DEA Programmation : Sémantique, Preuve et Langages 2003-2004
Option *Calcul des constructions inductives*

Projet : fonctions récursives et machines de Minski

Ce projet est individuel. Vous devez rendre un rapport écrit (succinct, expliquant essentiellement quelles ont été vos difficultés et comment vous les avez résolues) et le source Coq correspondant, dans une archive (les fichiers `.v` et le rapport en Postscript ou PDF) envoyée par mail à `Claude.Marche@lri.fr`. De plus une soutenance aura lieu le 29/03/2004. La note finale est le max des notes du projet et de l'examen.

Votre travail sera plus jugé sur la qualité que sur la quantité, aussi il n'est pas impératif de tout faire.

L'objet du projet est de montrer que le formalisme des machines de Minski est suffisant pour exprimer toutes les fonctions récursives, ceci est construisant un compilateur certifié vers ces machines.

On note F_k l'ensemble des fonctions partielles de $\mathbb{N}^k$ dans $\mathbb{N}$. On note $\pi_{i,k}$ la fonction de F_k définie par $\pi_{i,k}(x_0,\ldots,x_{k-1}) = x_i$, pour $0 \leq i < k$.

On définit les listes dépendantes d'entiers par le type inductif :

```
Inductive listnat : nat -> Set :=
| Lnil : listnat 0
| Lcons : forall l:nat, nat -> listnat l -> listnat (S l).
```

1 Modélisation des fonctions récursives

Pour chaque $k \in \mathbb{N}$, l'ensemble R_k des fonctions entières récursives à k arguments est le plus petit sous-ensemble de F_k tel que

- Les fonctions de F_0 (c-à-d les constantes) sont dans R_0.
- La fonction nulle et la fonction successeur de F_1 sont dans R_1.
- Pour tout i, $0 \leq i < k$, $\pi_{i,k}$ est dans R_k.
- Pour toute fonction f de R_k, et toute suite de k fonctions $g_1,\ldots,g_k$ de R_i, la fonction h *composée* de f et $g_1,\ldots,g_k$, notée $Compose(f,[g_1,\ldots,g_k])$, définie par
$$h(x_0,\ldots,x_{i-1}) = f(g_1(x_0,\ldots,x_{i-1}),\ldots,g_k(x_0,\ldots,x_{i-1}))$$
est dans R_i.
- Pour toute fonction f de R_k et toute fonction g de R_{k+2}, la fonction h obtenue par *récursion primitive* avec f et g, notée $Prim(f,g)$, définie par
$$\begin{aligned} h(0,x_1,\ldots,x_k) &= f(x_1,\ldots,x_k) \\ h(x+1,x_1,\ldots,x_k) &= g(x,h(x,x_1,\ldots,x_k),x_1,\ldots,x_k) \end{aligned}$$
est dans R_{k+1}.

- Pour toute fonction f de R_{k+1}, la fonction h obtenue par *minimisation* de f, notée $Min(f)$, définie par

$$h(x_1,\ldots,x_k) = \min\{n \mid f(n,x_1,\ldots,x_k) = 0\}$$

si ce min existe, et indéfinie sinon, est dans R_k.

1. *Définir en Coq un type dépendant* `Recfun: nat -> Set`, *tel que* `(Recfun k)` *modélise les fonctions récursives à k arguments. Définir la constante Coq* `Add` *de type* `(Recfun 2)` *qui représente la fonction* $Prim(\pi_{0,1}, Compose(Succ, [\pi_{1,3}]))$.
2. *Définir un prédicat*

