327 : Exemples de preuves d'algorithmes : correction & terminaison ! 09/11 2014

DVT importants : (complétude relative de Hoare) correction de Hoare.

L'objectif de ce cours est de présenter quelques méthodes de preuves d'algorithmes.
On dit qu'un algo. <u>termine</u> s'il s'exécute en un temps fini et qu'il est correct si il réalise sa spécification.

I Preuves informelles [Cormen]

1. Algorithmes impératifs : correction

<u>définition 1 :</u> Invariant de boucle
Un invariant de boucle est un prédicat vrai à chaque passe de la boucle.

La méthode de l'invariant de boucle permet de valider le fonctionnement d'une boucle.
Dans le cas de boucles imbriquées, on adopte une approche bottom-left.

<u>exemples :</u> • l'exponentiation rapide

```
pow (a, n) =
  p <- 1
  while n > 0
     si n impair
        p <- p x a
        n <- n - 1
     sinon
        n <- [n/2]
        p <- p * p
  renvoyer p
```

avec $n = \overline{n_\ell \dots n_0}^2$,
à l'étape $k \in [\![0, \ell]\!]$, $p = a^{\overline{n_k \dots n_0}}$
à la fin $p = a^n$.

• l'algorithme d'Euclide renvoit le pgcd de deux entiers.

2. Correction d'algorithmes récursifs

<u>définition 2 :</u> Relation bien fondée
Une relation binaire $\prec$ sur un ensemble E est dite bien fondée s'il n'existe pas de suite infinie décroissante.

<u>exemple :</u> $\mathbb{N}$ muni de $<$ est bien fondé.

<u>Théorème 1 :</u> Induction sur un ensemble bien fondé
Soit $(E, <)$, un ensemble bien fondé et soit p un prédicat.
~~Si p est vérifié pour les éléments minimaux de E~~
et si $\forall x \in E, \left[\forall y \in E, y < x, p(y)\right] \Rightarrow p(x)$
alors, $\forall x \in E, p(x)$

<u>remarque :</u> sur $\mathbb{N}$, on retrouve le théorème de récurrence.

<u>Théorème 2 :</u> Théorème de correction
Soit $f : A \to B$ récursive et soit $\phi : A \to E$ où $(E, <)$ est un ensemble bien fondé.
Soit $M = \{x \in A, \phi(x) \text{ est minimal dans } \phi(A)\}$
Soit P_f, un prédicat sur les valeurs calculées par f.
si $\begin{cases} \forall b \in M, P_f(b) \\ \forall x \in A, \left(\forall y \in A, \phi(y) < \phi(x), P_f(y)\right) \Rightarrow P_f(x) \end{cases}$
alors P_f est vérifié pour tout calcul de f.

<u>exemple :</u> Le calcul du binôme de Newton via le triangle de Pascal est correct :
$A = \mathbb{N}^2$, $E = \mathbb{N}$, $\phi : A \to E$
$(n, p) \mapsto 0$ si $p > n$, p sinon

DVP : Knuth-Morris-Pratt / lièvre & tortue / Dijkstra / Bellman Ford / factorielle (logique de Hoare) / division euclidienne

3. Terminaison des algorithmes impératifs

Définition 3 : Variant de boucle
Un variant de boucle est une quantitée $V \in E$ où $(E, <)$ est un ensemble bien fondé.

Théorème 3 : Théorème de terminaison
Une boucle qui possède un variant décroissant termine.

exemples : l'exponentiation rapide & l'algo. d'Euclide terminent.

4. Une preuve complète : lièvre et tortue [Winskel]

Développement 01 :

Soit E, un ensemble fini et soit $f : E \to E$
La suite $u_{n+1} = f(u_n)$ est periodique à partir d'un certain rang : r
L'algorithme du lièvre et de la tortue détermine le rang r et la periode T en temps $O(r+T) = O(E)$.

5. Terminaison : le cas récursif

Théorème 4 : Théorème de terminaison
Soit f, A, B, φ, E et M comme dans le théorème 2, si $\forall b \in M$, $f(b)$ termine et si $\forall x \in A$, la définition de $f(x)$ ne fait apparaitre que des appels (en nombre fini) de $f(y)$ où $\varphi(y) < \varphi(x)$, alors $f(x)$ termine $\forall x \in A$.

