Méthodes formelles

Daniel Sanz
Université de Fribourg,
daniel.sanz@unifr.ch

March 18, 2020

Abstract

Ceci est un résumé non officiel du cours de méthodes formelles du professeur Ultes Nietzche. Il s'agit principalement de ces slides traduites en français ainsi que quelques exos en guise d'exemple.

Introduction

Les formules logiques, les prédicats entre autres peuvent être utilisés afin d'éxprimer de l'information sur l'état d'un programme. $x = 10$; indique que x doit impérativement avoir la valeur 10.

Pré & post-condition

Définition Une précondition P nous indique ce qui peut être considéré comme vrai avant même l'execution d'une séquence d'instructions S.
Une postcondition Q nous indique ce qui sera vrai après l'execution des instructions S.

Notation On écrit : $\{P\}\ S\ \{Q\}$ qui veut dire que si P est vrai alors, après l'execution de S, Q est vrai. Il s'agit d'un triplet d'Hoarce.

Exemple

$$\{x = 2;\}\ x = x \cdot 3;\ \{x = 6;\}$$

On utilise $\hat{x}$ comme notation de la variable x pour indiquer la valeur de x après l'execution du programme et x avant l'execution.

Exemple

$$\{true\}\ x = x + 1;\ \{\hat{x} > x\}$$
$$\{true\}\ x = x + 1;\ \{\hat{x} = x + 1\}$$

Les assertions

Il est possible d'écrire des prédicats entre deux lignes de code. On présume alors que ce prédicats est la postcondition de la ligne de code précédente et qu'il est la précondition de la ligne suivante.

Définition De tels prédicats sont dits *assertions* ou *annotations*. Pour savoir si des triplets sont corrects il faut tout d'abord transformer le programme S en une formule ϕ_S. Ainsi il est possible de prouver l'exactitude d'un triplet:

$$\{P\}\ S\ \{Q\}$$

en verifiant la formule:

$$P \wedge \phi_S \rightarrow Q$$

ou de façon analogue:

$$\phi_S \rightarrow (P \rightarrow Q)$$

Exemple

$$\{true\}\ x = 10;\ \{x > 0\}$$
$$\phi_S \equiv x = 10$$

donc, $(true \wedge (x = 10)) \rightarrow (x > 10)$.

Exemple

$$\{x \neq 0\}\ x = 1/x;$$
$$x = 1/x;\ \{\hat{x} = x\}$$

Soit $x'' = 1/x$ et $\hat{x} = 1/x''$ alors on vérifie:

$$(x \neq 0) \wedge (x'' = 1/x) \wedge (\hat{x} = 1/x'') \rightarrow \hat{x} = x$$

Ce qui est vrai par du calcul élémentaire. Biensûr, ici on ne tient pas compte de la précision limitée des *floats*.

Les clauses If

Soit la clause générale *If* suivante. Avec la précondition P et la postcondition Q.

$$\{P\}\ \text{if}(\mathit{condition})\ \{\text{progI}\}$$
$$\text{else}\ \{\text{progII}\}$$
$$\{Q\}$$

Afin de prouver cette clause il faut procéder aux transformation suivantes (et les prouver indépendament):

$$\{P \wedge \mathit{condition}\}\ \text{progI}\ \{Q\} \qquad \underline{\text{et}}$$
$$\{P \wedge \neg \mathit{condition}\}\ \text{progII}\ \{Q\}$$

Les deux triplets doivent être vrais pour vérifier la clause.

Exemple Soit la clause *If* suivant avec sa précondition P et postcondition Q respéctive.

$$\overbrace{\{\mathit{true}\}}^{\text{P}}\ \text{if}(x < 0)\ \{x = -x\}$$
$$\text{else}\ \{x = x\}$$
$$\underbrace{\{\hat{x} \geqslant 0\}}_{\text{Q}}$$

Ainsi on fait la transformation :

$$\{true \wedge (x < 0)\}\ x = -x\ \{\hat{x} \geqslant 0\} \quad \underline{\text{et}}$$
$$\{true \wedge \neg(x < 0)\}\ x = x\ \{\hat{x} \geqslant 0\}$$

Formellement, il faudrait encore faire las transformation du programme en une formule logique ϕ_S afin de prouver les deux triplets. Mais il est évident que c'est juste.

Les boucles

On s'intérresse ici à comment prouver les boucles. En particulier la boucle while. Mais tout d'abord nous avons besoin de quelques définitions et d'outils supplémentaires.

Définitions

- Quand une boucle se termine et que le résultat éspéré est atteint on dit qu'il s'agit d'une *exactitude partielle*.
- Quand il est garantit qu'une boucle se atteint une fin on dit : *termination*
- Quand les deux conditions précédentes sont vérifiées on parle d'*exactitude total*

Invariant de boucle

Il s'agit d'une formule logique qui est vraie dans les cas suivants:

- avant la boucle
- avant chaque execution du corps de la boucle
- après chaque execution du corps de la boucle
- après la boucle

et elle doit rendre la postcondition vraie. Donc grosso merdo c'est vrai tout le temps, d'où l'invariance.

Variant de boucle

Il s'agit d'une expression évaluée dans les entiers $\mathbb{N}^+$ qui

- est décrémentée de 1 à chaque itération
- ne peut pas aller en dessous de 0

Donc en d'autres termes int i = CST; et dans la boucle i = i - 1;

Boucle While

Soit la boucle suivante:

$$\{P\}\ initialisation;$$
$$\text{while}(condition)\,\{\text{loop body}\}\,;\ \{Q\}$$

On vérifie l'exactitude partielle et la terminaison séparement.

L'exactitude partielle à l'aide de l'invariant de boucle nous donne les formules suivantes:

$$\begin{aligned} \{P\}\ initialisation; &\qquad \{Inv\} \\ \{Inv \wedge condition\}\ \text{loop body}; &\qquad \{Inv\} \\ \{Inv \wedge \neg condition\}\ \text{skip}; &\qquad \{Q\} \end{aligned}$$

La terminaison nous donne encore la formule suivante à vérifier:

$$\{intvar \wedge var > 0\}\ \text{loop body};\ \{var > var' \geqslant 0\}$$

Exemple Soit ce programme qui calcul de façon itérative la somme des entiers de 1 à n.

$$\begin{aligned} &\overbrace{\{n > 0 \wedge x = 1\}}^{P}\ sum = 1; \text{//initialisation} \\ &\quad \text{while}(x < n)\{ \\ &\qquad\qquad x = x + 1; \\ &\qquad\qquad sum = sum + x; \\ &\qquad\qquad \}; \qquad\qquad \underbrace{\{sum = n(n+1)/2\}}_{Q} \end{aligned}$$