   ```
   RecSem: forall k:nat, (Recfun k) -> (listnat k) -> nat -> Prop
   ```

   tel que (`RecSem` k f $[x_0,\ldots,x_{k-1}]$ v) *spécifie le fait que la fonction récursive* f *prend, sur les arguments* $x_0,\ldots,x_{k-1}$, *la valeur* v. *Prouver que la sémantique de* `Add` *est bien l'addition des entiers.*
3. *Construire de même une constante* `Mult` *de type* `(Recfun 2)` *pour représenter la multiplication, et prouver que votre construction a bien la sémantique de la multiplication.*
4. *Définir une constante* `Sqrt` *pour représenter la fonction récursive* $sqrt(x) = y$ *si* $y^2 = x$, *indéfinie si* x *n'est pas un carré. Prouver qu'elle a la bonne sémantique. Suggestion : on pourra construire successivement des définitions récursives pour les fonctions :*

$$\begin{aligned} null(x) &= 0 \text{ si } x = 0, \text{ 1 sinon} \\ null2(x,y) &= 0 \text{ si } x = 0 \text{ et } y = 0, \text{ 1 sinon} \\ pred(x) &= x-1 \text{ si } x > 0, \text{ 0 sinon} \\ subinv(x,y) &= y-x \text{ si } y > x, \text{ 0 sinon} \\ eq(x,y) &= 0 \text{ si } x = y, \text{ 1 sinon} \end{aligned}$$

puis se baser sur la formule $sqrt(x) = \min\{y \mid y \times y = x\}$

Remarque : les questions 3 et 4 ne sont pas essentielles pour la suite, elle peuvent être omises dans un premier temps.

2 Les machines de Minski

Une machine de Minski à n registres est un quadruplet (Q, q_i, q_f, δ) où Q (l'ensemble des états) est un ensemble fini, $q_i \in Q$ est l'état initial, $q_f \in Q$ est l'état final, et δ (la fonction de transition) est une application de $Q - \{q_f\}$ dans $(\{1,\ldots,n\} \times Q) \cup (\{1,\ldots,n\} \times Q \times Q)$. Une configuration de la machine est la donnée d'un état et de n entiers naturels $r_0,\ldots,r_{n-1}$ que l'on appelle les registres. Depuis une configuration $(q, r_0,\ldots,r_{n-1})$ la transition s'effectue comme suit : si $q = q_f$ la machine s'arrête, sinon

- si $\delta(q) = (i, q')$ alors on passe dans la configuration $(q', r_0,\ldots,r_i + 1,\ldots r_{n-1})$;

- si $\delta(q) = (i, q', q'')$ alors si $r_i = 0$ on passe dans la configuration $(q', r_0, \ldots, r_i, \ldots r_{n-1})$ sinon dans la configuration $(q'', r_0, \ldots, r_i - 1, \ldots r_{n-1})$.

La fonction partielle $f : \mathbb{N}^k \to \mathbb{N}$ calculée par une machine de Minski à au moins k registres est définie comme suit : pour tout k-uplet $(x_0, \ldots, x_{k-1})$, si à partir de la configuration $(q_i, x_0, \ldots, x_{k-1}, 0, \ldots, 0)$ la machine s'arrête avec la valeur r dans le registre r_0, alors $f(x_0, \ldots, x_{k-1}) = r$, et si elle ne s'arrête pas alors $f(x_0, \ldots, x_{k-1})$ est indéfinie.

On pourra décrire une machine de façon plus lisible par un pseudo programme formé d'une suite d'instructions où chaque instruction est soit de la forme

q : $r_i := r_i + 1$; goto q'

qui signifie $\delta(q) = (i, q')$, soit de la forme

q : if $r_i = 0$ then goto q' else $r_i := r_i - 1$; goto q"

qui signifie $\delta(q) = (i, q', q'')$. Pour simplifier les choses on considèrera toujours que l'ensemble des états est $\{0, \ldots, m\}$ où 0 est l'état initial et m est l'état final.

Remarque : dans la suite, on pourra rendre les types demandés (`Minski`, `config`, etc.) dépendants si on le souhaite.

