alpha \to \tau)}\ \text{(abs.)}$$

$[e : \beta \leftarrow f(x)(g(x)) : \tau]$ (abs.)

$[x : \alpha \leftarrow g : \alpha \to \beta,\ e : \beta \leftarrow f(x)(g(x)) : \beta \to \tau]$ (abs.)

$$\dfrac{f : \alpha \to (\beta \to \tau) \vdash \lambda g . \lambda x . f(x)(g(x)) : (\alpha \to \beta) \to (\alpha \to \tau)}{\lambda f . \lambda g . \lambda x . f(x)(g(x)) : (\alpha \to (\beta \to \tau)) \to ((\alpha \to \beta) \to (\alpha \to \tau))}\ \text{(abs.)}$$

$[x : \alpha \leftarrow f : \alpha \to (\beta \to \tau),\ e : \beta \leftarrow \lambda g . \lambda x . f(x)(g(x)) : (\alpha \to \beta) \to (\alpha \to \tau)]$

Exercice 3)

Q1)

$$\dfrac{\dfrac{\overline{a : T_F},\ \overline{b : T_F}}{r(a, b) : T_F}\ [s \leftarrow a,\ t \leftarrow b] \qquad \overline{c : T_F}}{i(r(a, b), c) : T_F}\ [s \leftarrow r(a, b),\ t \leftarrow c]$$

NOLLE
Damien

Q3)

reduce(x) = x si x ∈ {a, b, c}

reduce(p(s, t)) = p(reduce(s), reduce(t))

reduce(i(p(s, t), u)) = reduce(i(reduce(s), reduce(i(t, u))))

reduce(i(x, t)) = i(x, reduce(t)) si x ∈ {a, b, c}

reduce(i(i(s, t), u)) = reduce(i(reduce(i(s, t)), reduce(u)))

Q4)

Si on prend :

reduce(i(p(s, t), u)) = i(reduce(s), i(reduce(t), reduce(u)))

Et la formule : i(p(p(a, b), c), b), par exemple, elle ne sera pas suffisamment réduite avec l'égalité précédente comme cas de la fonction reduce.

i(p(p(a, b), c), b)
= i(reduce(p(a, b)), i(reduce(c), reduce(b)))
<Règle 4 (correspondant donc à l'égalité précédente), subst. [s ← p(a, b), t ← c, u ← b]>
= i(p(reduce(a), reduce(b)), i(reduce(c), reduce(b)))
<Règle 3, subst. [s ← a, t ← b]>
= i(p(a, reduce(b)), i(reduce(c), reduce(b)))
<Règle 2, subst. [x ← a]>
= i(p(a, b), i(reduce(c), reduce(b)))
<Règle 2, subst. [x ← b]>
= i(p(a, b), i(c, reduce(b)))
<Règle 2, subst. [x ← c]>
= i(p(a, b), i(c, b))
<Règle 2, subst. [x ← b]>

Il reste encore p(a, b) qui n'est pas réduit selon

NOLLE
Pomien

La fonction reduce

Q5)

$nbp : T_F \rightarrow \mathbb{N}$

$nbp(x) = 0$ (8)
$nbp(p(s,t)) = 1 + nbp(s) + nbp(t)$ (9)
$nbp(i(s,t)) = nbp(s) + nbp(t)$ (10)

Q6)

$nbp(i(p(x,y),z))$
$= nbp(p(x,y)) + nbp(z)$
< Règle 10, sub. [$s \leftarrow p(x,y)$, $t \leftarrow z$] >
$= 1 + nbp(x) + nbp(y) + nbp(z)$
< Règle 9, sub. [$s \leftarrow x$, $t \leftarrow y$] >

$1 + nbp(i(x, i(y,z)))$
$= 1 + nbp(x) + nbp(i(y,z))$
< Règle 10, sub. [$s \leftarrow x$, $t \leftarrow i(y,z)$] >
$= 1 + nbp(x) + nbp(y) + nbp(z)$
< Règle 10, sub. [$s \leftarrow y$, $t \leftarrow z$] >

Q7) Démontrons $nbp(reduce(n)) \leq nbp(t)$ par induction sur n :

- Cas de base :
Considérons $x \in \{a, b, c\}$ car la démonstration est la même pour les 3 constantes.

Donc : $P(x) = nbp(reduce) \leq nbp(t)$

NOLLE
Pomien

nbp (reduce (x))
= nbp (x)
< Règle 2 >
= 0
< Règle 8 >

Quel que soit t, si t contient plusieurs ou aucun p, $0 \leq$ nbp (t) est vrai.

Si t est une constante, alors nbp (t) = 0

Si t est une formule contenant au moins un symbole p, alors nbp (t) > 0.

- Cas d'induction :
Quelque soit les formules n et t, reduce (n) réduit et retire tous les p de n, de ce fait, nbp (reduce (n)) sera toujours égal à 0.

Quelque soit le nombre de termes de t >= 1, si t contient au moins p, alors nbp (t) > 0 et donc nbp (t) > nbp (reduce (n)).

Même si t ne contient aucun p et donc nbp (t) = 0, alors nbp (t) = nbp (reduce (n)).

Donc nbp (reduce (n)) $\geq$ nbp (t) est toujours vrai, pour tout n : T_F et pour tout t : T_F.

Le soucis de la preuve par induction sur T_F est qu'on ne peut pas déterminer des hypothèses qui

NOLLE
Pomien

seront utilisées lors de la démonstration du pas d'induction. Ce qui s'explique par le fait que la fonction nlp change complètement la structure de la formule.

Q2) On peut affirmer que l'application de cette égalité (1) termine toujours en fonction de sa structure.

En effet, à chaque application de cette égalité, un symbole $\neg$ sera retiré de la formule ou d'un terme de cette dernière, qui seront de la forme : $i(\neg(x,y), z)$.

Etant donné que cette égalité n'introduit pas de nouveaux symboles $\neg$, et que le nombre de symbole $\neg$ ainsi que des termes ayant la structure : $i(\neg(x,y), z)$ sera forcément fini, il n'y a pas de risque de répétition de cette égalité, empêchant sa terminaison.