Preuve de programmes

Notions de Logique

E. CHABBAR

Logique

- La logique a joué un rôle décisif dans le développement de l'informatique, notamment en informatique théorique:

  - Définition d'un modèle théorique des premiers ordinateurs (Machine de Turing)

  - Calcul booléen pour la conception et l'étude des circuits.

  - La récursivité pour définir la calculabilité.

  - la décidabilité et la complexité pour étudier la limite de la machine.

  - La programmation fonctionnelle.

Logique

- Actuellement l'informatique a envahi tous les domaines de la vie. Le problème le plus important qui se pose dans la conception d'une application informatique est de prouver qu'elle est exempte d'erreurs et qu'elle résout le problème pour laquelle elle a été conçue. Pour cela, on définit une tache par une formule de logique et on montre qu'elle est satisfaite par un modèle de cette application. Ce type de preuve est appelé « Vérification Formelle ».

Logique

- **Une logique, par définition, est un ensemble de formules.**
- Une formule est construite, sur un alphabet, suivant certaines règles (Syntaxe).
- La sémantique d'une formule est une valuation (ou interprétation dans un modèle) qui détermine la valeur de vérité de la formule.
- Il y a plusieurs types de logiques:
  - Logique des propositions (d'ordre 0)
  - Logique du premier ordre (les prédicats en font partie)
  - logique du second ordre et logique d'ordre supérieur.

Logique des propositions

- - On note P = {p, q, …} l'ensemble des propositions atomiques. Chaque proposition atomique est une variable qui ne peut prendre que « vrai » ou « faux ».

  - {¬ , ∧, ∨, ⇒, ⇔} l'ensemble des connecteurs (ou opérateurs) logique plus les parenthèses.
- - On note F(P) l'ensemble des formules déduit de P.
- **F(P) est le plus petit ensemble qui contient P et qui est stable par les connecteurs.**

  En d'autres termes:

  Une formule est une suite de symboles de P ∪ {¬ , ∧, ∨, ⇒, ⇔, (,)} construite selon les règles suivantes:

Logique des propositions

- Toute proposition atomique est une formule.
- si f est une formule alors $\neg$f est aussi une formule.
- si f1 et f2 sont des formules alors :
  - f1 $\wedge$ f2 est une formule
  - f1 $\vee$ f2 « « «
  - f1 $\Rightarrow$ f2
  - f1 $\Leftrightarrow$ f2 « « «

Exemple: $(\neg p \Rightarrow q) \vee (p \wedge q)$ est une formule.

Logique des propositions

- La sémantique des opérateurs logiques est donnée par des tables de vérité.

p	q	¬ p	¬ q	p ∨ q	p ∧ q	p⇒q (¬p∨q)
0	0	1	1	0	0	1
0	1	1	0	1	0	1
1	0	0	1	1	0	0
1	1	0	0	1	1	1

Prédicats

- Les prédicats sont construits avec :
  - les constantes (0,1,2,….)
  - les variables (x, y, ….)
  - les fonctions (f, g, +, *, …)
  - des connecteurs logiques ($\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$)
  - des parenthèses
  - des quantificateurs ($\forall$, $\exists$)
- Un prédicat atomique est un prédicat qui ne contient ni connecteur ni quantificateur.
  - Exemples: x < y ; pair(2 x) ;

Prédicats

- Règles de formations des formules de prédicats:

  - Tout prédicat atomique est une formule

  - si f1 et f2 sont des formules alors

    $\neg$ f1, f1 $\vee$ f2, f1 $\wedge$ f2, f1 $\Rightarrow$ f2, f1 $\Leftrightarrow$ f2

    sont des formules.