1. *Définir un type* `Minski` *pour modéliser une machine de Minski. Dans votre représentation, définir une constante* `MAdd` *qui représente la machine*

   0 : if $r_1 = 0$ *then goto 2 else* $r_1 := r_1 - 1$ *; goto 1*
   1 : $r_0 := r_0 + 1$ *; goto 0*

2. *Définir un type* `config` *pour modéliser une configuration, une fonction*

   ```
   step: Minski -> config -> config
   ```

   telle que (`step` M c) *donne la configuration obtenue en un pas d'exécution de* M *sur* c *(qui retournera* c *quand il n'y a pas de transition possible), et un prédicat*

   ```
   transition: Minski -> config -> config -> Prop
   ```

   tel que (`transition` M c_1 c_2) *affirme qu'il existe une suite de transitions de* c_1 *à* c_2 *par* M.

3. *Définir un prédicat qui donne la sémantique d'une machine de Minski :* (`MinskiSem` M $[x_0, \ldots, x_{k-1}]$ v) *affirme que, exécutée sur une configuration où* $r_i = x_i$*, la machine* M *s'arrête avec la valeur* v *dans* r_0*. Puis prouver que* `MAdd` *calcule l'addition.*

4. *Définir un prédicat*

   ```
   termine: Minski -> config -> Prop
   ```

   qui affirme que le calcul de M *s'arrête en temps fini sur la configuration* c*, puis une fonction*

   ```
   execute: M:Minski -> c:config -> (termine M c) -> config
   ```

   qui à partir d'une machine M*, d'une configuration* c *et d'une preuve de terminaison, retourne la configuration finale obtenue. En déduire une fonction qui exécute jusqu'à son arrêt la machine* `MAdd`.

Remarque : la question 4 n'est pas utile pour la suite.

3 Compilation d'une fonction récursive en machine de Minski

Le but est maintenant de définir un schéma de compilation pour construire, à partir d'une définition récursive d'une fonction, une machine de Minski qui la calcule.

1. *Pour tous entiers n et m avec $n \neq m$, construire une machine de Minski Copie(n,m) à $k = \max(n,m)+2$ registres, qui copie la valeur de son registre r_n dans son registre r_m. À la fin de l'exécution de Copie(n,m), tous les registres de numéro inférieur ou égal à $\max(n,m)$, excepté r_m, doivent avoir la même valeur qu'au début. On se servira de r_{k-1} comme registre temporaire. Prouver que Copie(n,m) a bien le comportement attendu.*
2. *En déduire des machines de Minski pour calculer les fonctions de projections $\pi_{i,k}$. Prouver qu'elles ont bien la sémantique attendue.*
3. *Montrer qu'à toute machine de Minski M à k registres on peut associer une autre machine Decale-regs(M,n) (avec $n > 0$) qui travaille comme M mais sur les registres $r_n,\ldots,r_{n+k-1}$, en laissant inchangés les registres $r_0,\ldots,r_{n-1}$.*
4. *Montrer qu'à tout couple de machines (M_1,M_2) on peut associer une machine Enchaine(M_1,M_2) qui revient à exécuter M_1, puis exécuter M_2 sur la configuration obtenue.*
5. *Pour tous entiers n et m avec $n < m$, construire une machine de Minski Copie-multiple(n,m) qui copie les valeurs de ses registres $r_0,\ldots,r_{n-1}$ dans les registres $r_m,\ldots,r_{m+n-1}$ respectivement. À la fin de l'exécution de Copie-multiple(n,m), tous les registres de numéro inférieur ou égal à $m-1$ doivent avoir la même valeur qu'au début.*
6. *En supposant que f et $g_1,\ldots,g_k$ sont calculées respectivement par les machines $M,M_1,\ldots,M_k$, construire une machine qui calcule la composition Compose$(f,[g_1,\ldots,g_k])$*
7. *En supposant données deux machines de Minski calculant respectivement les fonctions f et g, construire une machine de Minski calculant la fonction Prim(f,g). (On pourra écrire un pseudo programme calculant cette fonction par une boucle.)*
8. *En supposant données une machine de Minski calculant la fonction f, construire une machine de Minski calculant la fonction Min(f).*
9. *En déduire une fonction qui à une fonction récursive associe une machine qui la calcule, et prouver la correction de cette compilation.*
10. *Construire une machine de Minski certifiée qui calcule la racine carrée.*