4. PREUVES D'INVARIANCE

4. PREUVES D'INVARIANCE

4.1 RELATIONS ENTRE SEMANTIQUES CONSERVANT L'INVARIANCE

4.1.1 CONSERVATION DE PROPRIETES D'INVARIANCE POUR DES SEMANTIQUES CONCORDANTES A DES RELATIONS ENTRE ETATS PRES

4.1.2 CONSERVATION DE PROPRIETES D'INVARIANCE APRES REDUCTION DES ETATS INOBSERVABLES

4.1.3 CONSERVATION DE PROPRIETES D'INVARIANCE PAR RETRACTION DE LA SEMANTIQUE PAR TRANSITIONS

4.2 PRINCIPES D'INDUCTION

4.2.1 PRINCIPES D'INDUCTION POUR LES SEMANTIQUES CLOSES

4.2.1.1 Principe d'induction de base pour l'invariance

4.2.1.2 Transformations de principes d'induction

4.2.1.2.1 Transformation par distinction/confusion des états initiaux ou finaux

4.2.1.2.2 Transformation par déduction/prédiction

4.2.1.2.3 Transformation par inversion

4.2.1.2.4 Transformation contrapositive

4.2.1.2.5 Transformation relation/assertion

4.2.1.3 Principes d'induction dérivés par transformations

4.2.1.4 Equivalence forte des principes d'induction dérivés

4.4 REFERENCES

4. PREUVES D'INVARIANCE

Floyd [67] et Naur [66] sont souvent cités comme étant à l'origine des preuves de correction partielle des programmes déterministes séquentiels (bien que l'idée remonte certainement aux origines de la programmation (Turing, Von Neumann). Hoare [68] introduisit la présentation des preuves par induction sur la syntaxe des programmes. La méthode de Morris-Wegbreit [77] (dite "subgoal induction") a montré beaucoup plus tard qu'il n'y a pas qu'une seule manière de faire des preuves d'invariance. La généralisation au cas des programmes parallèles conduisit à une profusion de méthodes qu'il est bien difficile de comparer ne serait-ce que parceque les langages de programmation sont différents (Ashcroft [75], Ashcroft-Manna [70], Hoare [75], Howard [76], Keller [76], Lamport [77], Mazurkiewicz [77], Newton [75], Owicki-Gries [76a], [76b], etc.).

Le but de notre travail est de présenter un modèle abstrait pour étudier les méthodes de preuve d'invariance. Il s'agit d'en formaliser l'essence à l'aide de principes d'induction, d'en étudier la correction et la complétude relativement à une sémantique, de les comparer notamment du point de vue de l'équivalence forte et de proposer une méthode de construction systématique d'une méthode de preuve d'invariance à partir d'une définition de la sémantique opérationnelle.

4.1 RELATIONS ENTRE SEMANTIQUES CONSERVANT L'INVARIANCE

Pour démontrer une propriété d'un programme relativement à une sémantique, on cherche souvent à se ramener à une sémantique plus simple conservant la propriété à démontrer.

La relation entre ces deux sémantiques est souvent exprimée indirectement, par exemple elle est induite par une transformation du programme :

- Un exemple très courant est celui de la compilation. Pour éviter d'avoir à prendre en compte la compilation dans une preuve d'invariance, on ne raisonne jamais sur la sémantique du code objet mais toujours sur une sémantique du code source. Cette démarche est implicitement basée sur une hypothèse de correction du compilateur que nous pouvons formuler en disant qu'après réduction des états inobservables, les sémantiques source et objet sont concordantes à une relation entre états et actions près.

- Un autre exemple est fourni par l'utilisation que font Owicki-Gries [76a] de variables auxiliaires dans les preuves de programmes : une preuve de correction d'un programme P se fait en raisonnant sur un programme transformé P' qui utilise un ensemble VA de variables dites auxiliaires qui n'apparaissent que dans des commandes d'affectation $X := E$ telles que $X \in VA$, et tel que P s'obtient à partir de P' en enlevant toutes les commandes d'affectation à ces variables auxiliaires. En définissant les états inobservables comme ceux dont l'état de contrôle désigne une commande d'affectation à une variable auxiliaire, on observe qu'après réduction des états inobservables, les sémantiques de P' et P sont concordantes à une fonction des états près qui consiste à éliminer les variables auxiliaires de l'état mémoire.

4.1.1 CONSERVATION DE PROPRIETES D'INVARIANCE POUR DES SEMANTIQUES CONCORDANTES A DES RELATIONS ENTRE ETATS PRES

Théorème 4.1.1 n°1

Si $\simeq <rs, ra>(<S,A,\Sigma>, <S',A',\Sigma'>)$ alors

est invariante pour $<S,A,\Sigma>$ $\Rightarrow$ $\not\Leftarrow$ $[rs^{-1} \circ \psi \circ rs$ est invariante pour $<S',A',\Sigma'>]$

Démonstration

($\Rightarrow$) Si ψ est invariante pour $<S,A,\Sigma>$, $p' \in \Sigma'$ et $i \in |p'|$ alors il existe $p \in \Sigma$ tel que $|p| = |p'|$, $rs(p_0, p'_0)$ et $rs(p_i, p'_i)$. Par conséquent $\psi(p_0, p_i)$ entraine $rs^{-1} \circ \psi \circ rs\ (p'_0, p'_i)$.

($\not\Leftarrow$) Choisir $S = \{0,1\}$, $A = \phi$, $\Sigma = \{0,1\}$, $S' = \{0'\}$, $\Sigma' = \{0'\}$, $rs(0,0')$, $rs(1,0')$, $\psi(0,0)$, $\neg\psi(1,1)$. Nous avons $rs^{-1} \circ \psi \circ rs\ (0',0')$ et donc $rs^{-1} \circ \psi \circ rs$ est invariante pour Σ' mais ψ n'est pas invariante pour Σ.

□

La réciproque est vraie si nous ajoutons une condition supplémentaire :

Théorème 4.1.1 n°2

$$\simeq <rs, ra>(<S,A,\Sigma>, <S',A',\Sigma'>) \qquad (1)$$

$$(rs^{-1} \circ \psi \circ rs\ (\Delta'_1, \Delta'_2) \wedge rs^{-1}(A'_1, A_1) \wedge rs(A_2, A'_2)) \Rightarrow \psi(A_1, A_2) \qquad (2)$$

$[\psi$ est invariante pour $<S,A,\Sigma>] \Leftrightarrow [rs^{-1} \circ \psi \circ rs$ est invariante pour $<S',A',\Sigma'>]$

Démonstration

($\Rightarrow$) La même que pour 4.1.1 v 1. ($\Leftarrow$) Si $rs^{-1} \circ \psi \circ rs$ est invariante pour $\langle S', A', \Sigma' \rangle$, $p \in \Sigma$ et $i \in |p|$ alors il existe $p' \in \Sigma'$ tel que $rs(p_0, p'_0)$ et $rs(p_i, p'_i)$ - d'après (1). Par conséquent, $rs^{-1} \circ \psi \circ rs\ (p'_0, p'_i)$ entraine d'après (2), $\psi(p_0, p_i)$.

□

Observons que les théorèmes 4.1.1 v 1 et 4.1.1 v 2 ne sont pas vrais pour l'invariance conditionnelle. Un contre-exemple (pour 4.1.1 v 1) est donné par $S = \{0,1,2\}$, $A = \{a\}$, $\Sigma = \{0 \xrightarrow{a} 1\}$, $S' = \{0', 1'\}$, $A' = \{a'\}$, $\Sigma' = \{0' \xrightarrow{a'} 1'\}$, $\neg\phi(0,0)$, $\psi(0,0)$, $\neg\psi(0,1)$ (de sorte que ψ est invariant sous condition ϕ pour Σ), $rs(0,0')$, $rs(1,1')$, $ra(a,a')$ (de sorte que $\simeq \langle rs, ra \rangle (\Sigma, \Sigma')$) et $\phi(2,2)$, $rs(2,0')$ et $\neg\psi(2,1)$ (de sorte que $rs^{-1} \circ \psi \circ rs\ (0,0')$ est vrai tandis que $rs^{-1} \circ \psi \circ rs\ (0', 1')$ est faux). Toutefois sous des conditions plus restrictives, nous obtenons les théorèmes 4.1.1 v 3 et 4.1.1 v 4 suivants (dont les théorèmes 4.1.1 v 1 et 4.1.1 v 2 sont des cas particuliers respectifs) :

Théorème 4.1.1 v 3

Si

$$\simeq \langle rs, ra \rangle (\langle S, A, \Sigma \rangle, \langle S', A', \Sigma' \rangle) \tag{1}$$

$$\wedge \quad (\phi'(A'_1, A'_2) \wedge rs^{-1}(A'_1, A_1) \wedge rs(A_2, A'_2)) \Rightarrow \phi(A_1, A_2) \tag{2}$$

$$\wedge \quad (\psi(A_1, A_2) \wedge rs(A_1, A'_1) \wedge rs^{-1}(A'_2, A_2)) \Rightarrow \psi'(A'_1, A'_2) \tag{3}$$

alors

[ψ est invariante sous ϕ pour $\langle S, A, \Sigma \rangle$]

$\not\Rightarrow$

[ψ' est invariante sous ϕ' pour $\langle S, A, \Sigma \rangle$]

Démonstration

($\Rightarrow$) Si $p' \in \Sigma'$, $i \in |p'|$ et $\forall j \in i . \phi'(p'_0, p'_j)$ alors d'après (1), $\exists p \in \Sigma$ tel que $|p| = |p'|$ et $\forall j < i . rs(p_j, p'_j)$. Par conséquent d'après (2) nous avons $\forall j \in i . \phi(p_0 p_j)$ ce qui implique $\psi(p_0, p_i)$ et donc $\psi'(p'_0, p'_i)$ d'après (3).

