INF-565 - Projet Coq

2016-2017

On se propose ici de développer en Coq un calcul de plus petite condition pour le langage While. Cette procédure sera prouvée correcte en Coq et extraite vers le langage OCaml.

1 Le langage While

Définition 1.1 (c.p.o. pointé)**.** *Soit X un ensemble. On dénote par $X^\perp$ le c.p.o. pointé $(X \cup \{\perp\}, \preceq)$ où $\perp$ est symbole arbitraire que l'on suppose* frais *(notamment $\perp \notin X$) et $\preceq$ est le plus petit pré-ordre sur $X \cup \{\perp\}$ t.q. $\forall x \in X.\ \perp \preceq x$. Ainsi défini, $X^\perp$ est un ω-c.p.o.*

Définition 1.2 (Langage While)**.** *Soit $\mathcal{X} = \{x, y, \ldots\}$ un ensemble infini dénombrable de* variables de programme *et $\mathcal{E} = \{e, \ldots\}$ un ensemble d'expressions de programme. La syntaxe du langage* While *est définie comme suit :*

$$\begin{array}{rr} s ::= \textbf{skip} & \textit{(no-op)} \\ \mid \textbf{abort} & \textit{(arrêt)} \\ \mid x \leftarrow e & \textit{(affectation)} \\ \mid s; s & \textit{(séquence)} \\ \mid \textbf{if } (e) \textbf{ then } s \textbf{ else } s & \textit{(conditionnelle)} \\ \mid \textbf{while } (e) \textbf{ do } s & \textit{(boucle while)} \end{array}$$

On écrit **if** (e) **then** s *pour* **if** (e) **then** s **else skip**.

Définition 1.3 (Sémantique)**.** *Une mémoire est une fonction de $\mathcal{X}$ dans $\mathbb{Z}$. On dénote par $\mathcal{M} = \{m, \ldots\}$ l'ensemble de toutes les mémoires. Pour une mémoire $m \in \mathcal{M}$, une variable $x \in \mathcal{X}$ et une valeur $z \in \mathbb{Z}$, on écrit $m[x \leftarrow z]$ pour la mémoire définie par :*

$$\begin{cases} m[x \leftarrow z](y) = z & \textit{si } x = y \\ m[x \leftarrow z](y) = m(y) & \textit{sinon.} \end{cases}$$

On se donne également une fonction $[\![e]\!]_m \in \mathbb{Z}$ d'interprétation des expressions. La sémantique $[\![s]\!]$ d'un programme While *est une fonction de $\mathcal{M}^\perp$ dans $\mathcal{M}^\perp$ définie inductivement sur la structure de s :*

$$[\![s]\!](\perp) = \perp \qquad [\![\textbf{skip}]\!](m) = m$$
$$[\![\textbf{abort}]\!](m) = \perp \qquad [\![x \leftarrow e]\!](m) = m[x \leftarrow [\![e]\!]_m]$$
$$[\![s_1; s_2]\!](m) = [\![s_2]\!]([\![s_1]\!](m)) \qquad [\![\textbf{while } (e) \textbf{ do } s]\!] = \bigsqcup_{n \in \mathbb{N}} [\![s^n_{\neg e}]\!](m)$$
$$[\![\textbf{if } (e) \textbf{ then } s_1 \textbf{ else } s_2]\!](m) = \begin{cases} [\![s_1]\!](m) \textit{ si } [\![e]\!]_m \neq 0 \\ [\![s_2]\!](m) \textit{ sinon} \end{cases}$$

où $s^n = s; \cdots; s$ (n fois), $s^n_{\neg e} = ((\textbf{if } (e) \textbf{ then } s)^n; \textbf{if } (e) \textbf{ then abort})$ et $\bigsqcup_{n \in N} \alpha_n$ est le plus petit majorant de la ω-chaîne $\{\alpha_n\}_{n \in \mathbb{N}}$ (ici, sur $\mathcal{M}^\perp$).

