Alexandre DIKOVSKY
Université de Nantes

Logique Mathématique pour l'Informatique

Cours 8. Applications de la logique.

8.1. Analyse logique des programmes.

Spécifications de programmes en termes de *triplets de Hoare* ont la forme :
$\{pre\}[Prog]\{post\}$ où $Prog$ est un programme et $pre \in \mathcal{L}^1$ (*précondition*) et $post \in \mathcal{L}^1$ (*pos condition*) sont des formules en signature de $Prog$.
Sémantique de triplets : $\models \{pre\}[Prog]\{post\}=_{df}$ pour tout contexte initial C_I tel qu $Prog(C_I) = C_F$ est défini, si $C_I \models pre$, alors $C_F \models post$.

Exemple 1. *Soient deux programmes :*

$PROG_1$:
if $(x > 2)$
  {**if** $(y == 2)$
    z :=x-y
  else
    $z := 0$
  }

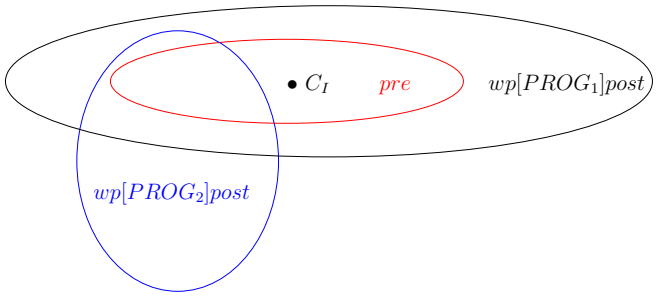
$PROG_2$:
if $(x > 2)$ {
  if $(y == 2)$
    $z := x$-y
  } **else**
    $z := 0$

Soient les conditions :
$pre=_{df}\ (x > 2\ \wedge\ \neg(y == 2)\ \wedge\ z == 3)$ *(précondition) et*
$post=_{df}\ (z == 0)$ *(postcondition).*
Question : *Peut on distinguer $PROG_1$ et $PROG_2$ par ces conditions ? C'est-à-dire, peut-o trouver un contexte initial $C_I \models pre$ tel que $PROG_1(C_I) = C_{F1}$, $PROG_2(C_I) = C_{F2}$ $C_{F1} \models post$ mais $C_{F2} \not\models post$ (ou vice versa) ?*

On peut résoudre ce problème par une méthode logique. Représentons-nous qu'il y ait un formule $wp[PROG]post \in \mathcal{L}^1$ telle que :

$$\boxed{\exists\ C'\ (PROG(C) = C'\ \wedge\ C' \models post)\ \Leftrightarrow\ C \models wp[PROG]post}$$

et qu'on montre que :
(i) $pre \models wp[PROG_1]post$,
(ii) $pre \not\models wp[PROG_2]post$.
Alors, un tel C_I existe :



Cette formule s'appèle *condition la plus faible (CPF) de correction* de $PROG$ vis-à-v *post*. On peut définir un compilateur des paires (programme ; postcondition) en CPF.

Fragment du compilateur :

($\mathbf{R_{skip}}$) $wp[skip]\Phi = \Phi$
($\mathbf{R_{:=}}$) $wp[x := e]\Phi = \Phi[x\backslash e]$
($\mathbf{R_{;}}$) $wp[P_1; P_2]\Phi = wp[P_1](wp[P_2]\Phi)$
($\mathbf{R_{if}}$) $wp[\textbf{if } \phi \textbf{ then } P_1 \textbf{ else } P_2]\Phi = (\phi \rightarrow wp[P_1]\Phi) \ \wedge \ (\neg\phi \rightarrow wp[P_2]\Phi)$
($\mathbf{R_{while}}$) :
$wp[\textbf{while } \phi \textbf{ do } P](\neg\phi \ \wedge \ \Phi) = \neg\phi \rightarrow \Phi$,
$wp[\textbf{while } \phi \textbf{ do } P](\phi \ \wedge \ \Phi) = \phi \rightarrow wp[P](\phi \ \wedge \ \Phi)$