($\Leftarrow$) Même choix que dans la démonstration de 4.1.1 v 1 avec $\phi'(s, s') = tt$.

□

Théorème 4.1.1 n°4

Si

$$\simeq \langle rs, ra\rangle(\langle S,A,\Sigma\rangle, \langle S',A',\Sigma'\rangle) \quad (1)$$

$$\wedge\ (\phi'(A'_1,A'_2) \wedge rs^{-1}(A'_1,A_1) \wedge rs(A_2,A'_2)) \Rightarrow \phi(A_1,A_2) \quad (2)$$

$$\wedge\ (\psi(A_1,A_2) \wedge rs(A_1,A'_1) \wedge rs^{-1}(A'_2,A_2)) \Rightarrow \psi'(A'_1,A'_2) \quad (3)$$

$$\wedge\ (\phi(A_1,A_2) \wedge rs(A_1,A'_1) \wedge rs^{-1}(A'_2,A_2)) \Rightarrow \phi'(A'_1,A'_2) \quad (4)$$

$$\wedge\ (\psi'(A'_1,A'_2) \wedge rs^{-1}(A'_1,A_1) \wedge rs(A_2,A'_2)) \Rightarrow \psi(A_1,A_2) \quad (5)$$

[ψ est invariante sous ϕ pour $\langle S,A,\Sigma\rangle$]

[ψ' est invariante sous ϕ' pour $\langle S',A',\Sigma'\rangle$]

Démonstration

($\Rightarrow$) La même que pour 4.1.1 n°3. ($\Leftarrow$) Si $p\in\Sigma$, $i\in|p|$ et $\forall j\in i.\ \phi(p_0,p_j)$ alors d'après (1), $\exists p'\in\Sigma'$ tel que $|p'|=|p|$ et $\forall j\in i.\ rs(p_j,p'_j)$. Par conséquent d'après (4) nous avons $\forall j\in i.\ \phi'(p'_0,p'_j)$ ce qui implique $\psi'(p'_0,p'_i)$ et donc $\psi(p_0,p_i)$ d'après (5).

□

Dans le cas particulier d'une concordance entre sémantiques à une fonction entre états près, nous obtenons le corollaire suivant :

Corollaire 4.1.1 n°5

Si

$$\langle S',A',\Sigma'\rangle \quad \simeq \langle f_A\rangle(\langle S,A,\Sigma\rangle)$$

$$\wedge\ \phi(A_1,A_2) \quad \phi'(f_A(A_1), f_A(A_2))$$

$$\wedge\ \psi(A_1,A_2) \quad \psi'(f_A(A_1), f_A(A_2))$$

[ψ est invariante sous condition ϕ pour $\langle S,A,\Sigma\rangle$]

$\Leftrightarrow$

[ψ' est invariante sous condition ϕ' pour $\langle S',A',\Sigma'\rangle$]

4.1.2 CONSERVATION DE PROPRIETES D'INVARIANCE APRES REDUCTION DES ETATS INOBSERVABLES

La conservation d'une propriété d'invariance après réduction des états inobservables est décrite (dans un cas simplifié mais dont la généralisation est aisée) par :

Théorème 4.1.2v1

Si

$$\langle S', A', \Sigma' \rangle = \underline{\text{Redei}}\ \langle S' \rangle (\langle S, A, \Sigma \rangle) \quad \wedge \quad \forall p \in \Sigma . p_0 \in S'$$

alors

$$[\Psi' \text{ est invariante sous condition } \Phi' \text{ pour } \langle S', A', \Sigma' \rangle]$$

$$\Leftrightarrow$$

$$[\Psi(s,s') = [s' \in S' \Rightarrow \Psi'(s,s')] \text{ est invariante sous condition } \Phi(s,s') = [s' \in S' \Rightarrow \Phi'(s,s')] \text{ pour } \langle S, A, \Sigma \rangle]$$

Démonstration

($\Rightarrow$) Soit p une trace de $\langle S, A, \Sigma \rangle$, $i \in |p|$ tel que $\forall j \in i . \Phi(p_0, p_j)$. Si $p_i \notin S'$ alors de manière évidente nous avons $\Psi(p_0, p_i)$. Si $p_i \in S'$, il faut montrer que $\Psi'(p_0, p_i)$ est vrai. Il suffit que $\Phi'(p_0, p_{r(k)})$ soit vrai pour tout $k \geq 0$ tel que $r(k) < i$ (où $r(k)$ a été défini en 2.5.4.1). Si $k = 0$ alors $p_0 \in S'$ implique $r(k) = 0$. Si $i > 0$ alors $p_0 \in S'$ et $\Phi(p_0, p_0)$ impliquent $\Phi'(p_0, p_{r(0)})$. Si $k > 0$ et $r(k) < i$ alors $p_{r(k)} \in S'$ et donc à nouveau $\Phi(p_0, p_{r(k)})$ implique $\Phi'(p_0, p_{r(k)})$.

($\Leftarrow$) Soit p' une trace de $\langle S', A', \Sigma' \rangle$, $i' \in |p'|$ tel que $\forall j \in i' . \Phi'(p'_0, p'_j)$. Ceci entraîne que pour tout $k \geq 0$ tel que $r(k) < r(i')$ nous avons $\Phi(p_0, p_{r(k)})$. Soit maintenant $j \in r(i')$ tel que $p_j \notin S'$. Ceci entraîne de manière évidente $\Phi(p_0, p_j)$. Finalement nous avons $\forall j \in r(i') . \Phi(p_0, p_j)$, d'où nous déduisons $\Psi(p_0, p_{r(i')})$ et donc $\Psi'(p'_0, p'_{i'})$.

□

4.1.3 CONSERVATION DE PROPRIETES D'INVARIANCE PAR RETRACTION DE LA SEMANTIQUE PAR TRANSITIONS

manière générale, on fait des preuves d'invariance par induction sur la longueur des calculs. On cherche donc à utiliser un système de transition. Cette démarche qui consiste à remplacer une preuve d'invariance relative à une sémantique par une preuve relative à la rétraction de cette sémantique par transitions est justifiée par les résultats suivants :

Théorème 4.1.3-1

$$\Rightarrow \begin{array}{l}[\psi \text{ est invariante sous condition } \phi \text{ pour } \langle S,A,\Sigma\rangle \ \wedge \ \langle S',A',\Sigma'\rangle \subseteq \langle S,A,\Sigma\rangle] \\ [\psi \text{ est invariante sous condition } \phi \text{ pour } \langle S',A',\Sigma'\rangle]\end{array}$$

Théorème 4.1.3-2

$$\Leftrightarrow \begin{array}{l}[\psi \text{ est invariante sous condition } \phi \text{ pour } \langle S,A,\Sigma\rangle] \\ [\psi \text{ est invariante sous condition } \phi \text{ pour } Pref^{<\omega}(\langle S,A,\Sigma\rangle)]\end{array}$$

Démonstration

($\Rightarrow$) Supposons $p \in \Sigma^{<\omega}\langle S,A\rangle$, $q \in \Sigma$, $p \mapsto q$, $\forall r \in \Sigma, k \in |r|. [\forall j \in k. \phi(r_0, r_j)] \Rightarrow \psi(r_0, r_k)$ et $i \in |p|$. Alors $\forall k \in |p|. p_k = q_k$ et donc $[\forall j \in i. \phi(p_0, p_j)] \Rightarrow [\forall j \in i. \phi(q_0, q_j)] \Rightarrow \psi(q_0, q_i) \Rightarrow \psi(p_0, p_i)$.

($\Leftarrow$) Si $p \in \Sigma$, $i \in |p|$ et $\forall j \in i. \phi(p_0, p_j)$ alors $p^{\leq i}$ est un préfixe fini de p et donc $\psi(p_0, p_i)$ est vrai.