Lemme 1.4. *La définition précédente est correcte.*

Démonstration. Par induction sur s. Le cas délicat est celui de la boucle **while**. La preuve se fait en montrant que $\{s^n_{\neg e}\}_{n \in \mathbb{N}}$ est une ω-chaîne, prouvant de facto l'existence du plus petit majorant. □

Lemme 1.5 (Equation de point-fixe)**.** *Soit s un programme et e une expression. Alors*

$$[\![\textbf{while } (e) \textbf{ do } s]\!](m) = [\![\textbf{if } (e) \textbf{ then } s; \textbf{while } (e) \textbf{ do } s]\!](m).$$

Notamment, $[\![\textbf{while } (e) \textbf{ do } s]\!](m) = \bigsqcup_{n\in\mathbb{N}} [\![s^{n+p}_{\neg e}]\!](m)$ *pour n'importe quel* $p \in \mathbb{N}$ *arbitraire.*

Démonstration. Conséquence immédiate de la définition de la sémantique de la boucle **while** en tant que plus petit point-fixe de $n \mapsto [\![s^n_{\neg e}]\!](m)$. □

Lemme 1.6 (Terminaison certaine)**.** *Soit s un programme, e une expression et m une mémoire t.q.* $[\![\textbf{while } (e) \textbf{ do } s]\!](m) \neq \bot$*. Alors, il existe un entier n t.q. i)* $[\![s^n]\!](m) \models \neg e$*, ii)* $\forall p < n. [\![s^p]\!](m) \models e$ *et iii)* $[\![\textbf{while } (e) \textbf{ do } s]\!](m) = [\![s^n]\!](m)$*.*

Démonstration. Soit $E = \{n \in \mathbb{N} \mid [\![(\textbf{if } (e) \textbf{ then } s)^n]\!](m) \models \neg e\}$. Si E est vide, alors pour tout $n \in \mathbb{N}$, $[\![s^n_{\neg e}]\!](m) = \bot$. Ainsi, $[\![\textbf{while } (e) \textbf{ do } s]\!](m) = \bot$, ce qui contredit les hypothèses du lemme. Ainsi, E est non vide. Soit $\mathfrak{n}$ le plus petit élément de E. Tout d'abord, on montre que pour tout $p \leq \mathfrak{n}$, on a $[\![(\textbf{if } (e) \textbf{ then } s)^p]\!](m) = [\![s^p]\!](m)$. Si $p = 0$, c'est immédiate. Si $p = k + 1 > 0$, alors

$$\begin{aligned}
[\![(\textbf{if } (e) \textbf{ then } s)^{k+1}]\!] &= [\![\textbf{if } (e) \textbf{ then } s]\!](\underbrace{[\![(\textbf{if } (e) \textbf{ then } s)^k]\!](m)}_{\models e \text{ par minimalité de } \mathfrak{n}}) \\
&= [\![s]\!](\underbrace{[\![(\textbf{if } (e) \textbf{ then } s)^k]\!](m)}_{= [\![s^k]\!](m) \text{ par I.H.}} = [\![s^{k+1}]\!](m),
\end{aligned}$$

ce qui termine l'induction. Ainsi, de ce qui précède et par minimalité de $\mathfrak{n}$, $[\![s^{\mathfrak{n}}]\!](m) \models \neg e$ et $[\![s^p]\!](m) \models e$ pour tout $p < \mathfrak{n}$. Maintenant, pour n'importe quel $l \in \mathbb{N}$, nous avons

$$\begin{aligned}
[\![s^{\mathfrak{n}+l}_{\neg e}]\!](m) &= [\![s^l_{\neg e}]\!](\underbrace{[\![(\textbf{if } (e) \textbf{ then } s)^{\mathfrak{n}}]\!](m)}_{\models \neg e \text{ par définition de } \mathfrak{n}}) \\
&= [\![s^0_{\neg e}]\!]([\![(\textbf{if } (e) \textbf{ then } s)^{\mathfrak{n}}]\!]) && \text{(induction sur } l\text{)} \\
&= [\![\textbf{if } (e) \textbf{ then abort}]\!](\underbrace{[\![(\textbf{if } (e) \textbf{ then } s)^{\mathfrak{n}}(m)]\!]}_{\models \neg e \text{ par définition de } \mathfrak{n}}) \\
&= [\![(\textbf{if } (e) \textbf{ then } s)^{\mathfrak{n}}]\!](m) = [\![s^{\mathfrak{n}}]\!](m).
\end{aligned}$$

