Logiques de programmes

1 Logique de Floyd–Hoare.

On considère des formules logiques, des *assertions* (définies formellement ci-après), que l'on notera A, A', B, *etc.* Un triplet de Hoare est de la forme $\{A\}c\{A'\}$ (la notation est inhabituelle pour les triplets, mais c'est une notation commune dans le cas des triplets de Hoare), où l'on nomme A la *précondition* et A' la *postcondition*.

Exemple 1. Les triplets suivants sont des triplets de Hoare :

1. $\{x \geq 1\}y := x + 2\{x \geq 1 \wedge y \geq 3\}$ qui est une conclusion naturelle ;
2. $\{n \geq 1\}c_{\text{fact}}\{r = n!\}$ où l'on note c_{fact} la commande

   $$x := n \;;\; z := 1 \;;\; \texttt{while}\,(x > 0)\,\texttt{do}\,(z := z \times x \;;\; x := x - 1)\,,$$

   qui calcule naturellement la factorielle de n ;
3. $\{x < 0\}c\{\texttt{true}\}$ même s'il ne nous dit rien d'intéressant (tout état mémoire vérifie `true`) ;
4. $\{x < 0\}c\{\texttt{false}\}$ qui diverge dès lors que $x < 0$.

On considère un ensemble $I \ni i$ infini d'*index*, des « inconnues ». On commence par définir les expressions arithmétiques étendues

$$a ::= \underline{k} \mid a_1 \oplus a_2 \mid x \mid i,$$

puis définit les *assertions* par la grammaire ci-dessous :

$$A ::= bv \mid A_1 \vee A_2 \mid A_1 \wedge A_2 \mid a_1 \geq a_2 \mid \exists i, A.$$

On s'autorisera à étendre, implicitement, les opérations réalisées dans les expressions arithmétiques, et les comparaisons effectuées dans les assertions.

On ajoute la liaison d'α-conversion : les assertions $\exists i, x = 3 * i$ et $\exists j, x = 3 * j$ sont α-équivalentes. On note $i\ell(A)$ l'ensemble des index libres de l'assertion A, et on dira que A est *close* dès lors que $i\ell(A) = \emptyset$. On note aussi $A[^k/_i]$ l'assertion A où $k \in \mathbb{Z}$ remplace $i \in I$.

Définition 1. Considérons A close et $\sigma \in \mathscr{M}$. On définit par induction sur A (4 cas) une relation constituée de couples (σ, A), notés $\sigma \models A$ (« σ satisfait A »), et en notant $\sigma \not\models A$ lorsque (σ, A) n'est pas dans la relation :

- $\sigma \models \texttt{true}\ \forall \sigma \in \mathscr{M}$;
- $\sigma \models A_1 \vee A_2$ si et seulement si $\sigma \models A_1$ ou $\sigma \models A_2$;
- $\sigma \models a_1 \geq a_2$ si et seulement si on a $a_1, \sigma \Downarrow k_1$ et $a_2, \sigma \Downarrow k_2$ et $k_1 \geq k_2$;
- $\sigma \models \exists i, A$ si et seulement s'il existe $k \in \mathbb{Z}$ tel que $\sigma \models A[^k/_i]$.

On écrit $\models A$ (« A est valide ») lorsque pour tout σ tel que $\mathrm{dom}(\sigma) \supseteq \mathsf{vars}(A)$, on a $\sigma \models A$.

1.1 Règles de la logique de Hoare : dérivabilité des triplets de Hoare.

Les triplets de Hoare, notés $\{A\}c\{A'\}$ avec A et A' closes, où A est *précondition*, c est commande IMP, et A' est *postcondition*. On définit

une relation $\vdash \{A\}c\{A'\}$ sur les triplets de Hoare :

$$\frac{\vdash \{A \wedge b\}c_1\{A'\} \quad \vdash \{A \wedge \neg b\}c_2\{A'\}}{\vdash \{A\}\texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2\{A'\}} \qquad \frac{}{\vdash \{A\}\texttt{skip}\{A\}}$$

$$\frac{\vdash \{A\}c_1\{A'\} \quad \vdash \{A'\}c_2\{A''\}}{\vdash \{A\}c_1 \texttt{ ; } c_2\{A''\}} \qquad \frac{\vdash \{A \wedge b\}c\{A\}}{\{A\}\texttt{while } b \texttt{ do } c\{A \wedge \neg B\}}$$

$$\frac{\models B \implies A \quad \models A' \implies B' \quad \{A\}c\{A'\}}{\{B\}c\{B'\}} \qquad \frac{}{\{A[a/x]\}x \texttt{ := } a\{A\}}$$

La dernière règle semble à l'envers, mais c'est parce que la logique de Hoare fonctionne fondamentalement à l'envers.

Dans la règle de dérivation pour la boucle `while`, l'assertion manipulée, A, est un *invariant*.