□

Nous en déduisons que pour faire une preuve d'invariance de ψ sous condition ϕ pour une sémantique $\langle S,A,\Sigma\rangle$, il est toujours correct de faire la preuve relativement à $\underset{\sim}{Rtran}(\langle S,A,\Sigma\rangle)$ c'est-à-dire en raisonnant sur le système de transition qu'elle engendre. Cette démarche n'est pas toujours complète. Plus précisément :

Corollaire 4.1.3 ν3

$[\psi$ est invariante sous condition ϕ pour $\langle S,A,\Sigma\rangle]$

(1) $\Leftrightarrow [\psi$ est invariante sous condition ϕ pour $\underset{\sim}{Pref}(\langle S,A,\Sigma\rangle)]$

(2) $\not\Leftrightarrow [\psi$ est invariante sous condition ϕ pour $\underset{\sim}{Suff}(\langle S,A,\Sigma\rangle)]$

(3) $\Leftrightarrow [\psi$ est invariante sous condition ϕ pour $\underset{\sim}{Redeaa}(\langle S,A,\Sigma\rangle)]$

(4) $\not\Leftrightarrow [\psi$ est invariante sous condition ϕ pour $\underset{\sim}{Efus}(\langle S,A,\Sigma\rangle)]$

(5) $\Leftrightarrow [\psi$ est invariante sous condition ϕ pour $\underset{\sim}{Flim}(\langle S,A,\Sigma\rangle)]$

(6) $\not\Leftrightarrow [\psi$ est invariante sous condition ϕ pour $\underset{\sim}{Retus}(\langle S,A,\Sigma\rangle)]$

(7) $\not\Leftrightarrow [\psi$ est invariante sous condition ϕ pour $\underset{\sim}{Rtran}(\langle S,A,\Sigma\rangle)]$

Si $\underset{\sim}{card}(\alpha)<\omega$ alors

(8) $\Leftrightarrow [\psi$ est invariante sous condition ϕ pour $\underset{\sim}{Wfair}\langle\alpha\rangle(\langle S,A,\Sigma\rangle)]$

(9) $\Leftrightarrow [\psi$ est invariante sous condition ϕ pour $\underset{\sim}{Sfair}\langle\alpha\rangle(\langle S,A,\Sigma\rangle)]$

Démonstration

(1) ($\Leftarrow$) Pref est extensive et 4.1.3 ν1. ($\Rightarrow$) 2.6.1 ν2 et 4.1.3 ν2.

(2) ($\Leftarrow$) Suff est extensive et 4.1.3 ν1. ($\not\Rightarrow$) $\psi(s,s')=[s'=1]$ est invariante sous condition $\phi(s,s')=[s'\in\{0,2\}]$ pour la trace $0\xrightarrow{a}1\xrightarrow{a}2\xrightarrow{a}3$ mais pas pour son suffixe $2\xrightarrow{a}3$.

(3) Trivial.

(4) ($\Leftarrow$) Efus est extensive et 4.1.3 ν1. ($\not\Rightarrow$) Si $\psi(0,0)$, $\psi(0,1)$, $\psi(1,1)$, $\psi(1,2)$ et $\phi(s,s')$ sont vrais alors ψ est invariant sous condition ϕ pour les traces $0\xrightarrow{a}1$ et $1\xrightarrow{a}2$ mais pas pour la fusion $0\xrightarrow{a}1\xrightarrow{a}2$.

(5) 4.1.3 ∨ 2 et 2.6.7 ∨ 3-(1).

(6) (⇒) Retps est réductive et 4.1.3 ∨ 1. (⇍) $\psi(s,s') = ff$ est invariante sous condition $\phi(s,s') = tt$ pour $\langle\{0\},\{a\},\phi\rangle = \text{Retps}(\langle\{0\},\{a\},\Sigma\rangle)$ avec $\Sigma = \{0, 0\xrightarrow{a}0, \ldots, 0\xrightarrow{a}0\xrightarrow{a}0 \ldots 0\xrightarrow{a}0, \ldots\}$ mais pas pour $\langle\{0\},\{a\},\Sigma\rangle$.

(7) (⇐) ψ est invariant sous condition ϕ pour $\text{Rtran}(\langle S,A,\Sigma\rangle)$ si et seulement si d'après 4.1.3 ∨ 3.1 ψ est invariant sous condition ϕ pour $\text{Pref}_0 \circ \text{Rtran}(\langle S,A,\Sigma\rangle)$ ce qui entraine d'après 2.6.8 ∨ 1 et 4.1.3 ∨ 1 que ψ est invariant sous condition ϕ pour $\langle S,A,\Sigma\rangle$. (⇍) Même contre-exemple que pour (4).

(8), (9) 4.1.3 ∨ 2 et 2.6.4 ∨ 1.

□

Observons que la démarche qui consiste à remplacer une preuve d'invariance relative à une sémantique $\langle S,A,\Sigma\rangle$ par une preuve relative $\text{Rtran}(\langle S,A,\Sigma\rangle)$ est toujours correcte mais pas toujours complète (cf. 4.1.3 ∨ 3).

Toutefois la complétude est obtenue si la sémantique $\langle S,A,\Sigma\rangle$ est ermée par fusions :

Théorème 4.1.3 ∨ 4

$$\Rightarrow \begin{array}{l}[\psi \text{ est invariante sous condition } \phi \text{ pour } \langle S,A,\Sigma\rangle \wedge \text{Efus}(\langle S,A,\Sigma\rangle) = \langle S,A,\Sigma\rangle] \\ [\psi \text{ est invariante sous condition } \phi \text{ pour } \text{Rtran}(\langle S,A,\Sigma\rangle)]\end{array}$$

Démonstration

ψ est invariante sous condition ϕ pour $\langle S,A,\Sigma\rangle$ si et seulement si ψ est invariante sous condition ϕ pour $\text{Pref}^{<\omega}(\langle S,A,\Sigma\rangle)$ d'après 4.1.3 ∨ 2. Ceci entraine d'après 2.6.8 ∨ 7 et 4.1.3 ∨ 1 que ψ est invariante sous condition ϕ pour $\text{Pref}^{<\omega} \circ \text{Rtran}(\langle S,A,\Sigma\rangle)$ qui d'après 4.1.3 ∨ 2 implique que ψ est invariante sous condition ϕ pour $\text{Rtran}(\langle S,A,\Sigma\rangle)$.

□

Par conséquent, d'après 4.1.3∨3.8, 4.1.3∨3.9 et 4.1.3∨4, les preuves d'invariance pour le langage que nous avons considéré en 2.8 peuvent toujours se faire en raisonnant sur un système de transition.

Dans le cas d'une sémantique non fermée par fusions, il est tout de même possible de se ramener à un raisonnement sur un système de transition, par exemple en spécifiant cette sémantique par concordance avec une sémantique close (cf. 2.7.2.2).

4.2 PRINCIPES D'INDUCTION

Un principe d'induction est l'essence d'une méthode de preuve.

4.2.1 PRINCIPES D'INDUCTION POUR LES SEMANTIQUES CLOSES

Nous commençons par considérer le cas des sémantiques engendrées par un système de transition.

4.2.1.1 Principe d'induction de base pour l'invariance

Le principe d'induction de base est l'essence de la méthode de Floyd-Naur. Toutes les autres méthodes de preuve de propriétés d'invariance en dérivent. Ce principe est déduit de l'

Exemple 4.2.1.1-1

Pour démontrer la correction partielle du programme suivant qui calcule le quotient et le reste de deux entiers x et y :

```
1:
     Q := 0;
2:
     while X ≥ Y do
3:
          Q := Q+1;
4:
          X := X-Y;
5:
     od;
6:
```

il faut établir la relation $\underline{x} = \bar{q} \times \underline{y} + \bar{x} \wedge \bar{x} < \underline{y}$ entre les valeurs initiales $\underline{x}, \underline{y}, \underline{q}$ et finales $\bar{x}, \bar{y}, \bar{q}$ des variables X, Y et Q.

Comme la méthode de Floyd-Naur n'utilise que des assertions $P(x,y,q)$ sur les valeurs des variables X, Y et Q, il faut introduire une variable auxiliaire XI à laquelle est affectée la valeur initiale de X en début de programme et qui n'est plus modifiée par la suite :

XI:=X; Q:=0; <u>while</u> X ≥ Y <u>do</u> Q:=Q+1; X:=X-Y; <u>od</u>

de sorte qu'à la terminaison nous pourrons prouver que $[xi = q \times y + x \wedge x < y]$.