6. Terminaison : cas général

Certains algorithmes ne terminent pas :
exemple :

Morris $(m, n) =$
si $m = 0$, renvoyer 1
sinon Morris $(m-1, \text{Morris}(m, n))$

Le problème de terminaison est indécidable dans le cas général.

demo :

Si il existe Termine : $X \mapsto$ True si X termine
False sinon
alors pour TEST : $() \mapsto$ TEST si Termine(TEST) = True
$\mapsto$ True sinon
il y a contradiction.

exemple d'algorithme dont on ne sait pas si il termine :

COLLATZ (n) : Si $n = 1$, renvoyer 1
Sinon
si n pair, COLLATZ $(n/2)$
sinon COLLATZ $(3n+1)$

II. Preuves Formelles et logique de Hoare

1. Assertions et sémantique dénotationnelle

L'objectif est de définir formellement la notion de propriété.

définition 1 : Assertion

Cette notion est inductive :

- Toute expression booléenne est une assertion
- Si p et q sont des assertions, alors, $\neg p$, $p \vee q$, $p \wedge q$, $p \Rightarrow q$, $p \Leftrightarrow q$ aussi
- Si p : assertion et x : variable, $\forall x : p$ et $\exists x : p$: assertions

définition 2 : état de programme ; sémantique

- On appelle état d'un programme une application σ qui associe à chaque variable une valeur.
- On note Σ l'ensemble des états.
- On appelle sémantique d'une assertion l'application

$$\models P : \Sigma \to \{V/F\}$$
$$\sigma \mapsto V \text{ si } p \text{ est vérifié pour } \sigma, \quad F \text{ sinon}$$

rem : on note parfois $S[\![P]\!](\sigma)$ au lieu de $\sigma \models P$.

2. Règles de Hoare & Preuves d'Algorithmes

Le principe de la logique de Hoare est d'introduire des règles permettant de décomposer un programme pour prouver sa correction.

On écrit alors le programme $\{Pré\}\ Prog\ \{Post\}$.

définition 3 : Corrections partielle et totale.

- $\{P\} S \{Q\}$ est partiellement correct si toute exécution de S démarrant dans un état P ~~et terminant~~ termine, alors elle termine dans un état Q.
- $\{P\} S \{Q\}$ est totalement correct si toute exécution de S démarrant en P termine en Q.

On définit les règles de Hoare :

$$\{A\}\ SKIP\ \{A\} \ ; \quad \{B[a/X]\}\ X := a\ \{B\} \ ; \quad \frac{\{A\} P_1 \{B\} \ ; \ \{B\} P_2 \{C\}}{\{A\} P_1 ; P_2 \{C\}}$$

$$\frac{\{A \wedge b\} P_1 \{B\} \ ; \ \{A \wedge \neg b\} P_2 \{B\}}{\{A\}\ \text{if } b \text{ then } P_1 \text{ else } P_2\ \{B\}} \ ; \quad \frac{\{A \wedge b\} c \{A\}}{\{A\}\ \text{while } b \text{ do } c\ \{A \wedge \neg b\}}$$

$$\frac{\models (A \Rightarrow A') \quad \{A'\} c \{B'\} \quad \models (B' \Rightarrow B)}{\{A\} c \{B\}}$$

; on écrit $\vdash \{A\} c \{B\}$ quand $\{A\} c \{B\}$ est un théorème.

Théorème : Cohérence de la logique de Hoare

Si $\vdash \{A\} c \{B\}$, alors $\models \{A\} c \{B\}$

développement 02

3. Plus faibles précondition : vers une complétude relative

définition 4 : Plus faible précondition

On appelle plus faible précondition du couple (c, B) l'ensemble
$$wp(c, B) = \{\sigma \in \Sigma,\ S[\![c]\!](\sigma) \models B\}.$$

Théorème : Complétude relative

$$\models \{A\} c \{B\} \Leftrightarrow A \subset wp(c, B)$$

Sémantique dénotationnelle

Correction des règles de Hoare

On omet dans les assertions les notations pour les interprétations, ce qui n'a aucune conséquence dans le déroulement de la preuve. Pour le reste, la preuve est celle de [1]. expressions arithmétiques avec des variables de prog. et de logique.