Ainsi, $[\![\textbf{while } (e) \textbf{ do } s]\!](m) = \bigsqcup_{l\in\mathbb{N}} [\![s^l_{\neg e}]\!](m) = \bigsqcup_{l\in\mathbb{N}} [\![s^{\mathfrak{n}+l}_{\neg e}]\!](m) = \bigsqcup_{l\in\mathbb{N}} [\![s^{\mathfrak{n}}]\!](m) = [\![s^{\mathfrak{n}}]\!](m)$. □

Définition 1.7 (Assertion)**.** *On appelle* assertion *toute partie (ou sous-ensemble) de $\mathcal{M}$ et on écrit $\mathcal{A} = \{\phi, \psi, \ldots\}$ pour l'ensemble des assertions. La notation de validité d'une assertion est réduite à celle d'appartenance ensembliste : $m \models \phi$ ssi $m \in \phi$. Il est usuel décrire $\models \phi$ pour $\forall m \in \mathcal{M}. m \models \phi$. On définit les connecteurs suivants sur les assertions :*

$$\begin{aligned}
\top &\triangleq \mathcal{M} & \bot &\triangleq \emptyset \\
\neg\phi &\triangleq \phi^{\complement} & \phi \wedge \psi &\triangleq \phi \cap \psi \\
\phi \vee \psi &\triangleq \phi \cup \psi \quad \phi \implies \psi \triangleq \neg\phi \vee \psi \\
\forall x. \phi &\triangleq \{m \mid \forall z \in \mathbb{Z}. m[x \leftarrow z] \models \phi\} \\
\exists x. \phi &\triangleq \{m \mid \exists z \in \mathbb{Z}. m[x \leftarrow z] \models \phi\}.
\end{aligned}$$

La notion de validité et de satisfiabilité est réifié au niveau des assertions :

$$\begin{aligned}
\forall \mathfrak{m}. \phi &\triangleq \top \textit{ si } \forall m. m \models \phi, \bot \textit{ sinon} \\
\exists \mathfrak{m}. \phi &\triangleq \top \textit{ si } \exists m. m \models \phi, \bot \textit{ sinon}
\end{aligned}$$

On définit également une notion de substitution pour les assertions :

$$\begin{aligned}
\phi[x \leftarrow z] &\triangleq \{m \mid m[x \leftarrow z] \models \phi\} \\
\phi[x \leftarrow e] &\triangleq \{m \mid m[x \leftarrow [\![e]\!]_m] \models \phi\}.
\end{aligned}$$

Enfin, toute expression e définit une assertion $\hat{e}$ comme suit :

$$\hat{e} \triangleq \{m \in \mathcal{M} \mid [\![e]\!]_m \neq 0\}.$$

On confond $\hat{e}$ et e quand le contexte le permet.

$$\textsc{Skip}\ \frac{}{\vdash \{\phi\}\ \textbf{skip}\ \{\phi\}} \qquad \textsc{Abort}\ \frac{}{\vdash \{\phi\}\ \textbf{abort}\ \{\psi\}} \qquad \textsc{Assgn}\ \frac{}{\vdash \{\phi[x \leftarrow e]\}\ x \leftarrow e\ \{\phi\}}$$

$$\textsc{Conseq}\ \frac{\phi \implies \phi' \qquad \psi' \implies \psi \qquad \vdash \{\phi'\}\ s\ \{\psi'\}}{\vdash \{\phi\}\ s\ \{\psi\}} \qquad \textsc{Seq}\ \frac{\vdash \{\phi\}\ s_1\ \{\xi\} \qquad \vdash \{\xi\}\ s_2\ \{\psi\}}{\vdash \{\phi\}\ s_1; s_2\ \{\psi\}}$$

$$\textsc{If}\ \frac{\vdash \{\phi \wedge e\}\ s_1\ \{\psi\} \qquad \vdash \{\phi \wedge \neg e\}\ s_2\ \{\psi\}}{\vdash \{\phi\}\ \textbf{if}\ (e)\ \textbf{then}\ s_1\ \textbf{else}\ s_2\ \{\psi\}} \qquad \textsc{While}\ \frac{\vdash \{\mathcal{I} \wedge e\}\ s\ \{\mathcal{I}\}}{\vdash \{\mathcal{I}\}\ s\ \{\mathcal{I} \wedge \neg e\}}$$

FIGURE 1 – Règles de HOARE

2 Triplet de HOARE

Définition 2.1 (Triplet de HOARE)**.** *Un triplet de* HOARE *est la donnée de deux assertions ϕ et ψ (la pré- et la post-condition) et d'un programme s t.q.*

$$\forall m \models \phi.\ [\![s]\!](m) \models^{\perp} \psi.$$

où $m \models^{\perp} \phi$ ssi $m = \perp$ ou $m \models \phi$. On note ce triplet $\models \{\phi\}\ s\ \{\psi\}$.