Cette transformation peut être évitée en utilisant la méthode de Manna[71] qui est tout à fait similaire à la méthode de Floyd-Naur mais qui consiste à utiliser des relations entre valeurs initiales et courantes des variables plutôt que des assertions sur les valeurs courantes. La méthode consiste à associer un invariant local P_i à chaque point i, $i=1,\dots,6$ du programme. L'invariant local P_i associé au point i du programme est une relation entre les valeurs initiales $\underline{x}, \underline{y}$ (nous omettons q qui est inutile) des variables X, Y et les valeurs courantes x, y, q de ces variables X, Y, Q qui est vrai quand le contrôle atteint ce point i :

$P_1(\underline{x},\underline{y},x,y,q) = [x=\underline{x} \wedge y=\underline{y}]$

$P_2(\underline{x},\underline{y},x,y,q) = [x=\underline{x} \wedge y=\underline{y} \wedge q=0]$

$P_3(\underline{x},\underline{y},x,y,q) = [y=\underline{y} \wedge \underline{x}=q\times y+x]$

$P_4(\underline{x},\underline{y},x,y,q) = [y=\underline{y} \wedge \underline{x}=(q-1)\times \underline{y}+x]$

$P_5(\underline{x},\underline{y},x,y,q) = [y=\underline{y} \wedge \underline{x}=q\times \underline{y}+x]$

$P_6(\underline{x},\underline{y},x,y,q) = [\underline{x}=q\times \underline{y}+x \wedge x<\underline{y} \wedge y=\underline{y}]$

Les conditions de vérification sont similaires à celles obtenues par la méthode de Floyd-Naur, excepté pour la condition d'entrée :

$P_1(\underline{x},\underline{y},x,y,q) \Leftarrow [x=\underline{x} \wedge y=\underline{y}]$

$P_2(\underline{x},\underline{y},x,y,q) \Leftarrow [\exists q' . P_2(\underline{x},\underline{y},x,y,q') \wedge q=0]$

$P_3(\underline{x},\underline{y},x,y,q) \Leftarrow [(P_2(\underline{x},\underline{y},x,y,q) \vee P_5(\underline{x},\underline{y},x,y,q)) \wedge x \geq y]$

$P_4(\underline{x},\underline{y},x,y,q) \Leftarrow [\exists q' . P_3(\underline{x},\underline{y},x,y,q') \wedge q=q'+1]$

$P_5(\underline{x},\underline{y},x,y,q) \Leftarrow [\exists x' . P_4(\underline{x},\underline{y},x',y,q) \wedge x=x'-y]$

$P_6(\underline{x},\underline{y},x,y,q) \Leftarrow [(P_2(\underline{x},\underline{y},x,y,q) \vee P_5(\underline{x},\underline{y},x,y,q)) \wedge x<y]$

$[\underline{x}=q\times\underline{y}+x \wedge x<\underline{y} \wedge y=\underline{y}] \Leftarrow P_6(\underline{x},\underline{y},x,y,q)$

Ces conditions de vérification expriment que

- $P_1(\underline{x},\underline{y},\underline{x},\underline{y},\underline{q})$ est vrai quand l'exécution commence avec des valeurs $\underline{x},\underline{y},\underline{q}$ de X,Y,Q
- Si l'exécution commencée avec les valeurs $\underline{x},\underline{y},\underline{q}$ de X,Y,Q atteint le point i du programme qui est immédiatement suivi par le point j du programme, alors l'hypothèse que $P_i(\underline{x},\underline{y},x',y',q')$ est vrai en i pour les valeurs courantes x',y',q' des variables X,Y,Q implique que $P_j(\underline{x},\underline{y},x,y,q)$ est vrai quand le contrôle est en j avec les valeurs x,y,y des variables X,Y,Q.

Par induction sur la longueur des calculs, on en déduit que si l'exécution commence avec $X=\underline{x}$ et $Y=\underline{y}$ et atteint le point i avec $X=x$, $Y=y$ et $Q=q$ alors $P_i(\underline{x},\underline{y},x,y,q)$ est vrai. En particulier, si $i=6$ la dernière condition de vérification permet de conclure que le programme est partiellement correct.

Pour dégager l'essence de cette preuve nous considérons la sémantique du programme (comme elle a été définie en 2.8.1.2) :

Les états $\langle L,M\rangle$ de ce programme consistent en un état de contrôle L (i.e. un point du programme) et un état mémoire (i.e. une fonction M qui définit les valeurs $M(X)$, $M(Y)$, $M(Q)$ des variables X,Y,Q) :

$\mathcal{L} = \{1,2,3,4,5,6\}$

$\mathcal{V} = \{X, Y, Q\}$

$\mathcal{M} = (\mathcal{V} \to \mathcal{D})$

$S = \mathcal{L} \times \mathcal{M}$

Les états initiaux $\underline{s}$ du programme correspondent au point 1 du programme avec des valeurs arbitraires des variables :

$\mathcal{E}(\underline{s}) = [\exists M \in (\mathcal{V} \to \mathcal{D}) . \ \underline{s} = \langle 1, M \rangle]$

La relation de transition est définie par cas en écrivant $\langle L, M \rangle \xrightarrow{t} \langle L', M' \rangle$ quand $t(\langle L, M \rangle, \langle L', M' \rangle)$ est vraie (et en omettant l'action unique, cf. 2.8.1.8.2). En plus (x, y, q) dénote la fonction M lorsque $M(X) = x$, $M(Y) = y$ et $M(Q) = q$ et ces valeurs des variables X, Y, Q du programme sont implicitement universellement quantifiées sur $\mathbb{Z}$:

$\langle 1, (x,y,q) \rangle \xrightarrow{t} \langle 2, (x,y,0) \rangle$

$\langle 2, (x,y,q) \rangle \xrightarrow{t} \langle 3, (x,y,q) \rangle$ si et seulement si $x \geq y$

$\langle 2, (x,y,q) \rangle \xrightarrow{t} \langle 6, (x,y,q) \rangle$ si et seulement si $x < y$

$\langle 3, (x,y,q) \rangle \xrightarrow{t} \langle 4, (x,y,q+1) \rangle$

$\langle 4, (x,y,q) \rangle \xrightarrow{t} \langle 5, (x-y,y,q) \rangle$

$\langle 5, (x,y,q) \rangle \xrightarrow{t} \langle 3, (x,y,q) \rangle$ si et seulement si $x \geq y$

$\langle 5, (x,y,q) \rangle \xrightarrow{t} \langle 6, (x,y,q) \rangle$ si et seulement si $x < y$

Définissons

$\bar{\Psi}(\underline{x}, \underline{y}, \underline{q}, \bar{x}, \bar{y}, \bar{q}) = [\underline{x} = \bar{q} \times \underline{y} + \bar{x} \wedge \bar{x} < \underline{y} \wedge \bar{y} = \underline{y}]$

$\Psi(\underline{s}, \bar{s}) = [\exists \underline{x}, \underline{y}, \underline{q}, \bar{x}, \bar{y}, \bar{q} \in \mathbb{Z} . \ \underline{s} = \langle 1, (\underline{x}, \underline{y}, \underline{q}) \rangle \wedge \bar{s} = \langle 6, (\bar{x}, \bar{y}, \bar{q}) \rangle \wedge \bar{\Psi}(\underline{x}, \underline{y}, \underline{q}, \bar{x}, \bar{y}, \bar{q})]$

alors lorsque nous disons que le programme est partiellement correct cela signifie :

$\forall p \in \Sigma\langle S, A, t, \varepsilon \rangle, \ i \in |p| . \ \Psi(p_0, p_i)$

Nous pouvons maintenant dériver le principe d'induction de l'exemple par abstractions successives :

Notre premiere abstraction consiste à noter que les invariants locaux P_i associés aux points i, $i=1,\dots,6$ du programme peuvent être compris comme une relation I sur des états. Nous avons

$P_1(\underline{x},\underline{y},x,y,q) = [x=\underline{x} \wedge y=\underline{y}]$

$P_2(\underline{x},\underline{y},x,y,q) = [x=\underline{x} \wedge y=\underline{y} \wedge q=0]$

$P_3(\underline{x},\underline{y},x,y,q) = [y=\underline{y} \wedge \underline{x}=q\times\underline{y}+x]$

$P_4(\underline{x},\underline{y},x,y,q) = [y=\underline{y} \wedge \underline{x}=(q-1)\times\underline{y}+x]$

$P_5(\underline{x},\underline{y},x,y,q) = [y=\underline{y} \wedge \underline{x}=q\times\underline{y}+x]$

$P_6(\underline{x},\underline{y},x,y,q) = [\underline{x}=q\times\underline{y}+x \wedge x<\underline{y} \wedge y=\underline{y}]$

de sorte que

$I \in (S\times S \to \{tt,ff\})$

$I(\underline{s},s) = [\exists j\in\mathcal{L}, \underline{x},\underline{y},\underline{q},x,y,q\in\mathbb{Z}.\ \underline{s}=\langle 1,(\underline{x},\underline{y},\underline{q})\rangle \wedge s=\langle j,(x,y,q)\rangle \wedge P_j(\underline{x},\underline{y},x,y,q)]$

Notre seconde abstraction consiste à comprendre les conditions de vérification sur les P_i, $i=1,\dots,6$ en termes de conditions de vérification équivalentes faisant intervenir I et nos définitions abstraites du programme et de sa correction partielle (c'est à dire en termes de S, t, ε, σ et ψ) :

- La premiere condition de vérification était

$[x=\underline{x} \wedge y=\underline{y}] \Rightarrow P_1(\underline{x},\underline{y},x,y,q)$

c'est à dire $P_1(\underline{x},\underline{y},\underline{x},\underline{y},\underline{q})$ qui est équivalent à $\forall \underline{s},\ \varepsilon(\underline{s}) \Rightarrow I(\underline{s},\underline{s})$ puisque $[\exists \underline{x},\underline{y},\underline{q}\in\mathbb{Z}.\ \underline{s}=\langle 1,(\underline{x},\underline{y},\underline{q})\rangle] \Rightarrow I(\underline{s},\underline{s})$ est équivalent à $I(\langle 1,(\underline{x},\underline{y},\underline{q})\rangle,\langle 1,(\underline{x},\underline{y},\underline{q})\rangle)$ soit $[\exists j\in\mathcal{L}.\ j=1 \wedge P_j(\underline{x},\underline{y},\underline{x},\underline{y},\underline{q})] = P_1(\underline{x},\underline{y},\underline{x},\underline{y},\underline{q})$.