T

Lemme 1. Soient $a, a_0 \in$ Aexpv, et $X \in$ Var. Alors pour tout état σ,

$$\mathcal{A}v[\![a_0[a/X]]\!]\sigma = \mathcal{A}v[\![a_0]\!]\sigma[\mathcal{A}v[\![a]\!]\sigma/X]$$

$\mathcal{A}v[\![\cdot]\!]$: aexpv → état → valeur

Démonstration. Induction structurelle sur a_0. □

Aexp :
$a ::= n \mid X \mid a+a \mid a-a \mid a \times a$

Aexpv
$a ::= n \mid X \mid a+a \mid a-a \mid a \times a \mid i$

Lemme 2. Soit $B \in$ Assn (assertions), $X \in$ Var et $a \in$ Aexp (expression arithmétique). Alors pour tout état σ :

$$\sigma \models B[a/X] \text{ ssi } \sigma[\mathcal{A}[\![a]\!]\sigma/X] \models B.$$

Démonstration. Induction structurelle sur B. □

Théorème 3. Soit un triplet de Hoare $\{A\}c\{B\}$. Alors $\vdash \{A\}c\{B\}$ implique $\models \{A\}c\{B\}$.

On remarque que pour montrer le théorème, il suffit de prouver que chaque règle de Hoare est correcte, car on peut alors déduire le théorème par induction sur le nombre de règles dans une preuve.

Skip : $\models \{A\}\mathbf{skip}\{A\}$ est clair.

Affectation : Notons $c \equiv (\mathtt{X} := \mathtt{a})$. On a $\sigma \models B[a/X]$ ssi $\sigma[\mathcal{A}[\![a]\!]\sigma/X] \models B$ d'après le lemme , donc $\sigma \models B[a/X] \implies \mathcal{C}[\![\mathtt{X} := \mathtt{a}]\!]\sigma \models B$, d'où $\models \{B[a/X]\}X := a\{B\}$.

?

$\dfrac{\{A\}c\{C\} \quad \models \{C\}c_2\{B\}}{\models \{A\}c_0\,c_2\{B\}}$

Séquence : Supposons $\models \{A\}c\{C\}$ et $\models \{C\}c_1\{B\}$. Supposons $\sigma \models A$. Alors $\mathcal{C}[\![c_0]\!]\sigma \models C$ car $\models \{A\}c_0\{C\}$. On a aussi $\mathcal{C}[\![c_1]\!](\mathcal{C}[\![c_0]\!]\sigma) \models B$ car $\models \{C\}c_1\{B\}$. D'où $\models \{A\}c_0\ c_1\{B\}$.

Conditionnelle : Supposons $\models \{A \wedge b\}c_0\{B\}$ et $\models \{A \wedge \neg b\}c_1\{B\}$. Supposons $\sigma \models A$. Ou bien $\sigma \models b$ ou bien $\sigma \models \neg b$. Dans le premier cas $\sigma \models A \wedge b$ donc $\mathcal{C}[\![c_0]\!]\sigma \models B$. Dans le deuxième analoguement $\mathcal{C}[\![c_1]\!]\sigma \models B$ aussi, donc $\models \{A\}$ **if** b **then** c_0 **else** $c_1\{B\}$.

$\dfrac{\models \{A \wedge b\}c_1\{B\} \quad \models \{A \wedge \neg b\}c_2\{B\}}{\vdash \{A\} \text{ si } b \text{ alors } c_2 \text{ sinon } c_2 \{B\}}$

Boucles While : Supposons $\models \{A \wedge b\}c\{A\}$, i.e A invariant de $w \equiv$ while b do c. On sait que $\mathcal{C}[\![w]\!] = \lim_n \theta_n$ avec des θ_n de domaine croissant, et définis de sorte que :

$$\theta_{n+1} : \sigma \mapsto \begin{cases} \sigma & \text{si } \mathcal{B}[\![b]\!]\sigma = \text{ false} \\ (\theta_n \circ \mathcal{C}[\![c]\!])\sigma & \text{si } \mathcal{B}[\![b]\!]\sigma = \text{ true.} \end{cases}$$

Montrons par induction sur n, $P(n) : \forall \sigma, \sigma' \in \Sigma$,

$$(\theta_n(\sigma) = \sigma' \text{ et } \sigma \models A) \implies (\sigma' \models A \wedge \neg b)$$

On aura alors $\sigma \models A \implies \mathcal{C}[\![w]\!]\sigma \models A \wedge \neg b$ pour tout $\sigma \in \Sigma$, d'où $\models \{A\}w\{A \wedge \neg b\}$ comme voulu.

