Informatique Théorique
TD5
SI3-MAM3

1 Opérations

Donnez des définitions inductives des fonctions de $\mathbb{N}$ dans $\mathbb{N}$:

- add_m , fonction qui ajoute la constante m
- multiply_by_m, fonction qui multiplie par la constante m

En utilisant ce schéma inductif libre définissant $\mathbb{N}$

- Base={0}
- Constructeurs {succ}

On peut définir la fonction Add_m par
Add_m(0)=m
Add_m(succ(n))=succ($Add_m(n)$)
Ce qui conduit à l'écriture de ce code récursif :

```
public int addM(int n){
if (n==0) return m
else return 1+addM(n-1)
}
```

La fonction $Multiply\_by_m$ peut alors être définie par
$Multiply\_by_m$(0)=0
$Multiply\_by_m(succ(n))$= $Add_m(Multiply\_by_m$(n))
Ou

```
public int Multiply_by_m(int n){
if (n==0) return 0
else return m+ Multiply_by_m (n-1)
}
```

---

2 Modulos

En vous appuyant sur une définition inductive de $\mathbb{N}$, donnez une définition inductive des fonctions suivantes :

- n modulo 2
- division entière de n par 2

On peut aussi utiliser le schéma libre suivant pour définir $\mathbb{N}$:

- Base {0,1}
- Constructeurs $\Omega = \{succ_2\}$ $succ_2(n) = n + 2$

On pourra alors définir la fonction modulo 2 par
0 mod 2 = 0
1 mod 2 = 1
$succ_2(n)$ mod 2 = n mod 2
Ce qui pourrait se programmer sous la forme

```
public int mod2 (int n) {
(if n==0 | n==1)  {return n}
else return (mod2(n-2))}
}
```

On pourra aussi définir la fonction division entière par deux par
0/2 =0
1/2 =0
$(succ_2(n))/2$ = n/2 +1
Ce qui pourrait se programmer sous la forme

```
public int div2 (int n) {
(if n==0 | n==1)  {return 0}
else return (div(n-2)+1 )}
}
```

---

3 Les vraies maintenant

En vous appuyant sur une définition inductive de $\mathbb{N} \times \mathbb{N}$, définir inductivement les fonction suivantes

- Addition de deux entiers
- Multiplication de deux entiers
- Calcul de m^n.

En utilisant pour $\mathbb{N} \times \mathbb{N}$le schéma libre suivant :

- Base : {(0,0)}
- Constructeurs $\Omega = \{\omega_g, \omega\}$ avec $\omega_g((m, 0)) = (succ(m), 0)$ et $\omega((m, n)) = (m, succ(n))$

On peut définir la fonction Add par

- $Add(0, 0) = 0$
- $Add(succ(m), 0) = succ(m)$
- $Add(m, succ(n)) = succ(Add(m, n))$

ou en java ou en C

```
public int add (int m, int n) {
if (n==0) return m
else return add (m,n-1)+1
}

int add(int n, int m)
{
  if (n > 0)
    return add(n - 1, m) + 1;
  else
    return m;
}
```

et la fonction $Mult$ par

- Mult(0,0)=0
- Mult(succ(m),0)=0

- Mult(m,succ(n))=Add(Mult(m,n),m)

qui conduit au code récursif :

```
public int mult (int m, int n) {
if (n==0) return 0
else return mult (m,n-1)+m
}

int mult(int n, int m)
{
  if (n > 0)
    return mult(n - 1, m) + m;
  else
    return 0;
}
```

- Puiss(0,0)=0 – ou 1 c'est affaire de convention ici
- Puiss(succ(m),0)=1
- Puiss(m,succ(n))=Mult(Puiss(m,n),m)

et code récursif

```
public int puiss (int m, int n) {
if (n==0) {if (m==0) return 0 else return 1}
else return puiss (m,n-1)*m
}
```

ou

```
int puiss(int n, int m)
{
  if (n > 0)
    return puiss(n - 1, m) * m;
  else
    return 1;
}
```

---

4 Narcisse

Soit A un alphabet quelconque, définir inductivement la fonction miroir de $A^* \to A^*$.

En utilisant pour A^* le schéma libre suivant :

- ϵ appartient à A^*
- Si a appartient à A, et m appartient à A^*, ma appartient à A^*

miroir (ϵ) = ϵ
miroir(ma) = a miroir(m)

---

5 Ecritures Binaires

Definir inductivement l'ensemble $EcrituresBin$ des écritures binaires des entiers, comme sous-ensemble de l'ensemble des mots écrits sur l'alphabet binaire 0, 1.

Définir inductivement la fonction :
$val : EcrituresBin \to N$
où $val(u)$ est l'entier représenté par u.
Définir inductivement la fonction :
$EB : N \to EcrituresBin$
où $EB(n)$ est l'écriture binaire de n.