- Les conditions de vérification 2 à 6

$[\exists q'.\ P_1(\underline{x},\underline{y},x,y,q') \wedge q=0] \Rightarrow P_2(\underline{x},\underline{y},x,y,q)$

$[(P_2(\underline{x},\underline{y},x,y,q) \vee P_5(\underline{x},\underline{y},x,y,q)) \wedge x\geq y] \Rightarrow P_3(\underline{x},\underline{y},x,y,q)$

$[\exists q'.P_3(\underline{x},\underline{y},x,y,q') \wedge q=q'+1] \Rightarrow P_4(\underline{x},\underline{y},x,y,q)$

$[\exists x'.P_4(\underline{x},\underline{y},x',y,q) \wedge x=x'-y] \Rightarrow P_5(\underline{x},\underline{y},x,y,q)$

$[(P_2(\underline{x},\underline{y},x,y,q) \vee P_5(\underline{x},\underline{y},x,y,q)) \wedge x<y] \Rightarrow P_6(\underline{x},\underline{y},x,y,q)$

sont équivalentes à :

$$[\exists x',y',q'.\ P_1(\underline{x},\underline{y},x',y',q') \wedge x=x' \wedge y=y' \wedge q=0] \Rightarrow P_2(\underline{x},\underline{y},x,y,q)$$

$$[\exists x',y',q'.\ P_2(\underline{x},\underline{y},x',y',q') \wedge x'\geq y' \wedge x=x' \wedge y=y' \wedge q=q'] \Rightarrow P_3(\underline{x},\underline{y},x,y,q)$$

$$[\exists x',y',q'.\ P_5(\underline{x},\underline{y},x',y',q') \wedge x'\geq y' \wedge x=x' \wedge y=y' \wedge q=q'] \Rightarrow P_3(\underline{x},\underline{y},x,y,q)$$

$$[\exists x',y',q'.\ P_3(\underline{x},\underline{y},x',y',q') \wedge x=x' \wedge y=y' \wedge q=q'+1] \Rightarrow P_4(\underline{x},\underline{y},x,y,q)$$

$$[\exists x',y',q'.\ P_4(\underline{x},\underline{y},x',y',q') \wedge x=x'-y' \wedge y=y' \wedge q=q'] \Rightarrow P_5(\underline{x},\underline{y},x,y,q)$$

$$[\exists x',y',q'.\ P_2(\underline{x},\underline{y},x',y',q') \wedge x'<y' \wedge x=x' \wedge y=y' \wedge q=q'] \Rightarrow P_6(\underline{x},\underline{y},x,y,q)$$

$$[\exists x',y',q'.\ P_5(\underline{x},\underline{y},x',y',q') \wedge x'<y' \wedge x=x' \wedge y=y' \wedge q=q'] \Rightarrow P_6(\underline{x},\underline{y},x,y,q)$$

Utilisant I, t, $\underline{M}=(\underline{x},\underline{y},q)$ $M'=(x',y',q')$ et $M=(x,y,q)$, elles peuvent s'écrire :

$$[I(\langle 1,\underline{M}\rangle,\langle 1,M'\rangle) \wedge t(\langle 1,M'\rangle,\langle 2,M\rangle)] \Rightarrow I(\langle 1,\underline{M}\rangle,\langle 2,M\rangle)$$

$$[I(\langle 1,\underline{M}\rangle,\langle 2,M'\rangle) \wedge t(\langle 2,M'\rangle,\langle 3,M\rangle)] \Rightarrow I(\langle 1,\underline{M}\rangle,\langle 3,M\rangle)$$

$$[I(\langle 1,\underline{M}\rangle,\langle 5,M'\rangle) \wedge t(\langle 5,M'\rangle,\langle 3,M\rangle)] \Rightarrow I(\langle 1,\underline{M}\rangle,\langle 3,M\rangle)$$

$$[I(\langle 1,\underline{M}\rangle,\langle 3,M'\rangle) \wedge t(\langle 3,M'\rangle,\langle 4,M\rangle)] \Rightarrow I(\langle 1,\underline{M}\rangle,\langle 4,M\rangle)$$

$$[I(\langle 1,\underline{M}\rangle,\langle 4,M'\rangle) \wedge t(\langle 4,M'\rangle,\langle 5,M\rangle)] \Rightarrow I(\langle 1,\underline{M}\rangle,\langle 5,M\rangle)$$

$$[I(\langle 1,\underline{M}\rangle,\langle 2,M'\rangle) \wedge t(\langle 2,M'\rangle,\langle 6,M\rangle)] \Rightarrow I(\langle 1,\underline{M}\rangle,\langle 6,M\rangle)$$

$$[I(\langle 1,\underline{M}\rangle,\langle 5,M'\rangle) \wedge t(\langle 5,M'\rangle,\langle 6,M\rangle)] \Rightarrow I(\langle 1,\underline{M}\rangle,\langle 6,M\rangle)$$

c'est-à-dire

$$\forall L',L \in \mathfrak{L}.\ [I(\langle 1,\underline{M}\rangle,\langle L',M'\rangle) \wedge t(\langle L',M'\rangle,\langle L,M\rangle)] \Rightarrow I(\langle 1,\underline{M}\rangle,\langle L,M\rangle)$$

qui est équivalent à :

$$\forall \underline{\Delta},\Delta',\Delta.\ [\varepsilon(\underline{\Delta}) \wedge I(\underline{\Delta},\Delta') \wedge t(\Delta',\Delta)] \Rightarrow I(\underline{\Delta},\Delta)$$

- La dernière condition de vérification est

$$P_6(\underline{x},\underline{y},\bar{x},\bar{y},\bar{q}) \Rightarrow [\underline{x}=\bar{q}\times\underline{y}+\bar{x} \wedge \bar{x}<\underline{y} \wedge \bar{y}=\underline{y}]$$

c'est-à-dire

$$I(\langle 1,\underline{M}\rangle,\langle 6,\bar{M}\rangle) \Rightarrow \Psi(\langle 1,\underline{M}\rangle,\langle 6,\bar{M}\rangle)$$

soit

$$\forall \underline{\Delta},\bar{\Delta}.\ [\varepsilon(\underline{\Delta}) \wedge I(\underline{\Delta},\bar{\Delta})] \Rightarrow \Psi(\underline{\Delta},\bar{\Delta})$$

Ainsi, nous avons montré que cette méthode de preuve consiste essentiellement à découvrir un invariant I et à prouver que :

$$\begin{array}{l}[\ \forall \underline{s} \in S.\ \varepsilon(\underline{s}) \Rightarrow I(\underline{s},\underline{s}) \\ \quad \wedge \\ \quad (\forall \underline{s},s',s \in S.\ [\varepsilon(\underline{s}) \wedge I(\underline{s},s') \wedge t(s',s)] \Rightarrow I(\underline{s},s)) \\ \quad \wedge \\ \quad (\forall \underline{s},\bar{s} \in S.\ [\ \varepsilon(\underline{s}) \wedge I(\underline{s},\bar{s})] \Rightarrow \psi(\underline{s},\bar{s}))\ ]\end{array}$$

pour en déduire

$$\forall p \in \Sigma\langle S,A,t,\varepsilon\rangle,\ i \in |p|.\ \psi(p_0,p_i)$$

□

Le principe d'induction de base est

Théorème 4.2.1.1 ∿1

$$\begin{array}{l}[\exists I \in (S^2 \to \{tt,ff\}).\ \forall \underline{s},s,\bar{s} \in S,\ a \in A. \\ \\ (\text{-}\mathcal{I}.\varepsilon) \qquad \varepsilon(\underline{s}) \Rightarrow I(\underline{s},\underline{s}) \\ \qquad\qquad \wedge \\ (\text{-}\mathcal{I}.i) \qquad [\exists s' \in S.\ \varepsilon(\underline{s}) \wedge I(\underline{s},s') \wedge t_a(s',s)] \Rightarrow I(\underline{s},s) \qquad (\text{-}\mathcal{I}) \\ \qquad\qquad \wedge \\ (\text{-}\mathcal{I}.s) \qquad [\varepsilon(\underline{s}) \wedge I(\underline{s},\bar{s})] \Rightarrow \psi(\underline{s},\bar{s})\ ] \\ \qquad\qquad \Leftrightarrow \\ [\forall p \in \Sigma\langle S,A,t,\varepsilon\rangle,\ i \in |p|.\ \psi(p_0,p_i)]\end{array}$$

Démonstration

($\Rightarrow$) Pour la preuve de correction, soit $p \in \Sigma\langle S,A,t,\varepsilon\rangle$. Démontrons par récurrence sur $i \in |p|$ que $I(p_0,p_i)$. Pour $i=0$, nous avons $\varepsilon(p_0)$ et donc $I(p_0,p_0)$ d'après $(\text{-}\mathcal{I}.\varepsilon)$. Si $I(p_0,p_i)$ et $i+1 \in |p|$ alors $t_{p_i}(p_i,p_{i+1})$ et $(\text{-}\mathcal{I}.i)$ impliquent $I(p_0,p_{i+1})$. Nous déduisons de $(\text{-}\mathcal{I}.s)$ que $\forall i \in |p|.\ \psi(p_0,p_i)$.