Compilation de $PROG_1, post$ **:**

$WP_1 =_{df} wp[\textbf{if } x > 2 \textbf{ then } (\textbf{if } y == 2 \textbf{ then } z := x - y \textbf{ else } z := 0) \textbf{ else } skip]z == 0 =_{\mathbf{R_{if}}}$
$(x > 2 \rightarrow wp[\textbf{if } y == 2 \textbf{ then } z := x - y \textbf{ else } z := 0]z == 0) \ \wedge$
$(x \leq 2 \rightarrow wp[skip]z == 0) =_{\mathbf{R_{skip}}}$
$(x > 2 \rightarrow wp[\textbf{if } y == 2 \textbf{ then } z := x - y \textbf{ else } z := 0]z == 0) \ \wedge \ (x \leq 2 \rightarrow z == 0) =_{\mathbf{R_{if}}}$
$(x > 2 \rightarrow ((y == 2 \rightarrow wp[\ z := x - y]z == 0) \ \wedge$
$(\neg(y == 2) \rightarrow wp[z := 0]z == 0)) \ \wedge$
$(x \leq 2 \rightarrow z == 0) =_{\mathbf{R_{:=}}}$
$(x > 2 \rightarrow ((y == 2 \rightarrow x - y == 0) \ \wedge \ (\neg(y == 2) \rightarrow 0 == 0)) \ \wedge \ (x \leq 2 \rightarrow z == 0) \ \equiv$
$(x > 2 \rightarrow (y == 2 \rightarrow x - y == 0)) \ \wedge \ (x \leq 2 \rightarrow z == 0)$.

Compilation de $PROG_2, post$ **:**

$WP_2 =_{df} wp[\textbf{if } x > 2 \textbf{ then } (\textbf{if } y == 2 \textbf{ then } z := x - y \textbf{ else } skip) \textbf{ else } z := 0]z == 0 =_{\mathbf{R_{if}}}$
$(x > 2 \rightarrow wp[\textbf{if } y == 2 \textbf{ then } z := x - y \textbf{ else } skip]z == 0) \ \wedge$
$(x \leq 2 \rightarrow wp[z := 0]z == 0) =_{\mathbf{R_{:=}}}$
$(x > 2 \rightarrow wp[\textbf{if } y == 2 \textbf{ then } z := x - y \textbf{ else } skip]z == 0) \ \wedge \ (x \leq 2 \rightarrow 0 == 0) \ \equiv$
$x > 2 \rightarrow wp[\textbf{if } y == 2 \textbf{ then } z := x - y \textbf{ else } skip]z == 0 =_{\mathbf{R_{if}}}$
$x > 2 \rightarrow ((y == 2 \rightarrow wp[z := x - y]z == 0) \ \wedge \ (\neg(y == 2) \rightarrow wp[skip]z == 0)) =_{\mathbf{R_{skip}}}$
$x > 2 \rightarrow ((y == 2 \rightarrow wp[z := x - y]z == 0) \ \wedge \ (\neg(y == 2) \rightarrow z == 0)) =_{\mathbf{R_{:=}}}$
$x > 2 \rightarrow ((y == 2 \rightarrow x - y == 0) \ \wedge \ (\neg(y == 2) \rightarrow z == 0))$.

Théorème 1. *(Théorème logique)*
(i) $pre \models WP_1$.
(ii) $pre \not\models WP_2$.

Preuve : $(x > 2 \ \wedge \ \neg(y == 2) \ \wedge \ z == 3)$ contredit à WP_2.

Corollaire 1. *Alors, il existe un contexte* C_I *tel que :*
(i) $C_I \models \{pre\}[PROG_1]\{post\}$,
(ii) $C_I \not\models \{pre\}[PROG_2]\{post\}$.

8.2. Programmation logique.

8.2.1. Programmes logiques.

Une proposition du 1^{er} ordre est *universelle* si elle est en forme prénexe et si les quantificateurs qui apparaissent dans son préfixe sont tous universels. Par exemple, les clauses du 1^{er} ordre sont propositions universelles.

Définition 1. Classes des clauses. Clause négative : $\bigvee_{i=1}^{m} \neg A_i,\ m \geq 1, A_i$: *atomes.*

Clause non-négative : $\bigvee_{i=1}^{m} A_i \ \vee \ \bigvee_{j=1}^{n} \neg B_j,\ m \geq 1, A_i, B_j$: *atomes.*

Requête logique *est une expression :*

$$\Gamma \ \Rightarrow \bar{\exists}\bar{x} \ \bigwedge_{i=1}^{k} A_i,$$

A_i : *atomes,* Γ : *un ensemble fini de clauses non-négatives.*

Requête logique $\Gamma \ \Rightarrow \bar{\exists}\bar{x} \ \bigwedge_{i=1}^{k} A_i$ *est* décidable : *il existe une telle substitution* $\theta : \{\bar{x} \mapsto \bar{t}\}$, *où* $\bar{t}$ *sont des termes, que* $\Gamma \ \Rightarrow (\bigwedge_{i=1}^{k} A_i) \circ \theta$ *est vrai.*

Exemple 2. *Requête logique :*
$\{member(X, [X|R]), \neg member(X, R) \vee member(X, [Y|R])\} \Rightarrow \exists Z \ member(b, [a, Z, c])$
est décidable : $Z \mapsto b$.