Un schéma inductif libre possible pour $EcrituresBin$ est

- Base : $\{"0","1"\}$
- Constructeurs $\Omega = \{\omega_0, \omega_1\}$ où pour $m \neq "0"$ $\omega_0(m) = m."0"$ et pour $m \neq 0$ $\omega_1(m) = m."1"$.

On peut alors définir la fonction val par

- val("0")=0 et val("1")=1
- Si $m \neq "0"$ val(m."0")=2*val(m) et val(m."1")=2*val(m)+1.

En utilisant pour N le schéma libre suivant

- Base $\{0, 1\}$
- Constructeurs $\Omega = \{\omega_p, \omega_i\}$ qui ne s'appliquent qu'à des n non nuls et tels que $\omega_p(n) = 2 * n$ et $\omega_i(n) = 2 * n + 1$

on peut définir EB par

- EB(0) = "0" et EB(1) = "1"
- Si n > 0, EB(2n) = EB(n)."0"
- Si n > 0, EB(2n+1) = EB(n)."1"

---

6 Ajouter un

Définir inductivement la fonction $AddUn$:
$AddUn$: $EcrituresBin \to EcrituresBin$, telle $val(AddUn(m)) = val(m) + 1$
sans utiliser les fonctions EB et val !!

En utilisant le même schéma inductif libre de définition pour $EcrituresBin$ qu'à l'exercice précédent on peut définir la fonction $AddUn$ par

- AddUn("0") = "1" et AddUn("1") = "1"."0"
- Si $m \neq "0"$, $AddUn(m."0") = m."1"$
- Si $m \neq "0"$, $AddUn(m."1") = AddUn(m)."0"$

---

7 Addition sur les mots

Définir inductivement la fonction
S : $EcrituresBin$ x $EcrituresBin \to EcrituresBin$,
où $S(m, n)$ est l'écriture binaire de l'entier $val(m) + val(n)$
Ne pas utiliser les fonctions EB et val !!

Pour définir la fonction S , on a besoin d'une définition inductive libre de $EcrituresBinxEcrituresBin$
Une définition possible :

- Base $\{("0","0"), ("0","1"), ("1","0"), ("1","1")\}$
- Constructeurs $\Omega = \{\omega_g^b, b \in \{"0","1"\}\} \cup \{\omega_d^b, b \in \{"0","1"\}\} \cup \{\omega^{a,b}, a, b \in \{"0","1"\}\}$ avec
  - Pour tout $a \in \{"0","1"\}$, pour tout $m' \neq "0"$, $n \in EcrituresBin$, $\omega_g^b(a, n) = (a, n.b)$
  - Pour tout $b \in \{"0","1"\}$, pour tout $m' \neq "0"$, $m \in EcrituresBin$, $\omega_g^a(m, b) = (m.a, b)$
  - Pour tout a et $b \in \{"0","1"\}$, $m \neq "0"$,$n \neq "0"$ $\omega^{a,b}(m, n) = (m.a, n.b)$

On vérifie que ce schéma définit bien des couples d'écritures binaires (aucun des mots produits n'a de zéro inutile en tête). Tous les couples d'écritures binaires sont produit par ce schéma. Il est libre, on peut donc l'utiliser pour définir la fonction S:

- $S("0","0") = "0"$, $S("0","1") = "1"$, $("1","0") = "1"$, $S("1","1") = "10"$
- Si $m \neq$ "0",

- $S(m.\text{"}0\text{"}, \text{"}0\text{"}) = m.\text{"}0\text{"}$
- $S(m.\text{"}0\text{"}, \text{"}1\text{"}) = m.\text{"}1\text{"}$
- $S(m.\text{"}1\text{"}, \text{"}0\text{"}) = m.\text{"}1\text{"}$
- $S(m.\text{"}1\text{"}, \text{"}1\text{"}) = S(m, \text{"}1\text{"}).\text{"}0\text{"}$

- Si $n \neq \text{"}0\text{"}$,
  - $S(\text{"}0\text{"}, n.\text{"}0\text{"}) = n.\text{"}0\text{"}$
  - $S(\text{"}0\text{"}, n.\text{"}1\text{"}) = n.\text{"}1\text{"}$
  - $S(\text{"}1\text{"}, n.\text{"}0\text{"}) = n.\text{"}1\text{"}$
  - $S(\text{"}1\text{"}, n.\text{"}1\text{"}) = S(n, \text{"}1\text{"}).\text{"}0\text{"}$