($\Leftarrow$) La preuve de complétude sémantique est également très simple en choisissant $I(\underline{s},s) = [\exists p \in \Sigma\langle S,A,t,\varepsilon\rangle,\ i \in |p|.\ (p_0 = \underline{s} \wedge p_i = s)]$. $(\text{-}\mathcal{I}.\varepsilon)$ et $(\text{-}\mathcal{I}.i)$ dérivent

alors de la définition de $\Sigma\langle S,A,t,\varepsilon\rangle$. (-Y.5) dérive de l'hypothèse $\forall p \in \Sigma\langle S,A,t,\varepsilon\rangle, i \in |p| . \psi(p_0,p_i)$.

□

Remarque 4.2.1.1-2 (Invariance conditionnelle)

Par soucis de simplicité, nous donnons uniquement les principes d'induction pour l'invariance car la généralisation à l'invariance conditionnelle est triviale. Par exemple le principe d'induction (-Y) se généralise immédiatement en :

$$
\begin{array}{l}
[\exists I \in (S\times S \to \{tt,ff\}) . \forall \underline{s},s,\bar{s} \in S, a \in A . \\
\qquad \varepsilon(\underline{s}) \Rightarrow I(\underline{s},\underline{s}) \\
\qquad \wedge \\
\qquad [\exists s' \in S . \varepsilon(\underline{s}) \wedge I(\underline{s},s') \wedge \phi(\underline{s},s') \wedge t_a(s',s)] \Rightarrow I(\underline{s},s) \\
\qquad \wedge \\
\qquad [\varepsilon(\underline{s}) \wedge I(\underline{s},\bar{s})] \Rightarrow \psi(\underline{s},\bar{s})] \\
\Leftrightarrow \\
[\forall p \in \Sigma\langle S,A,t,\varepsilon\rangle, i \in |p| . \ [\forall j \in i . \phi(p_0,p_j)] \Rightarrow \psi(p_0,p_i) \ ]
\end{array}
$$

La preuve de correction consiste essentiellement à démontrer que pour tout $p \in \Sigma\langle S,A,t,\varepsilon\rangle$ nous avons par récurrence sur $i \in |p|$, $[\forall j \in i . \phi(p_0,p_j)] \Rightarrow I(p_0,p_i)$. Pour la preuve de complétude nous choisissons $I(\underline{s},s) =$
$[\exists p \in \Sigma\langle S,A,t,\varepsilon\rangle, i \in |p| . (p_0 = \underline{s} \wedge \forall j \in i . \phi(p_0,p_j) \wedge p_i = s)]$.

□

4.2.1.2 Transformations de principes d'induction

Par transformation du principe d'induction de base $(\mathcal{I})$, nous obtenons un certain nombre de principes d'induction dérivés qui permettent de rendre compte des méthodes de preuve d'invariance existentes mais également d'en découvrir de nouvelles.

4.2.1.2.1 Transformation par distinction/confusion des états initiaux ou finaux

Observons que la définition de l'invariance

$$\forall p \in \Sigma\langle S, A, t, \varepsilon\rangle,\ i \in |p|.\ \psi(p_0, p_i)$$

est équivalente à

$$\forall p \in \Sigma\langle S, A, t, tt\rangle,\ i \in |p|.\ (\varepsilon(p_0) \Rightarrow \psi(p_0, p_i))$$

en posant $tt(\Delta) = tt$.

Par conséquent, nous obtenons un principe d'induction $(\mathcal{J})$ équivalent à $(\mathcal{I})$ en substituant tt à ε et $\psi'(\Delta, \Delta') = (\varepsilon(\Delta) \Rightarrow \psi(\Delta, \Delta'))$ à $\psi(\Delta, \Delta')$ dans $(\mathcal{J})$:

$$[\exists I \in (S^2 \to \{tt, ff\}).\ \forall \underline{\Delta}, \Delta, \bar{\Delta} \in S,\ a \in A.$$

$$\begin{array}{ll} (\mathcal{J}.\varepsilon) & I(\underline{\Delta}, \underline{\Delta}) \\ & \wedge \\ (\mathcal{J}.\mathcal{I}) & [\exists \Delta' \in S.\ I(\underline{\Delta}, \Delta') \wedge t_a(\Delta', \Delta)] \Rightarrow I(\underline{\Delta}, \Delta) \\ & \wedge \\ (\mathcal{J}.\mathcal{S}) & I(\underline{\Delta}, \bar{\Delta}) \Rightarrow \psi(\underline{\Delta}, \bar{\Delta})\ ] \end{array} \qquad (\mathcal{J})$$

$$\Longleftrightarrow$$

$$[\forall p \in \Sigma\langle S, A, t, tt\rangle,\ i \in |p|.\ \psi(p_0, p_i)]$$

Par symétrie avec les états initiaux, on peut être amené à distinguer des états finaux, en écrivant la définition de l'invariance sous la forme

$$\forall p \in \Sigma\langle S, A, t, \varepsilon\rangle, i \in |p| .\ (\sigma(p_i) \Rightarrow \psi(p_0, p_i))$$

Si nous substituons $\psi'(a,a') = [\sigma(a') \Rightarrow \psi(a,a')]$ à $\psi(a,a')$ dans $(-\mathcal{Y})$, nous obtenons $(-\mathcal{Y}-)$:

$$\begin{array}{lll} & [\exists I \in (S\times S \to \{tt, ff\}) .\ \forall \underline{a}, a, \bar{a} \in S,\ a \in A . & \\ (-\mathcal{Y}-.\varepsilon) & \quad \varepsilon(\underline{a}) \Rightarrow I(\underline{a}, \underline{a}) & \\ & \wedge & \\ (-\mathcal{Y}-.i) & \quad [\exists a' \in S .\ \varepsilon(\underline{a}) \wedge I(\underline{a}, a') \wedge t_a(a', a)] \Rightarrow I(\underline{a}, a) & (-\mathcal{Y}-) \\ & \wedge & \\ (-\mathcal{Y}-.\sigma) & \quad [\varepsilon(\underline{a}) \wedge I(\underline{a}, \bar{a}) \wedge \sigma(\bar{a})] \Rightarrow \psi(\underline{a}, \bar{a}) \ ] & \\ & \Longleftrightarrow & \\ & [\forall p \in \Sigma\langle S, A, t, tt\rangle, i \in |p| .\ (\ [\varepsilon(p_0) \wedge \sigma(p_i)] \Rightarrow \psi(p_0, p_i))\ ] & \end{array}$$

4.2.1.2.2 Transformation par déduction/prédiction

Pour l'affectation

```
4:
    X := X - Y ;
5:
```

la condition de vérification due à Floyd est "déductive" :

$$P_5(\underline{z}, \underline{y}, x, y, q) \Leftarrow [\exists x' .\ P_4(\underline{z}, \underline{y}, x', y, q) \wedge x = x' - y]$$

la plus forte post-condition déduite de la précondition doit entraîner la post-condition. La condition de vérification due à Hoare est "prédictive" :

$$P_4(\underline{z}, \underline{y}, x, y, q) \Rightarrow P_5(\underline{z}, \underline{y}, x - y, y, q)$$

La précondition doit entraîner la plus faible précondition prédite à partir de la post-condition. Ces deux conditions de vérification sont équivalentes.

Cette remarque se généralise comme suit :

La condition de vérification $(-\mathcal{Y}-.i)$

$$[\forall \underline{s}, s \in S,\ a \in A.\ [\exists s' \in S.\ \varepsilon(\underline{s}) \wedge I(\underline{s}, s') \wedge t_a(s', s)] \Rightarrow I(\underline{s}, s)$$

est équivalente à

$$[\forall \underline{s}, s', s \in S,\ a \in A.\ [\varepsilon(\underline{s}) \wedge I(\underline{s}, s')] \Rightarrow [t_a(s', s) \Rightarrow I(\underline{s}, s)]]$$

$$\Leftrightarrow [\forall \underline{s}, s' \in S,\ a \in A.\ [\varepsilon(\underline{s}) \wedge I(\underline{s}, s')] \Rightarrow [\forall s \in S.\ t_a(s', s) \Rightarrow I(\underline{s}, s)]]$$

$$\Leftrightarrow [\forall \underline{s}, s \in S,\ a \in A.\ [\varepsilon(\underline{s}) \wedge I(\underline{s}, s)] \Rightarrow \neg[\exists s' \in S.\ t_a(s, s') \wedge \neg I(\underline{s}, s')]$$