Définition 2.2 (Notion de prouvabilité pour HOARE)**.** *La Figure 1 définit inductivement une relation $\vdash \{\phi\}\ s\ \{\psi\}$ où ϕ et ψ sont des assertions et s un programme.*

Theorème 2.3. *Si $\vdash \{\phi\}\ s\ \{\psi\}$ alors $\models \{\phi\}\ s\ \{\psi\}$.*

Démonstration. On prouve $\models \{\phi\}\ s\ \{\psi\}$ par induction sur $\vdash \{\phi\}\ s\ \{\psi\}$ et par analyse de case sur la dernière règle utilisée dans $\vdash \{\phi\}\ s\ \{\psi\}$. On utilise les notations de la Figure 1.

[Skip] Soit $m \models \phi$. Alors, $[\![\textbf{skip}]\!](m) = m$ et $[\![\textbf{skip}]\!](m) \models \phi$ par hypothèse, donc $[\![\textbf{skip}]\!](m) \models^{\perp} \phi$.

[Abort] Soit $m \models \phi$. Alors $[\![\textbf{abort}]\!](m) \models^{\perp} \psi$ est toujours vrai car $[\![\textbf{abort}]\!](m) = \perp$.

[Assign] Soit $m \models \phi[x \leftarrow e]$ (ou encore, de manière équivalente $m \models \phi[x \leftarrow [\![e]\!]_m]$). On veut montrer que $[\![x \leftarrow e]\!] \models^{\perp} \phi$. Or, $[\![x \leftarrow e]\!] = m[x \leftarrow [\![e]\!]_m]$, et par définition de $\phi[x \leftarrow e]$, $m[x \leftarrow [\![e]\!]_m] \models \phi$. Donc $[\![x \leftarrow e]\!] \models^{\perp} \phi$.

[Conseq] Soit $m \models \phi$. On veut montrer que $[\![s]\!](m) \models^{\perp} \psi$. Par l'hypothèse $\models \phi \implies \phi'$, on sait que $m \models \phi'$. Or, par hypothèse d'induction, on sait également que $\models \{\phi'\}\ s\ \{\psi'\}$. Donc $[\![s]\!](m) \models^{\perp} \psi'$. Finalement, de l'hypothèse $\models \psi' \implies \psi$, on conclut que $[\![s]\!](m) \models^{\perp} \psi$.

[Seq] Soit $m \models \phi$. On veut montrer que $[\![s_1; s_2]\!](m) \models^{\perp} \psi$. Si $[\![s_1; s_2]\!](m) = \perp$, c'est immédiat. Sinon, par la première hypothèse d'induction, on sait que $\models \{\phi\}\ s_1\ \{\xi\}$, et donc $[\![s_1]\!](m) \models^{\perp} \xi$. Nécessairement, comme $[\![s_1; s_2]\!](m) \neq \perp$, $[\![s_1]\!](m) \neq \perp$. Ainsi, $[\![s_1]\!](m) \models \xi$. Maintenant, par la seconde hypothèse d'induction, on a $\models \{\xi\}\ s_2\ \{\psi\}$. Donc, $[\![s_2]\!]([\![s_1]\!](m)) \models \psi$ (et $[\![s_2]\!]([\![s_1]\!](m)) \models^{\perp} \psi$). On conclut en remarquant que $[\![s_2]\!]([\![s_1]\!](m)) = [\![s_1; s_2]\!](m)$.

[If] Soit $m \models \phi$. On veut prouver que $[\![\textbf{if}\ (e)\ \textbf{then}\ s_1\ \textbf{else}\ s_2]\!](m) \models \psi$. Supposons que $[\![e]\!]_m \neq 0$. Alors, $m \models \phi \wedge \hat{e}$, et par la première hypothèse d'induction, nous avons $[\![s_1]\!](m) \models^{\perp} \psi$. Or, comme $[\![e]\!]_m \neq 0$, nous avons également que $[\![\textbf{if}\ (e)\ \textbf{then}\ s_1\ \textbf{else}\ s_2]\!](m) = [\![s_1]\!](m)$, ce qui nous permet de conclure. Le cas $[\![e]\!]_m = 0$ est similaire et utilise la seconde hypothèse d'induction.