- Si $m \neq \text{"}0\text{"}$ et $n \neq \text{"}0\text{"}$,
  - $S(m.\text{"}0\text{"}, n.\text{"}0\text{"}) = S(m, n).\text{"}0\text{"}$
  - $S(m.\text{"}0\text{"}, n.\text{"}1\text{"}) = S(m, n).\text{"}1\text{"}$
  - $S(m.''1, n.\text{"}0\text{"}) = S(m, n).\text{"}1\text{"}$
  - $S(m.\text{"}1\text{"}, n.\text{"}1\text{"}) = AddUn(S(m, n)).\text{"}0\text{"}$

Remarque, on se simplifierait grandement la vie en utilisant le mot vide . On pourrait s'appuyer sur la définition libre suivante de $\{0,1\}^*$ x $\{0,1\}^*$

- Base $\{(\epsilon, \epsilon)\}$
- Constructeurs $\Omega = \{\omega_g^b, b \in \{\text{"}0\text{"}, \text{"}1\text{"}\}\} \cup \{\omega_d^b, b \in \{\text{"}0\text{"}, \text{"}1\text{"}\}\} \cup \{\omega^{a,b}, a, b \in \{\text{"}0\text{"}, \text{"}1\text{"}\}\}$ avec
  - $\forall b \in \{\text{"}0\text{"}, \text{"}1\text{"}\}, \omega_g^b(\epsilon, n) = (\epsilon, nb)$
  - $\forall a \in \{\text{"}0\text{"}, \text{"}1\text{"}\},\ \omega_g^a(m, \epsilon) = (ma, \epsilon)$
  - $\forall a, b\ \in \{\text{"}0\text{"}, \text{"}1\text{"}\},\ \omega^{a,b}(mn) = (ma, nb)$

On pourrait alors définir une fonction Σ de $\{0,1\}^*$ x $\{0,1\}^*$ dans $\{0,1\}^*$ par

- $\Sigma(\epsilon, \epsilon) = \epsilon,$
- $\Sigma(ma, \epsilon) = \Sigma(m, \epsilon).a$
- $\Sigma(\epsilon, nb) = \Sigma(\epsilon, n).b$
- $\Sigma(m.\text{"}0\text{"}, n.\text{"}0\text{"}) = \Sigma(m, n).\text{"}0\text{"}$
- $\Sigma(m.\text{"}0\text{"}, n.\text{"}1\text{"}) = \Sigma(m, n).\text{"}1\text{"}$
- $\Sigma(m.\text{"}1\text{"}, n.\text{"}0\text{"}) = \Sigma(m, n).\text{"}1\text{"}$
- $\Sigma(m.\text{"}1\text{"}, n.\text{"}1\text{"}) = PlusUn(\Sigma(m, n)).\text{"}0\text{"}$

ou encore

- $\Sigma(\epsilon, m) = m$
- $\Sigma(m, \epsilon) = m$
- $\Sigma(m.\text{"}0\text{"}, n.\text{"}0\text{"}) = \Sigma(m, n).\text{"}0\text{"}$
- $\Sigma(m.\text{"}0\text{"}, n.\text{"}1\text{"}) = \Sigma(m, n).\text{"}1\text{"}$
- $\Sigma(m.\text{"}1\text{"}, n.\text{"}0\text{"}) = \Sigma(m, n).\text{"}1\text{"}$
- $\Sigma(m.\text{"}1\text{"}, n.\text{"}1\text{"}) = PlusUn(\Sigma(m, n))).\text{"}0\text{"}$

et constater que la restriction de Σ à $EcrituresBin$x$EcrituresBin$ renvoit bien un $EcrituresBin$

---

8 Liste

Définir inductivement les fonctions

- longueur d'une liste
- concaténation de deux listes
- ajout_en_fin d'un élément à une liste
- miroir d'une liste
- appartenance d'un élément à une liste

Donner aussi un code récursif.

---

longueur(liste_vide)=0
longueur(ajoute(a,l))=longueur(l)+1
concat(liste_vide,l)=l
concat(ajoute(a,l'),l)=ajoute(a,concat(l',l))
ajout_en_fin(liste_vide,e)= ajoute(e, liste_vide)
ajout_en_fin(ajoute(a,l),e)=ajoute(a,ajout_en_fin(l,e)
miroir(liste_vide)=liste_vide
miroir(ajoute(a,l))=ajout_en_fin(miroir(l),a))

---

9 Exercice

On suppose que les éléments de la liste sont des entiers.
Ecrire une fonction inductive qui a une liste associe

- la somme de ses éléments
- la sous liste de ses éléments pairs
- la liste où tous les éléments ont été augmentés de un

---

somme(liste_vide)=0
somme(ajoute(a,l))=somme(l)+a
paire(liste_vide)=liste_vide
paire(ajoute(a,l))= paire (l) si a est impair et ajoute(a, paire(l)) si a est pair
add_un(liste_vide)=liste_vide
add_un(ajoute(a,l))=ajoute (a+1, add_un(l))