L'avant dernière règle s'appelle la *règle de conséquence* : on ne manipule pas le programme, la commande, mais plutôt les pré- et post-conditions.

La relation $\vdash \{A\}c\{A'\}$ s'appelle la *sémantique opérationnelle* de IMP.

Définition 2. On définit la relation de *satisfaction*, sur les triplets de la forme $\{A\}c\{A'\}$ avec A, A' closes, avec $\sigma \models \{A\}c\{A'\}$ si et seulement si dès lors que $\sigma \models A$ et $c, \sigma \Downarrow \sigma'$ alors on a $\sigma' \models A'$.

On définit ensuite la relation de *validité* par $\models \{A\}c\{A'\}$ si et seulement si pour tout $\sigma \in \mathscr{M}$, $\sigma \models \{A\}c\{A'\}$.

Théorème 1 (Correction de la logique de Hoare.)**.** Si $\vdash \{A\}c\{A'\}$ alors $\models \{A\}c\{A'\}$.

Preuve. On procède par induction sur $\vdash \{A\}c\{A'\}$. Il y a 6 cas.

▷ *Règle de conséquence.* On sait

$$\models B \implies A \text{ et } \models A' \implies B',$$

et l'hypothèse d'induction. On doit montrer $\models \{B\}c\{B'\}$. Soit σ tel que $\models B$, et supposons $c, \sigma \Downarrow \sigma'$. On a $\models A$ par hypothèse. Puis, par hypothèse d'induction, $\sigma' \models A'$ et donc $\sigma' \models B'$.

▷ *Règle* `while`. Considérons $c =$ `while` b `do` c_0. On sait par induction que $\models \{A \wedge b\}c_0\{A\}$ et l'hypothèse d'induction. Il faut montrer $\models \{A\}$`while` b `do` $c_0\{A \wedge \neg b\}$, c'est à dire, si $\sigma \models A$ et $(\star)$: `while` b `do` $c_0, \sigma \Downarrow \sigma'$ alors $\sigma' \models A \wedge \neg b$. Pour montrer cela, il est nécessaire de faire une induction sur la dérivation de $(\star)$, « sur le nombre d'itérations dans la boucle ».

▷ Autres cas en exercice.

□

Le sens inverse, la réciproque, s'appelle la *complétude*. On l'étudiera rapidement après.

Remarque 1. Concrètement, on écrit des programmes annotés.

$\{x \geq 1\}$

$\{x \geq 1 \wedge x+2+x+2 \geq 6\}$

$y := x + 2$;

$\{x \geq 1 \wedge y + y \geq 6\}$

$z := y + y$

$\{x \geq 1 \wedge z \geq 6\}$

$\{x \geq 1 \wedge z \geq 6\}$

Pour démontrer la complétude de la logique de Hoare, on s'appuie sur la notion de *plus faible précondition* : étant données une commande c et une assertion B, alors la *plus faible précondition* associée à c, B est l'ensemble des états mémoire

$$\mathrm{wp}(c, B) := \{\sigma \mid c, \sigma \Downarrow \sigma' \implies \sigma' \models B\}.$$

Ainsi, $\mathrm{wp}(c, B)$ est l'ensemble des états mémoire à partir duquels on aboutit à un état satisfaisant B, après une exécution terminante de c.

Proposition 1. Pour toute commande c et toute formule B, il existe une assertion $\mathrm{W}(c, B)$ telle que $\sigma \models \mathrm{W}(c, B)$ si et seulement si $\sigma \in \mathrm{wp}(c, B)$.

Preuve. On procède par induction sur c. Tout fonctionne, sauf pour *while*... Pour le cas de la boucle *while*, on utilise la caractérisation suivante :

$$\sigma \in \mathrm{wp}(\texttt{while}\ b\ \texttt{do}\ c_0, B)$$

$$\Updownarrow$$

$$\forall k, \forall \sigma_0, \ldots, \sigma_k \text{ si } \sigma_0 = \sigma \text{ et } \forall i < k, (\sigma_i, b \Downarrow \texttt{true} \text{ et } c_0, \sigma_i \Downarrow \sigma_{i+1})$$
$$\text{alors } \sigma_k \models b \vee B.$$

On peut définir cette assertion en définissant des assertions pour :

- ▷ décrire un état mémoire σ_i $(X_1^i = v_1 \wedge \cdots \wedge X_n^i = v_n)$;
- ▷ exprimer les conditions $\sigma_i, c \Downarrow \sigma_{i+1}$ par induction ;
- ▷ exprimer les quantifications $\forall k, \sigma_0, \ldots, \sigma_k \ldots$ on demande à Kurt Gödel.

Ainsi, on a bien une assertion $\mathrm{W}(c, B)$ telle que

$$\forall \sigma, \qquad \sigma \in \mathrm{wp}(c, B) \iff \sigma \models \mathrm{W}(c, B).$$

□