[While] Soit $m \models \mathcal{I}$. On veut prouver que $[\![\textbf{while}\ (e)\ \textbf{do}\ s]\!](m) \models^{\perp} \mathcal{I} \wedge \neg e$. Si $[\![\textbf{while}\ (e)\ \textbf{do}\ s]\!](m) = \perp$, c'est immédiat. Sinon par le Lemme 1.6, on sait qu'il existe un certain $n \in \mathbb{N}$ t.q. i) $[\![s^n]\!](m) \models \neg e$, ii) $\forall p < n.\ [\![s^p]\!](m) \models e$ et iii) $[\![\textbf{while}\ (e)\ \textbf{do}\ s]\!](m) = [\![s^n]\!](m)$. On prouve maintenant par induction sur n que $[\![s^n]\!](m) \models \mathcal{I}$. Si $n = 0$, c'est immédiat car $[\![s^0]\!](m) = m \models \mathcal{I}$. Si $n = p+1 > 0$, alors par hypothèse d'induction interne, $[\![s^p]\!](m) \models \mathcal{I}$. Par hypothèse sur n, on sait que $[\![s^p]\!](m) \models e$. Ainsi, $[\![s^p]\!](m) \models \mathcal{I} \wedge e$, et par hypothèse d'induction externe, on obtient $[\![s]\!]([\![s^p]\!](m)) \models \mathcal{I}$. On termine l'induction interne en remarquant que $[\![s]\!]([\![s^p]\!](m)) = [\![s^n]\!](m)$. Enfin, par propriété sur n, on sait que $[\![s^n]\!](m) \models \neg e$. Ainsi, $[\![s^n]\!](m) \models \mathcal{I} \wedge \neg e$, ou encore $[\![\textbf{while}\ (e)\ \textbf{do}\ s]\!](m) \models^{\perp} \mathcal{I} \wedge \neg e$. □

3 Calcul de plus petite précondition

Définition 3.1 (Plus petite précondition)**.** *Soit s un programme et ϕ une assertion. On suppose que toute boucle* **while** (e) **do** s' *de s est décorée d'un invariant $\mathcal{I} \in \mathcal{A}$, décoration que l'on note* $\mathbf{while}^{\mathcal{I}}\ (e)$ **do** s'*. La* plus petite précondition de ϕ pour s*, notée* $\mathrm{wp}^\star(s, \phi)$*, est définie inductivement comme suit :*

$$
\begin{aligned}
&\mathrm{wp}^\star(\mathbf{skip}, \phi) \triangleq \phi \qquad \mathrm{wp}^\star(\mathbf{abort}, \phi) \triangleq \top \\
&\mathrm{wp}^\star(x \leftarrow e, \phi) \triangleq \phi[x \leftarrow e] \quad \mathrm{wp}^\star(s_1; s_2, \phi) \triangleq \mathrm{wp}^\star(s_1, \mathrm{wp}^\star(s_2, \phi)) \\
&\quad \mathrm{wp}^\star(\mathbf{if}\ (e)\ \mathbf{then}\ s_1\ \mathbf{else}\ s_2, \phi) \triangleq \\
&\qquad (e \implies \mathrm{wp}^\star(s_1, \phi)) \wedge (\neg e \implies \mathrm{wp}^\star(s_2, \phi)) \\
&\quad \mathrm{wp}^\star(\mathbf{while}^{\mathcal{I}}\ (e)\ \mathbf{do}\ s, \phi) \triangleq \\
&\qquad \mathcal{I} \\
&\qquad \wedge \forall \mathtt{m}.\, (e \wedge \mathcal{I} \implies \mathrm{wp}^\star(s, \mathcal{I})) \\
&\qquad \wedge \forall \mathtt{m}.\, (\neg e \wedge \mathcal{I} \implies \phi)
\end{aligned}
$$

Theorème 3.2. *Il est toujours vrai que* $\models \{\mathrm{wp}^\star(s, \psi)\}\ s\ \{\psi\}$*. Notamment, si* $\models \phi \implies \mathrm{wp}^\star(s, \psi)$*, alors* $\models \{\phi\}\ s\ \{\psi\}$*.*