  - si f est une formule alors

    $\forall$ x f est une formule

    $\exists$ x f est une formule

Prédicats

- **Exemples:**

  1) $\forall x$ pair(x + x)

  2) $\exists x$ premier(x) $\wedge$ x < succ(succ(0))

  3) $\forall x$ (oiseau(x) $\Rightarrow$ vole(x)) (tous les oiseaux volent)

  4) $\exists x$ (oiseau(x) $\wedge \neg$ vole(x)) (4 $\equiv \neg$ 3)

  5) $\forall i \forall j$ ($1 \leq i \leq j \leq n \Rightarrow T[i] \leq T[j]$) (spécification d'un tableau trié)

- **Remarques**

  $\neg (\forall x\ P) \equiv \exists x \neg P$

  $\neg (\exists x\ P) \equiv \forall x \neg P$

Déduction logique

- Démonstrations logiques

- Un séquent est un couple de la forme ($\mathcal{J}$,**f**), où **f** est une formule et $\mathcal{J}$ un ensemble fini de formules . L'ensemble $\mathcal{J}$ est l'ensemble des prémisses du séquent, la formule **f** sa conclusion.

- Séquents prouvables
- Un séquent ($\mathcal{J}$,**f**) est prouvable, ce que l'on notera $\mathcal{J} \vdash \mathbf{f}$, s'il peut être construit en utilisant un nombre fini de fois les 6 règles suivantes :

1. si $\mathbf{f} \in \mathcal{J}$, alors $\mathcal{J} \vdash \mathbf{f}$ (Hypothèse)
2. si $\mathbf{g} \notin \mathcal{J}$ et $\mathcal{J} \vdash \mathbf{f}$, alors $\mathcal{J}, \mathbf{g} \vdash \mathbf{f}$
3. si $\mathcal{J} \vdash (\mathbf{f} \Rightarrow \mathbf{f'})$ et $\mathcal{J} \vdash \mathbf{f}$, alors $\mathcal{J} \vdash \mathbf{f'}$ (Modus Ponens)
4. si $\mathcal{J}, \mathbf{f} \vdash \mathbf{f'}$, alors $\mathcal{J} \vdash (\mathbf{f} \Rightarrow \mathbf{f'})$ (Synthèse)
5. $\mathcal{J} \vdash \mathbf{f}$ ssi $\mathcal{J} \vdash \neg\neg\mathbf{f}$
6. si $\mathcal{J}, \mathbf{f} \vdash \mathbf{f'}$ $\mathcal{J}, \mathbf{f} \vdash \neg\mathbf{f'}$ alors $\mathcal{J} \vdash \neg\mathbf{f}$ (Raisonnement par l'absurd (Contradiction)

- Démonstrations logiques
- Une démonstration d'un séquent prouvable $\mathcal{J} \vdash \mathbf{f}$ est une suite finie de séquents prouvables $\mathcal{J}i \vdash \mathbf{f}i$,

  i = 1,...,n, telle que :
- $\mathcal{J}n = \mathcal{J}$ et $\mathbf{f}n = \mathbf{f}$
- chaque séquent de la suite est obtenu à partir des séquents qui le précède en appliquant l'une des 6 règle
- Remarque : le premier séquent de la suite est nécessairement obtenu par utilisation d'une hypothèse

- Démonstrations logiques - Exemple
- Tous les hommes sont mortels, et
- les Grecs sont des hommes, donc les Grecs sont mortels

h = « être un homme »

m = « être mortel »

g = « être Grec »

$(h \Rightarrow m), (g \Rightarrow h) \vdash (g \Rightarrow m)$

- Démonstrations Logiques - Exemple
- Tous les hommes sont mortels, et
- les Grecs sont des hommes, donc les Grecs sont mortels

1.	$(h \Rightarrow m), (g \Rightarrow h), g \vdash g$	hypothès
2.	$(h \Rightarrow m), (g \Rightarrow h), g \vdash (g \Rightarrow h)$	hypothès
3.	$(h \Rightarrow m), (g \Rightarrow h), g \vdash h$	MP (1&2)
4.	$(h \Rightarrow m), (g \Rightarrow h), g \vdash (h \Rightarrow m)$	hypothès
5.	$(h \Rightarrow m), (g \Rightarrow h), g \vdash m$	MP (3&4)
6.	$(h \Rightarrow m), (g \Rightarrow h) \vdash (g \Rightarrow m)$	synthèse

16

Preuve de programmes

Logique de Hoare

Correction de programmes / Spécification

- un programme est correct s'il effectue la tâche qui lui est confiée dans tous les cas permis possibles

- nécessité de disposer d'un langage de spécification permettant de décrire **formellement** la tâche confiée à un programme.

