TD6 - Logique de Hoare I

Pourquoi: analyse statique du code. Difficile à faire dans les programmes compliqués, plus sûr que des tests qui ne couvrent potentiellement pas tous les cas de figure.

Logique de Hoare: fondement scientifique de

Sémantique axiomatique

Denotational Semantics	Big Step OS	Small Step OS	Hoare Logic
$\vdash (P) = f$	$\vdash (P, \sigma) \Downarrow (v, \sigma')$	$\vdash (P, \sigma) \rightarrow (P', \sigma')$	$\{P\}P\{Q\}$

Règles de Hoare

$$\frac{}{\{Q[\frac{\text{expr}}{v}]\}v = \text{expr}\{Q\}}(\text{aff})$$

$$\frac{\{P\}I_1\{Q_1\} \;\; \{Q_1\}I_2\{Q_2\} \;\; .. \;\; \{Q_{n-1}\}I_n\{Q\}}{\{P\}I_1; I_2; ..; I_n\{Q\}}(\text{seq})$$

$$\frac{P \Rightarrow P' \;\; \{P'\}C\{Q\}}{\{P\}C\{Q\}}(\text{mp-pre})$$

$$\frac{\{P_1\}C_1\{Q\} \;\; \{P_2\}C_2\{Q\}}{\{(B \Rightarrow P_1); (\neg B \Rightarrow P_2)\} \;\; \text{if } B \text{ then } C_1 \text{ else } C_2\{Q\}}(\text{if})$$

Ex1. Plus faible précondition

Q1. Affectations

Satisfaire $\{P\}i = i + 1\{i > 0\}$

1. $\{(i > 0)[\frac{i+1}{i}]\}i = i + 1\{i > 0\}$
   - $(i > 0)[\frac{i+1}{i}] \Leftrightarrow (i + 1 > 0) \Leftrightarrow (i > -1) \Leftrightarrow i \geq 0$

Satisfaire $\{P\}k = (lo + hi)\text{div}2\{lo \leq k \leq hi\}$

1. $\{(lo \leq \frac{lo+hi}{2} \leq hi)\}k = \frac{lo+hi}{2}\{lo \leq k \leq hi\}$
   - $(lo \leq \frac{lo+hi}{2} \leq hi)$
   - $\Leftrightarrow lo \leq \frac{lo+hi}{2} \wedge hi \geq \frac{lo+hi}{2}$

- $\Leftrightarrow \frac{lo}{2} \leq \frac{hi}{2} \wedge \frac{lo}{2} \leq \frac{hi}{2}$
- $\Leftrightarrow lo \leq hi \wedge lo \leq hi$
- $\Leftrightarrow lo \leq hi$

Q2. Séquencement

Satisfaire $\{P\}x = x - 1; y = y - 1\{x = y\}$

1. $\{x = y - 1\}y = y - 1\{x = y\}$ (aff)
2. $\{x - 1 = y - 1\}x = x - 1;\{x = y\}$ (aff)
   - $\Leftrightarrow x = y$
3. $\{x = y\}x = x - 1; y = y - 1\{x = y\}$ (seq)(2)(1)

Satisfaire $\{P\}y = x; u = 4 * x + 3 * y; t = 3 * x + 5 * y\{t = 8 \wedge u = 7\}$

```
$\left\{
\begin{tabular}{@{}l@{}}
    tomatoes \\
    onions \\
    cucumbers
\end{tabular}
\right\}$
```

1. $\{3 * x + 5 * y = 8u = 7\}\, t = 3 * x + 5 * y\{t = 8 \wedge u = 7\}$ (aff)
2. $\{3*x+5*y = 8 \wedge 4*x+3*y = 7\}u = 4*x+3*y\{3*x+5*y = 8 \wedge u = 7\}$ (aff)
3. $\{3*x+5*x = 8 \wedge 4*x+3*x = 7\}y = x\{3*x+5*y = 8 \wedge 4*x+3*y = 7\}$ (aff)
   - $\Leftrightarrow 8x = 8; 7x = 7$
   - $\Leftrightarrow x = 1$
4. $\{x = 1\}y = x; u = 4*x+3*y; t = 3*x+5*y\{t = 8 \wedge u = 7\}$ (seq)(3)(2)(1)

Q3. Alternative

Satisfaire $\{P\}$ if $(x > 0)$ $z = x$ else $z = -x\{z = |x|\}$

1. $\{-x = |x|\}z = -x\{z = |x|\}$(aff)
2. $\{x = |x|\}z = x\{z = |x|\}$(aff)
3. $\{(x > 0) \Rightarrow x = |x| \wedge \neg(x > 0) \Rightarrow -x = |x|\}$if..else..$\{z = |x|\}$ (if)(2)(1)
   - $\Leftrightarrow (x > 0) \Rightarrow x = |x| \wedge (x \leq 0) \Rightarrow -x = |x|$

- $\Leftrightarrow \top$

Satisfaire $\{P\}\ x = 4;$ if $(x > y)\ z = x$ else $z = y\ \{z = 3\}$

1. $\{y = 3\}z = y\{z = 3\}$(aff)
2. $\{x = 3\}z = x\{z = 3\}$(aff)
3. $\{x > y \Rightarrow x = 3 \wedge \neg x > y \Rightarrow y = 3\}if..\{z = 3\}$(if)(2)(1)
   - $\Leftrightarrow (x > y) \Rightarrow x = 3 \wedge (x \leq y) \Rightarrow y = 3$
4. $\{(y < 4) \Rightarrow 4 = 3 \wedge (y \geq 4) \Rightarrow y = 3\}x = 4\{(x > y) \Rightarrow x = 3 \wedge (x \leq y) \Rightarrow y = 3\}$(aff)
5. $\{(y < 4) \Rightarrow 4 = 3 \wedge (y \geq 4) \Rightarrow y = 3\}Prog\{(x > y) \Rightarrow x = 3 \wedge (x \leq y) \Rightarrow y = 3\}$(seq)(4)(3)
   - $(y < 4) \Rightarrow 4 = 3$: explosion si $y < 4$
   - $y \geq 4 \wedge y \geq 4 \Rightarrow y = 3 \wedge y = 3 \Leftrightarrow \bot \wedge y \geq 4 \Rightarrow y = 3$

Ex2. Preuve de programme

Démontrer $\{x > 2\}\ a = 1; y = x; y = y - a\ \{y > 0 \wedge x > y\}$

1. $\{y > a \wedge x + a > y\}\ y = y - a\ \{y > 0 \wedge x > y\}$(aff)
2. $\{x > a \wedge a > 0\}\ y = x\ \{y > a \wedge x + a > y\}$(aff)
3. $\{x > 1 \wedge 1 > 0\}\ a = 1\ \{x > a \wedge a > 0\}$(aff)
4. $\{x > 1 \wedge 1 > 0\}\ Prog\ \{y > 0 \wedge x > y\}$(seq)(3)(2)(1)
   - $x > 1$
5. $x \geq 2 \Rightarrow x \geq 1$(tautologie)
6. $\{x > 2\}Prog\{y > 0 \wedge x > y\}$(mp)(5)(4)

Ex3. Des contrats aux preuves

Code:

```
PRE
INV
<capture>
<corps>
INV
POST
```

Preuve: $\{PRE \wedge INV\} < capture >; < corps > \{POST \wedge INV\}$