Démonstration. On prouve par induction sur s que $\models \{\mathrm{wp}^\star(s, \psi)\}\ s\ \{\psi\}$. On ne détaille que le cas pour la boucle **while**, les autres cas étant immédiats. Supposons que $s = \mathbf{while}^{\mathcal{I}}\ (e)$ **do** s' et soit m t.q. $m \models \mathrm{wp}^\star(s, \phi)$. Notons que de cette dernière propriété, nous avons que $\models e \wedge \mathcal{I} \implies \mathrm{wp}^\star(s, \mathcal{I})$ et $\models \neg e \wedge \mathcal{I} \implies \phi$. Nous voulons montrer que $[\![s]\!](m) \models^\perp \phi$. Si $[\![s]\!](m) = \perp$, alors $[\![s]\!](m) \models^\perp \phi$ par définition et la preuve est finie. Sinon, par le Lemme 1.6, il existe un $n \in \mathbb{N}$ t.q. $[\![s]\!](m) = [\![(s')^n]\!](m)$, $[\![(s')^n]\!] \models \neg e$ et $\forall p < n.\, [\![(s')^p]\!](m) \models e$. Nous prouvons par induction sur n que $[\![(s')^n]\!] \models \phi$. Si $n = 0$, alors $m \models \neg e$. De plus, de $m \models \mathrm{wp}^\star(s, \phi)$, on a $m \models \mathcal{I}$ et nous savons que $\models \neg e \wedge \mathcal{I} \implies \phi$. Ainsi, $m \models \phi$. Si maintenant $n = p + 1 > 0$, alors par hypothèse d'induction interne, $[\![(s')^p]\!](m) \models \mathcal{I}$. Posons $m_p \triangleq [\![(s')^p]\!](m)$. Par hypothèse sur n, nous savons également que $m_p \models e$. Ainsi, de $\models e \wedge \mathcal{I} \implies \mathrm{wp}^\star(s, \mathcal{I})$, nous obtenons que $m_p \models \mathrm{wp}^\star(s', \mathcal{I})$. Par hypothèse d'induction externe, nous obtenons que $[\![s']\!](m_p) \models \mathcal{I}$, i.e. que $[\![(s')^n]\!](m) \models \mathcal{I}$. Par définition de n, $[\![(s')^n]\!](m) \models \neg e$. Ainsi, de $\models \neg e \wedge \mathcal{I} \implies \phi$, nous avons $[\![(s')^n]\!](m) \models \phi$, ce qui termine l'induction interne et le cas pour **while** — nous avons prouvé que $[\![s]\!](m) \models^\perp \phi$. □

4 Travail à faire

On vous demande de

1. Définir en Coq la syntaxe du langage While — le langage des expressions pourra être laissé abstrait en prenant n'importe quelle fonction de `mem` dans `int`, i.e. du type des mémoires dans le type des entiers relatifs.
2. Définir en Coq la sémantique des programmes de While. On pourra dans un premier temps axiomatiser la théorie des c.p.o.
3. Définir En Coq la notion de validité et de provabilité pour Hoare — une assertion sera alors une fonction de `mem` dans `Prop`.
4. Démontrer en Coq le Théorème 2.3.
5. Définir en Coq une fonction `wp` qui calcule la plus petite précondition d'une assertion ϕ pour un programme s. Cette dernière échouera dans un premier temps lorsque s comporte une boucle.
6. Démontrer en Coq que votre fonction `wp` de calcul de plus petite précondition est correcte, i.e. le Théorème 3.2.
7. Définir en Coq une syntaxe pour les assertions et leur interprétation en tant qu'assertion logique. Écrire une fonction `wps` qui calcule la plus petite précondition d'une assertion *syntaxique* ϕ pour un programme s.
8. Comme pour `wp`, montrer que `wps` est correcte. On pourra se ramener à la preuve de correction de `wp`.
9. Concevoir un mécanisme permettant de donner les invariants de boucle nécessaires au calcul de plus petite précondition pour les boucles. Étendre vos fonctions `wp` et `wps` en conséquence, ainsi que leurs preuves de correction.

10. Extraire à partir de votre développement Coq une fonction OCaml de calcul de plus petite précondition. Que dire de cette dernière ?