Correction de programmes / Spécification

- description des propriétés que doit satisfaire un programme pour répondre au problème posé

  - relation entre les entrées et les sorties du programme

Correction de programmes / Spécification

- Exemple

une spécification pour le problème du calcul de la racine carrée entière par défaut :

Données n : **entier ;**
Résultat r : **entier ;**

Pré condition : $n \geq 0$

Algorithme: **r := 0; tantque $(n \geq (r+1)^2)$ faire r := r + 1 ftantque**

Post condition : $(r^2 \leq n) \wedge (n < (r+1)^2)$

Test vs Vérification

- Les couples (n, r) de l'ensemble suivant {(0, 0), (1, 1), (2,), (3, 1), (4, 2), …} sont des tests qui réussissent

• Hoare propose une exécution de l'algorithme sur une valeur symbolique qui est un ensemble de valeurs défini par une expression logique (ou prédicat).

• L'ensemble {(0, 0), (1, 1), (2, 1), (3, 1), (4, 2), …} est défini par {(n,r) / $\mathbf{n \geq 0 \wedge r \geq 0 \wedge r^2 \leq n \wedge (n < (r+1)^2}$}

• On note $\mathbf{\{n \geq 0 \wedge r \geq 0 \wedge r^2 \leq n \wedge (n < (r+1)^2\}}$ la valeur symbolique des variables n et r.

• L'exécution d'un algorithme A sur une valeur symbolique de données définies par l'expression { p} qui donne une valeur symbolique résultat définie par l'expression { q} est notée {p} A {q} . **{p} A {q} est appelé triplet de Hoare (p : précondition, q : postcondition).**

Système formel

- **Interpretation d'un triplet de Hoare**

  {p} A {q} signifie :
  Si la propriété p est vraie avant l'exécution de A ET si **l'exécution** de A **se termine**, ALORS la propriété q est vraie après l'exécution de A.
  (Correction partielle)

• Un **système formel** est un triplet <L, Ax, R> où :
  - L est un langage définissant un ensemble de formules,
  - Ax est un sous-ensemble de L ; chaque formule de Ax est appelée axiome,
R est un ensemble de règles de déduction de formules à partir d'autres formules

Langage algorithmique

- Instructions :
  - **Affectation** (:= symbole d'affectation et = pour la comparaison)
  - **Contrôle** :

    Si <cond> **alors** instruction **fsi**

    ou **Si** <cond> **alors** instruction

    sinon instruction **fsi**
  - **Boucle** :

    Tantque <cond> **faire** instruction **ftantque**

- Restriction :
  - Pas de fonction (ou procédure)
  - ni de variable pointeur
  - pas de désignation de la forme t[i] où i fait référence à un autre tableau.

Logique de Hoare

• Définition de la logique de Hoare : La logique de HOARE est un triplet <L, Ax, R> avec :

- L est l'ensemble des formules { p} A { q} où p et q sont des prédicats et A est un fragment d'algorithme(ou de programme)

- Ax est l'ensemble des axiomes de la forme suivante : { p[x / e] } x := e { p(x)}

(p(x) est obtenu de p[x/e] en substituant toute occurrence de e par x)

- R est l'ensemble de règles de déduction suivant :

Logique de hoare

- (Séq) si $\{p\}\ A1\ \{r\}$; $\{r\}\ A2\ \{q\}$ alors
  $\{p\}\ A1 ; A2\ \{q\}$
- ($cons_g$) si $p \Rightarrow p'$, $\{p'\}\ A\ \{q\}$ alors
  $\{p\}\ A\ \{q\}$
- ($cons_d$) si $\{p\}\ A\ \{p'\}$, $p' \Rightarrow q$ alors
  $\{p\}\ A\ \{q\}$
- ($cond_1$) si $\{p \wedge c\}\ A\ \{q\}$, $(p \wedge \neg c) \Rightarrow q$ alors
  $\{p\}$ si c alors A fsi $\{q\}$
- ($cond_2$) si $\{p \wedge c\}\ A1\ \{q\}$, $\{p \wedge \neg c\}\ A2\ \{q\}$ alors
  $\{p\}$ si c alors A1 sinon A2 fsi $\{q\}$
- (tantque) si $\{I \wedge C\}\ A\ \{I\}$ alors
  $\{I\}$ tantque C faire A ftantque $\{I \wedge \neg C\}$