Alors à partir de $(-\mathcal{Y}-)$, nous dérivons le principe d'induction équivalent $(-\tilde{\mathcal{Y}}-)$:

$$[\exists I \in (S \times S \to \{tt, ff\}).\ \forall \underline{s}, s, \bar{s} \in S,\ a \in A.$$

$$(-\tilde{\mathcal{Y}}-.\varepsilon) \qquad \varepsilon(\underline{s}) \Rightarrow I(\underline{s}, \underline{s})$$

$$\wedge$$

$$(-\tilde{\mathcal{Y}}-.i) \qquad [\varepsilon(\underline{s}) \wedge I(\underline{s}, s)] \Rightarrow \neg[\exists s' \in S.\ t_a(s, s') \wedge \neg I(\underline{s}, s')] \qquad (-\tilde{\mathcal{Y}}-)$$

$$\wedge$$

$$(-\tilde{\mathcal{Y}}-.\sigma) \qquad [\varepsilon(\underline{s}) \wedge I(\underline{s}, \bar{s}) \wedge \sigma(\bar{s})] \Rightarrow \psi(\underline{s}, \bar{s})]$$

$$\Leftrightarrow$$

$$[\forall p \in \Sigma^{<S, A, t, tt>},\ i \in |p|.\ [\varepsilon(p_0) \wedge \sigma(p_i)] \Rightarrow \psi(p_0, p_i)]$$

4.2.1.2.3 Transformation par inversion

Définissons l'inverse p^{-1} d'une trace finie

$$p = s_0 \xrightarrow{a_0} s_1 \cdots s_{m-2} \xrightarrow{a_{m-2}} s_{m-1} = \langle m, s, a \rangle$$

comme étant :

$$p^{-1} = s_{m-1} \xrightarrow{a_{m-2}} s_{m-2} \cdots s_1 \xrightarrow{a_0} s_0 = \langle m, s', a' \rangle$$

où $\forall i \in m.\ s'_i = s_{m-i-1}$ et $\forall i \in (m-1).\ a'_i = a_{m-i-2}$

et l'inverse d'un ensemble Σ de traces comme étant l'ensemble Σ^{-1} des inverses des traces de Σ :

$$\Sigma^{-1} = \{p^{-1} : p \in \Sigma\}$$

Démontrer pour tout p qu'on a la propriété d'invariance

$$\forall i \in |p| . [[\varepsilon(p_0) \wedge \sigma(p_i)] \Rightarrow \psi(p_0, p_i)]$$

est équivalent, d'après le théorème 4.1.3 ci-2, à la preuve que pour tous les préfixes finis $q \in \Sigma^{<\omega}$ des traces de Σ, nous avons :

$$[\varepsilon(q_0) \wedge \sigma(q_{|q|})] \Rightarrow \psi(q_0, q_{|q|})$$

en posant $\langle S, A, \Sigma^{<\omega} \rangle = \underset{\sim}{\text{Pref}}^{<\omega}(\langle S, A, \Sigma \rangle)$, ou bien encore en raisonnant sur les traces inverses :

$$\forall q \in (\Sigma^{<\omega})^{-1} . [[\sigma(q_0) \wedge \varepsilon(q_{|q|})] \Rightarrow \psi^{-1}(q_0, q_{|q|})]$$

Dans le cas particulier où tout état p_i d'une trace p de Σ satisfaisant ε est origine du suffixe de la trace de p commençant à p_i :

$$\forall p \in \Sigma, i \in |p| . [\varepsilon(p_i) \Rightarrow (p^{\geq i} \in \Sigma)]$$

la propriété ci-dessus est équivalente à la propriété d'invariance suivante qui porte sur les inverses de préfixes finis de Σ :

$$\forall p \in (\Sigma^{<\omega})^{-1}, i \in |p| . [[\sigma(p_0) \wedge \varepsilon(p_i)] \Rightarrow \psi^{-1}(p_0, p_i)]$$

Dans ce cas, toute preuve d'invariance portant sur $S, A, \Sigma, \varepsilon, \sigma, \psi$ peut se faire en raisonnant respectivement sur $S, A, (\Sigma^{<\omega})^{-1}, \sigma, \varepsilon, \psi^{-1}$.

Dans le cas particulier où l'ensemble de traces est engendré par un système de transition $\langle S, A, t, \varepsilon \rangle$, nous obtenons :

$$[\forall \underline{s}, \bar{s} \in S . [\sigma(\underline{s}) \wedge (t^*(\underline{s}, \bar{s}))^{-1} \wedge \varepsilon(\bar{s})] \Rightarrow \psi^{-1}(\underline{s}, \bar{s})]$$

qui peut s'écrire puisque $(t^*(\underline{s},\bar{s}))^{-1} = t^{-1}{}^*(\underline{s},\bar{s})$,

$$[\forall \underline{s},\bar{s} \in S.\ [\sigma(\underline{s}) \wedge t^{-1}{}^*(\underline{s},\bar{s}) \wedge \varepsilon(\bar{s})] \Rightarrow \psi^{-1}(\underline{s},\bar{s})]$$

Ceci peut se démontrer en utilisant le principe d'induction $(\text{-}\mathcal{Y}\text{-})$ où $\varepsilon, t, \sigma, \psi$ sont respectivement choisis comme $\sigma, t^{-1}, \varepsilon, \psi^{-1}$ d'où les conditions de vérification suivantes :

$$\begin{aligned}
&[\exists I \in (S\times S \to \{tt,ff\}).\ \forall \underline{s}, s, \bar{s} \in S,\ a \in A.\\
&\quad \sigma(\underline{s}) \Rightarrow I(\underline{s},\underline{s})\\
&\quad \wedge\ [\exists s' \in S.\ \sigma(\underline{s}) \wedge I(\underline{s},s') \wedge t_a^{-1}(s',s)] \Rightarrow I(\underline{s},s)\\
&\quad \wedge\ [\sigma(\underline{s}) \wedge I(\underline{s},\bar{s}) \wedge \varepsilon(\bar{s})] \Rightarrow \psi^{-1}(\underline{s},\bar{s})\ ]
\end{aligned}$$

Soit J l'inverse I^{-1} de I. Ces conditions de vérification sont équivalentes à :

$$\begin{aligned}
&[\exists J \in (S\times S \to \{tt,ff\}).\ \forall \underline{s}, s, \bar{s} \in S,\ a \in A.\\
&\quad \sigma(\underline{s}) \Rightarrow J(\underline{s},\underline{s})\\
&\quad \wedge\ [\exists s' \in S.\ t_a^{-1}(s',s) \wedge J(s',\underline{s}) \wedge \sigma(\underline{s})] \Rightarrow J(s,\underline{s})\\
&\quad \wedge\ [\varepsilon(\bar{s}) \wedge J(\bar{s},\underline{s}) \wedge \sigma(\underline{s})] \Rightarrow \psi^{-1}(\underline{s},\bar{s})\ ]
\end{aligned}$$

Renommant les variables muettes $\underline{s},\bar{s}$ respectivement en $\bar{s},\underline{s}$, nous obtenons

$$\begin{aligned}
&[\exists J \in (S\times S \to \{tt,ff\}).\ \forall \underline{s}, s, \bar{s} \in S,\ a \in A.\\
&\quad \sigma(\bar{s}) \Rightarrow J(\bar{s},\bar{s})\\
&\quad \wedge\ [\exists s' \in S.\ t_a^{-1}(s',s) \wedge J(s',\bar{s}) \wedge \sigma(\bar{s})] \Rightarrow J(s,\bar{s})\\
&\quad \wedge\ [\varepsilon(\underline{s}) \wedge J(\underline{s},\bar{s}) \wedge \sigma(\bar{s})] \Rightarrow \psi^{-1}(\bar{s},\underline{s})\ ]
\end{aligned}$$

Utilisant la définition des relations inverses, nous venons de démontrer que le principe d'induction $(\text{-}\mathcal{Y}\text{-}^{-1})$ est correct et sémantiquement complet :

$$\begin{aligned}
&[\exists J \in (S\times S \to \{tt,ff\}).\ \forall \underline{s}, s, \bar{s} \in S,\ a \in A.\\
&(\text{-}\mathcal{Y}\text{-}^{-1}.\sigma) \qquad \sigma(\bar{s}) \Rightarrow J(\bar{s},\bar{s})\\
&(\text{-}\mathcal{Y}\text{-}^{-1}.i) \qquad \wedge\ [\exists s' \in S.\ t_a(s,s') \wedge J(s',\bar{s}) \wedge \sigma(\bar{s})] \Rightarrow J(s,\bar{s}) \qquad (\text{-}\mathcal{Y}\text{-}^{-1})\\
&(\text{-}\mathcal{Y}\text{-}^{-1}.\varepsilon) \qquad \wedge\ [\varepsilon(\underline{s}) \wedge J(\underline{s},\bar{s}) \wedge \sigma(\bar{s})] \Rightarrow \psi(\underline{s},\bar{s})\ ]\\
&\qquad \Longleftrightarrow\\
&[\forall p \in \Sigma^{<S,A,t,tt>}, i \in |p|.\ [\varepsilon(p_0) \wedge \sigma(p_i)] \Rightarrow \psi(p_0,p_i)\ ]
\end{aligned}$$

Ce principe d'induction est à la base de la méthode de Morris-Wegbreit [77] dite "subgoal induction".