Cas $n = 0$. Il est très vrai. *Induction.* On suppose $P(n)$ pour un $n \geqslant 0$. Supposons $\theta_{n+1}\sigma = \sigma'$, et $\sigma \models A$. Deux cas sont possibles :

1. $\mathcal{B}[\![b]\!]\sigma = \texttt{true}$ et $\sigma' = (\theta_n \circ \mathcal{C}[\![c]\!])\sigma$;
2. $\mathcal{B}[\![b]\!]\sigma = \text{false}$ et $\sigma = \sigma'$.

Montrons que dans les deux cas $\sigma' \models A \wedge \neg b$.

1. On a $\sigma \models b$ donc $\sigma \models A \wedge b$. D'où il existe $\sigma'' \in \Sigma$ tel que $\sigma'' = \mathcal{C}[\![c]\!]\sigma$ et $\sigma' = \theta_n\sigma''$. D'où $\sigma'' \models A$ car $\models \{A \wedge b\}c\{A\}$ (hypothèse du while). Par hypothèse d'induction $\sigma' \models A \wedge \neg b$.
2. Ici $\sigma \models A \wedge \neg b$. Mais $\sigma = \sigma'$ donc c'est fini.

Conséquence : Supposons $\models (A \implies A')$ et $\models \{A'\}c\{B'\}$ et $\models (B' \implies B)$. Supposons $\sigma \models A$. Alors $\sigma \models A'$, d'où $\mathcal{C}[\![c]\!]\sigma \models B'$ et donc $\mathcal{C}[\![c]\!]\sigma \models B$. D'où $\models \{A\}c\{B\}$.

Références

[1] G.Winskel, *The Formal Semantics of Programming Languages.*

Algorithme du lièvre et de la tortue

Benjamin Arnaud, Maths 3, ENS Rennes

09/12/2014

On considère E un ensemble fini, f : E $\to$ E.

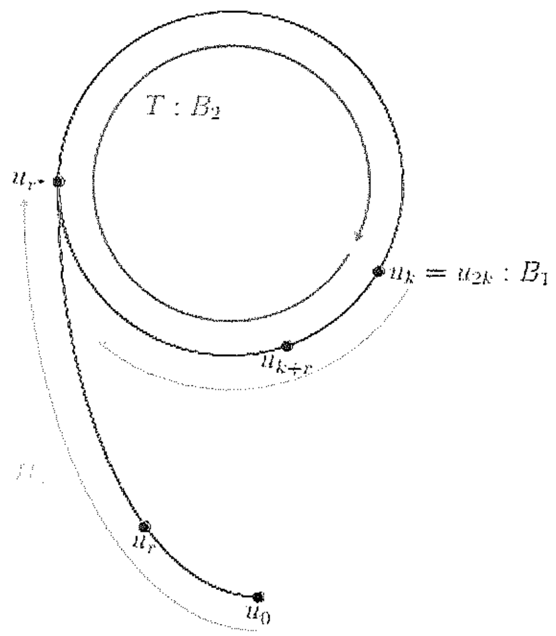
On conisdère u une suite définie de manière récursive par son premier terme u_0 et $\forall n \in \mathbb{N}, u_{n+1} = f(u_n)$.

Cette suite est périodique, de période T^* à partir d'un certain rang r^*, et l'algorithme suivant détermine la période et le rang de la suite en temps $O(T^* + r^*)$ et en espace constant.