(I doit être un invariant de la boucle)

Exemple : règles d'affectation et conséquence

- { p[x / e] } x := e { p(x)}

1. {n+1> 0} n:= n+1 {n>0} (x=n, e = n+1)
2. {k > 0} n :=0 {k > n} (x=n, e =0)
3. {x = 4} x := x+1 {x=5} en écrivant:

   {(x+1) -1 =4} x := x+1 { x - 1 =4} $\Rightarrow$ {x = 5}

4. {x $\geq$ 0} x := x + 1 {x $\geq$ 1}

   {x $\geq$ 0} $\Rightarrow$ {x + 1 $\geq$ 1}

   x := x + 1

   {x $\geq$ 1}

Exemple : règles de condition

5. {vrai}

si $x \geq 0$ alors

$\{vrai \wedge x \geq 0\} \Rightarrow \{x \geq 0\} \Rightarrow \{x+1 \geq 1\}$

$x := x + 1;$

$\{x \geq 1\}$

sinon

$\{vrai \wedge x < 0\} \Rightarrow \{x < 0\} \Rightarrow \{-x > 0\}$

$x := - x;$

$\{x > 0\} \Rightarrow$ **$\{x \geq 1\}$**

fsi

$\{x \geq 1\}$

Schéma de preuve d'un algorithme itératif Tant que

- Soit l'algorithme suivant:

début
{pré}
init;
{I}
tantque C faire
{I ∧ C}
A
{I}
ftantque
{I ∧ ¬ C} ⇒
{post}

Exemple: calcul de la racine carrée par défaut

- Algorithme A:

donnée : n : entier;
résultat : r : entier;
début
r := 0;
tantque $n \geq (r + 1)^2$ **faire**
r := r + 1;
ftantque
fin

- **Spécification:**

précondition : $\{n \geq 0\}$
postcondition : $\{ (r^2 \leq n \wedge n < (r + 1)^2) \}$

- **invariant de la boucle:** $I = \{r^2 \leq n\}$ (évident, sinon on le déduit de la postcondition)

Algorithme A annoté et prouvé

Algorithme A:

donnée : n : entier;

résultat : r : entier;

Début

$\{n \geq 0\} \Rightarrow \{n \geq 0*0\}$

r := 0;

$\{n \geq r*r\}$

tantque $n \geq (r + 1)^2$ **faire**

$\{n \geq r^2 \wedge n \geq (r + 1)^2\}$

r := r + 1;

$\{n \geq r^2\}$

ftantque

$\{r^2 \leq n \wedge n < (r + 1)^2\}$

fin

Exemple: calcul de n!

- **Algorithme B:**

donnée : n : entier
résultat : y : entier
début
x := n;
y := 1;
tantque x > 1 **faire**
y := y * x;
x := x – 1;
ftantque
fin

- **Spécification:**
  précondition : {n ≥ 0}
  postcondition : {y = n!}
- **Invariant:** I = { n! = y x! ∧ x ≥ 0}

Algorithme B annoté et prouvé

Algorithme B:

donnée: n : entier;

résultat: y : entier;

début

{$n \geq 0$}

x := n;

{$x \geq 0 \wedge x = n$}

y := 1;

{$n! = y\ x! \wedge x \geq 0$}

tantque x > 1 **faire**

{$n! = y\ x! \wedge x \geq 0 \wedge x > 1$} $\Rightarrow$ {$n! = (y\ x)(x-1)! \wedge x > 1$}

y := y * x;

{$n! = y\ (x-1)! \wedge x-1 > 0$}

x := x – 1;

{$n! = y\ x! \wedge x > 0$} $\Rightarrow$ {$n! = y\ x! \wedge x \geq 0$}

ftantque

{$n! = y\ x! \wedge x \geq 0 \wedge x \leq 1$} $\Rightarrow$ { $y = n!$}

Exercice: Faites la preuve de l'algo. de n! en faisant une boucle qui va de 1 à n