Une particularité importante de la solution dans cet exemple est qu'elle est *symbolique* : définie sur le domaine d'Herbrand.

Rappel du théorème central d'Herbrand :

Théorème 2. *[Herbrand] Un ensemble de propositions universelles est cohérent ssi il a un modèle d'Herbrand.*

Corollaire 2. *Si* Γ *est un ensemble de clauses, alors une requête logique* $\Gamma \Rightarrow \bar{\exists}\bar{x} \ \bigwedge_{i=1}^{k} A_i$ *est décidable ssi elle l'est sur les modèles d'Herbrand.*

Appelons **base de connaissance (BC)** tout ensemble de clauses non-négatives, *qui contient au moins un fait.* Il est évident que toute BC M est cohérente (donc a un modèle d'Herbrand) : en fait $H_{\Sigma_f} \models M$.

Définition 2. Sémantique déclarative. Sémantique d'une BC M *est l'ensemble de ses conséquences closes :* $sem(M) =_{df} \{A \in H_{\Sigma_f} | M \models A\}$.

Exemple 3. *BC* **member** *:* $sem(\{member(X, [X|R]), \neg member(X, R) \vee member(X, [Y|R])\})$ $= \{member(E, Liste) | E \in Liste\}$.

Théorème 3. $sem(M) = \bigcap\{H \mid H \models M$ *où* H *: un modèle d'Herbrand*$\}$ *pour toute BC* M.

Corollaire 3. *Soit une BC* Γ. *Alors :*

1. Une requête logique $\Gamma \Rightarrow \bar{\exists}\bar{x} \ \bigwedge_{i=1}^{k} A_i$ *est décidable ss'il existe une telle substitution* θ *que* $A_i \circ \theta \in sem(\Gamma)$ *pour tous* $1 \leq i \leq k$.

2. Une requête logique $\Gamma \Rightarrow \bar{\exists}\bar{x} \ \bigwedge_{i=1}^{k} A_i$ *est décidable ssi* $\Gamma \ \cup \ \{\neg\bar{\exists}\bar{x} \ \bigwedge_{i=1}^{k} A_i\}$ *est non cohérent.*

3. Si $sem(\Gamma) \neq \emptyset$, *alors* $sem(\Gamma)$ *est le modèle* minimal *d'Herbrand de* Γ.
4. Le modèle minimal de Γ *est unique.*

Par ailleurs, toutes les BC n'ont pas la sémantique $sem(M)$ non vide.

Exemple 4. *Soit* $M_0 =_{df} \{p(x) \vee q(x)\}$. *Cette BC a trois modèles d'Herbrand dont l'nter-section est vide :* $H_1 = \{p(c)\}, H_2 = \{q(c)\}, H_3 = \{p(c), q(c)\}$.

Pour cette raison, on considère les BC qui n'utilisent pas la disjonction dans les têtes (c'est-à-dire la disjonction des atomes positifs dans les clauses). On les appelle BC **à closes définies (BCD)**.

Théorème 4. *Toute BCD* M *a la sémantique* $sem(M)$ *non-vide et donc unique.* $sem(M)$ *est le modèle d'Herbrand minimal de* M.