1 Algorithme du lièvre et de la tortue

```
Floyd(u0, f)
    lièvre ← f(f(u0))
    tortue ← f(u0)
    k ← 1
    While lièvre ≠ tortue faire                B1
        lièvre ← f(f(lièvre))
        tortue ← f(tortue)
        k ← k + 1
    tortue ← f(tortue)
    T ← 1
    While lièvre ≠ tortue faire                B2
        tortue ← f(tortue)
        T ← T + 1
    lièvre ← u0
    r ← 0
    While lièvre ≠ tortue faire                B3
        lièvre ← f(lièvre)
        tortue ← f(tortue)
        r ← r + 1
    renvoyer r, T
```

2 Schéma d'explication



La première boucle, B1, permet de trouver un entier k tel que $u_k = u_2k$

B2, permet de déterminer la période de u, T^*, schématisé par la flèche bleue.

Enfin, B3, schématisé par les flèches oranges, nous permet de trouver le rang de la suite u, noté r^*.

La flèche bleue représente le parcours de "Tortue" pendant la boucle B2

Enfin, les flèches oranges représentent l'évolution simultanée de "Lièvre" et "Tortue" pendant la boucle B3

3 les invariants

3.1 Invariants globaux

$\forall n \leq r^* - 1; u_n \neq u_{n+T}$
$\forall 1 \leq t \leq T^* - 1; u_r \neq u_{r+t}$

3.2 Invariants de B1

$\forall k \geq 1$,
Lièvre $= u_{2k}$
Tortue $= u_k$

3.3 Invariants de B2

$\forall T \geq 1$,
Lièvre $= u_k$
Tortue $= u_k + T$
$\forall 1 \leq t \leq T^* - 1; u_k \neq u_{k+t}$

3.4 Invariants de B3

$\forall r \geq 0$,
Lièvre $= u_r$
Tortue $= u_k + r$
$\forall n \leq r^* - 1; u_n \neq u_{n+k}$

4 Preuve de correction et terminaison, boucle par boucle

4.1 Boucle 1

La boucle B1 termine, et à la fin de la boucle, on trouve $k \geq r^*$, avec k multiple de T^*

Preuve :

Soit n tel que $nT^* \geq r^*$.

Alors on a $u_{2nT^*} = u_{nT^*}$ par périodicité.

Ce qui veut dire qu'a l'itération nT^* de B1, Lièvre = Tortue.

Donc B1 termine. Et à ce moment, $u_{2k} = u_{k+k}$, donc k est un multiple de la période T^* et $k \geq r^*$.

4.2 Boucle 2

B2 termine et, à la fin de la boucle, T = T^*

Preuve : On a $k \geq r^*$, donc par périodicité $u_k = u_{k+T^*}$,

Et, d'après l'invariant de B2, $\forall 1 \leq t \leq T^* - 1; u_k \neq u_{k+t}$.

Donc $T^* \geq T$, et comme, à chaque itération, T augmente, $T^* - T$ est une quantité positive décroissante, donc B2 termine.

Au moment où B2 termine, $u_k = u_{k+T}$ et $\forall 1 \leq t \leq T^* - 1; u_k \neq u_{k+t}$, donc T est la plus petite période de u, donc T = T^*

4.3 Boucle 3

La boucle B3 termine et, à la fin de la boucle, $r = r^*$

Preuve : On a $T^*|k$, donc par périodicité $u_r^* = u_{k+r^*}$,

Et, d'après l'invariant de B3, $\forall n \leq r^* - 1; u_n \neq u_{n+k}$

Donc $r^* \geq r$, et comme, à chaque itération, r augmente, $r^* - r$ est une quantité positive décroissante, donc B2 termine.

Au moment où B3 termine, $u_r = u_{k+r}$ et $\forall n \leq r^* - 1; u_n \neq u_{n+k}$, donc r est le premier indice à partir duquel u devient périodique, donc r = r^*

5 Complexité

La complexité en espace de l'algorithme est constante, car nous avons toujours besoin du meme nombre de variables (ici 5)

Pour la compléxité en temps, il est évident que B2 va admettre au plus T^* itérations, et que B3 en admettra au plus r^*.

Enfin, pour B1, son nombre d'itérations est un multiple de T^*. Enfin, dans chaque boucle n'apparaissent que des opérations de complexité temporelle en O(1). Donc on a une complexité temporelle en O($T^* + r^*$)