DÃ» à cette sémantique particulière, on change aussi la syntaxe des BC et des BCD :

Cas général des BC : Pour tous $k, l > 0$

$$(h_1; ...; h_k :- b_1, ..., b_l.)^{po} = \forall \bar{X}\ (b_1 \wedge ... \wedge b_l \longrightarrow h_1 \vee ... \vee h_k) \quad (\textit{une } \textbf{clause}).$$

Pour $k > 0, l = 0$

$$(h_1; ...; h_k.)^{po} = \forall \bar{X}\ (h_1 \vee ... \vee h_k) \quad (\textit{un } \textbf{fait}).$$

Pour $l > 0, k = 0$

$$(:- b_1, ..., b_l.)^{po} = \forall \bar{X}\ \neg(b_1 \wedge ... \wedge b_l) \ \equiv\ \forall \bar{X}\ (\neg b_1 \vee ... \vee \neg b_l) \ \equiv\ \neg \exists \bar{X}\ (b_1 \wedge ... \wedge b_l)\ (\textit{un } \textbf{objectif}).$$

$$\Box^{po} = \mathbf{1} \longrightarrow \mathbf{0} \quad (\textit{une } \textbf{contradiction}).$$

BCD : clauses définies : $(h :- b_1, ..., b_l.)^{po} = \forall \bar{X}\ (b_1 \wedge ... \wedge b_l \longrightarrow h)$.
Faits définis : $(h.)^{po} = \forall \bar{X}\ (h)$.
Un **programme logique à clauses définies (PLD)** est une paire $P = (M; G)$, où M est une BCD et G est un objectif.

Exemple 5. *BCD de concaténation des listes* **append**.
$append([\,], L, L).$
$append([E|L1], L2, [E|L]) :-$
  $append(L1, L2, L).$
$prefix(P, L) :-$
  $append(P, L_0, L).$
Un PLD utilisant **append** *avec l'obgectif :*
$:- prefix([a, X], [Z, b, c, d]).$
En particulier, $prefix([a, b], [a, b, c, d]) \in sem(append)$.

8.2.2. Sémantique opérationnelle de PLD.

Résolution linéaire : Soit un PLD $P = (M;\ :- G_1, ..., G_k)$. Soit un objectif

$$:- A_1, ..., A_i, p(\bar{u}), A_{i+1}, ..., A_n.$$

et une clause (ou un fait)

$$p(\bar{v}) :- B_1, ..., B_m. \in M \ (m \geq 0).$$

Alors,

$$:- A_1, ..., A_i, p(\bar{u}), A_{i+1}, ..., A_n. \vdash^{\theta} (:- A_1, ..., A_i, B_1, ..., B_m, A_{i+1}, ..., A_n.) \circ \theta,$$

si $\bar{u} \circ \theta = \bar{v} \circ \theta$ (θ est *l'unifieur minimal*).
Un calcul de P est une suite :

$$R_0 = (:- G_1, ..., G_k \vdash^{\theta_1}, ..., \vdash^{\theta_k} \square).$$

La substitution $\theta_1 \circ ... \circ \theta_k$ est une *solution* de l'objectif $:- G_1, ..., G_k$.

Remarque. La traduction R^{po} de ce calcul est une réfutation par résolution du premier ordre. Alors, on obtient :

Théorème 5. *$A \in sem(M)$ ssi il existe un calcul $(:- A \vdash^{\theta_1}, ..., \vdash^{\theta_k} \square)$ du PLD $(M; :- A)$.*

Exemple 6. *Un calcul de* $(append; :- prefix([a, X], [Z, b, c, d]).)$
$:- prefix([a, X], [Z, b, c, d]). \vdash$
$:- append([a, X], L_0, [Z, b, c, d]). \vdash^{\theta_1}$
$append([E|L_1], L_2, [E|L]) = append([a, X], L_0, [Z, b, c, d]) \Rightarrow$
$\theta_1 = \{E = a = Z, L_1 = [X], L = [b, c, d], L_0 = L_2\}.$
$:- append([X], L_2, [b, c, d]). \vdash^{\theta_2}$
$append([E^1|L_1], L_2^1, [E^1|L^1]) = append([X], L_2, [b, c, d]) \Rightarrow$
$\theta_2 = \{E^1 = X = b, L_1^1 = [], L^1 = [c, d], L_2^1 = L_2\}.$
$:- append([], L_2^1, [c, d]). \vdash^{\theta_3}$
$append([\ ], L^2, L^2) = append([\ ], L_2^1, [c, d]) \Rightarrow$
$\theta_3 = \{L_2^1 = L_2 = L_0 = [c, d]\}.$
$\square$
Solution : $\theta_1 \circ \theta_2 \circ \theta_3 = \{X = b, Z = a, L_0 = [c, d]\}.$

Les programmes logiques ainsi définis sont non-déterministes. Si on organise la recherche d'un calcul dans la profondeur, de gauche à droite et avec les backtracking aux cas d'échecs, alors on obtient **prolog**.