Plus généralement, la transformation notée -1 consiste à remplacer la preuve :

$$P(s,A,t,\varepsilon,\sigma,\psi) \Leftrightarrow \exists I.\, Cv[\![s,A,t,\varepsilon,\sigma,\psi]\!](I)$$

par une preuve :

$$P'(s,A,t^{-1},\sigma,\varepsilon,\psi^{-1}) \Leftrightarrow \exists J.\, Cv'[\![s,A,t^{-1},\sigma,\varepsilon,\psi^{-1}]\!](J)$$

(où $J = I^{-1}$) quand :

$$P(s,A,t,\varepsilon,\sigma,\psi) \Leftrightarrow P'(s,A,t^{-1},\sigma,\varepsilon,\psi^{-1})$$

4.2.1.2.4 Transformation contrapositive

Utilisant la propriété que $\neg\neg J = J$, nous pouvons réécrire les conditions de vérification $(\mathcal{Y}^{-1})$ comme suit :

$$\begin{aligned}
&[\exists J \in (S\times S \to \{tt, ff\}).\ \forall \underline{s}, s, \bar{s} \in S, a \in A.\\
&\quad [\varepsilon(\underline{s}) \wedge \neg\neg J(\underline{s},\bar{s}) \wedge \sigma(\bar{s})] \Rightarrow \psi(\underline{s},\bar{s})\\
&\quad \wedge\\
&\quad [\exists s' \in S.\ t_a(s,s') \wedge \neg\neg J(s',\bar{s}) \wedge \sigma(\bar{s})] \Rightarrow \neg\neg J(s,\bar{s})\\
&\quad \wedge\\
&\quad \sigma(\bar{s}) \Rightarrow \neg\neg J(\bar{s},\bar{s})\ ]
\end{aligned}$$

Posons $\bar{J} = \neg J$ et utilisons le fait que $P \Rightarrow Q$ si et seulement si $\neg Q \Rightarrow \neg P$ dans la condition ci-dessus. Nous obtenons la condition équivalente suivante :

$$\begin{aligned}
&[\exists \bar{J} \in (S\times S \to \{tt, ff\}).\ \forall \underline{s}, s', \bar{s} \in S, a \in A.\\
&\quad [\varepsilon(\underline{s}) \wedge \neg\psi(\underline{s},\bar{s}) \wedge \sigma(\bar{s})] \Rightarrow \bar{J}(\underline{s},\bar{s})\\
&\quad \wedge\\
&\quad [\exists s \in S.\ \bar{J}(s,\bar{s}) \wedge t_a(s,s') \wedge \sigma(\bar{s})] \Rightarrow \bar{J}(s',\bar{s})\\
&\quad \wedge\\
&\quad \sigma(\bar{s}) \Rightarrow \neg\bar{J}(\bar{s},\bar{s})]
\end{aligned}$$

à partir de laquelle nous concluons que le principe d'induction $(\overline{\mathcal{Y}^{-1}})$ est équivalent à $(\mathcal{Y}^{-1})$ et est donc correct et sémantiquement complet :

$$[\,\exists \bar{J} \in (S\times S \to \{tt, ff\}).\ \forall \underline{a}, a, \bar{a} \in S,\ \alpha \in A.$$

$$(\overline{\neg J^{-1}}.\varepsilon) \qquad [\varepsilon(\underline{a}) \wedge \neg\psi(\underline{a},\bar{a}) \wedge \sigma(\bar{a})] \Rightarrow \bar{J}(\underline{a},\bar{a})$$

$$(\overline{\neg J^{-1}}.i) \qquad \wedge \quad [\exists a' \in S.\ \bar{J}(a',\bar{a}) \wedge t_\alpha(a',a) \wedge \sigma(\bar{a})] \Rightarrow \bar{J}(a,\bar{a}) \qquad (\overline{\neg J^{-1}})$$

$$(\overline{\neg J^{-1}}.s) \qquad \wedge \quad \sigma(\bar{a}) \Rightarrow \neg \bar{J}(\bar{a},\bar{a})]$$

$$\Longleftrightarrow$$

$$[\forall p \in \Sigma^{<}\langle S, A, t, tt\rangle, i \in |p|.\ [\varepsilon(p_0) \wedge \sigma(p_i)] \Rightarrow \psi(p_0, p_i)\,]$$

Nous obtenons ainsi une nouvelle méthode de preuve par l'absurde.

Exemple 4.2.1.2.4-1

En utilisant cette méthode pour démontrer la correction partielle du programme 4.2.1.2-1, nous procédons comme suit :

Cette méthode de preuve étant contrapositive, les invariants locaux décrivent ce qui n'arrivera pas pendant l'exécution du programme :

$$P_1(x,y,q,\bar{x},\bar{q}) = [(x = \bar{q}\times y + \bar{x}) \Rightarrow (\bar{x} \geq y)]$$

$$P_2(x,y,q,\bar{x},\bar{q}) = [(x = (\bar{q}-q)\times y + \bar{x}) \Rightarrow (\bar{x} \geq y)]$$

$$P_3(x,y,q,\bar{x},\bar{q}) = [(x = (\bar{q}-q)\times y + \bar{x}) \Rightarrow (\bar{x} \geq y)]$$

$$P_4(x,y,q,\bar{x},\bar{q}) = [(x = (\bar{q}-q+1)\times y + \bar{x}) \Rightarrow (\bar{x} \geq y)]$$

$$P_5(x,y,q,\bar{x},\bar{q}) = [(x = (\bar{q}-q)\times y + \bar{x}) \Rightarrow (\bar{x} \geq y)]$$

$$P_6(x,y,q,\bar{x},\bar{q}) = [((x = (\bar{q}-q)\times y + \bar{x}) \Rightarrow (\bar{x} \geq y)) \wedge (x < y)]$$

Soient x,y,q et $\bar{x},\bar{y},\bar{q}$ les valeurs initiales et finales des variables X,Y,Q du programme. Si le programme n'était pas partiellement correct, alors on aurait $\neg[x = \bar{q}\times y + \bar{x} \wedge \bar{x} < y]$ et donc $P_1(x,y,q,\bar{x},\bar{q})$ serait vrai d'après la première condition de vérification :

$$P_1(x,y,q,\bar{x},\bar{y}) \Leftarrow \neg[x = \bar{q}\times y + \bar{x} \wedge \bar{x} < y]$$

Puis par induction sur le nombre n de pas de calcul durant l'exécution du programme, l'hypothèse que $P_1(x,y,q,\bar{x},\bar{q})$ est vrai et les conditions

de vérification impliquent que les $P_i(x,y,q,\bar{x},\bar{q})$, $i = 1,\dots,6$ sont vrais :

$$P_2(x,y,q,\bar{x},\bar{q}) \Leftarrow [\exists q'.\ P_1(x,y,q',\bar{x},\bar{q}) \wedge q = 0]$$

$$P_3(x,y,q,\bar{x},\bar{q}) \Leftarrow [(P_2(x,y,q,\bar{x},\bar{q}) \vee P_5(x,y,q,\bar{x},\bar{q})) \wedge (x \leq y)]$$

$$P_4(x,y,q,\bar{x},\bar{q}) \Leftarrow [\exists q'.\ P_3(x,y,q',\bar{x},\bar{q}) \wedge q = q'+1]$$

$$P_5(x,y,q,\bar{x},\bar{q}) \Leftarrow [\exists x'.\ P_4(x',y,q,\bar{x},\bar{q}) \wedge x = x'-y]$$

$$P_6(x,y,q,\bar{x},\bar{q}) \Leftarrow [(P_2(x,y,q,\bar{x},\bar{q}) \vee P_5(x,y,q,\bar{x},\bar{q})) \wedge (x > y)]$$

Si nous supposons que l'exécution du programme se termine, alors la dernière condition de vérification :

$$\neg P_6(x,y,q,\bar{x},\bar{q}) \Leftarrow [x = \bar{x} \wedge q = \bar{q}]$$

implique que $P_6(x,y,q,\bar{x},\bar{q})$ n'est pas vrai, donc contradiction. Nous avons donc montré par l'absurde que le programme est partiellement correct.

□

4.2.1.2.5 Transformation relation/assertion

Dans le cas d'une assertion invariante

$$\forall p \in \Sigma^{<S,A,t,tt>}, i \in |p| .\ [\varepsilon(p_0) \wedge \sigma(p_i)] \Rightarrow \psi(p_i)$$

l'invariant I utilisé dans le principe d'induction $(\text{-}I\text{-})$ peut également être unaire. Nous obtenons le principe d'induction :

$$[\exists i \in (S \to \{tt, ff\}) .\ \forall \underline{s}, s, \bar{s} \in S, a \in A .$$

$$(\text{-}i\text{-}.\varepsilon) \qquad \varepsilon(\underline{s}) \Rightarrow i(\underline{s})$$

$$\wedge$$

$$(\text{-}i\text{-}.i) \qquad [\exists s' \in S .\ i(s') \wedge t_a(s',s)] \Rightarrow i(s)$$

$$\wedge$$

$$(\text{-}i\text{-}.\sigma) \qquad [i(\bar{s}) \wedge \sigma(\bar{s})] \Rightarrow \psi(\bar{s})] \qquad (\text{-}i\text{-})$$

$$\Leftrightarrow$$

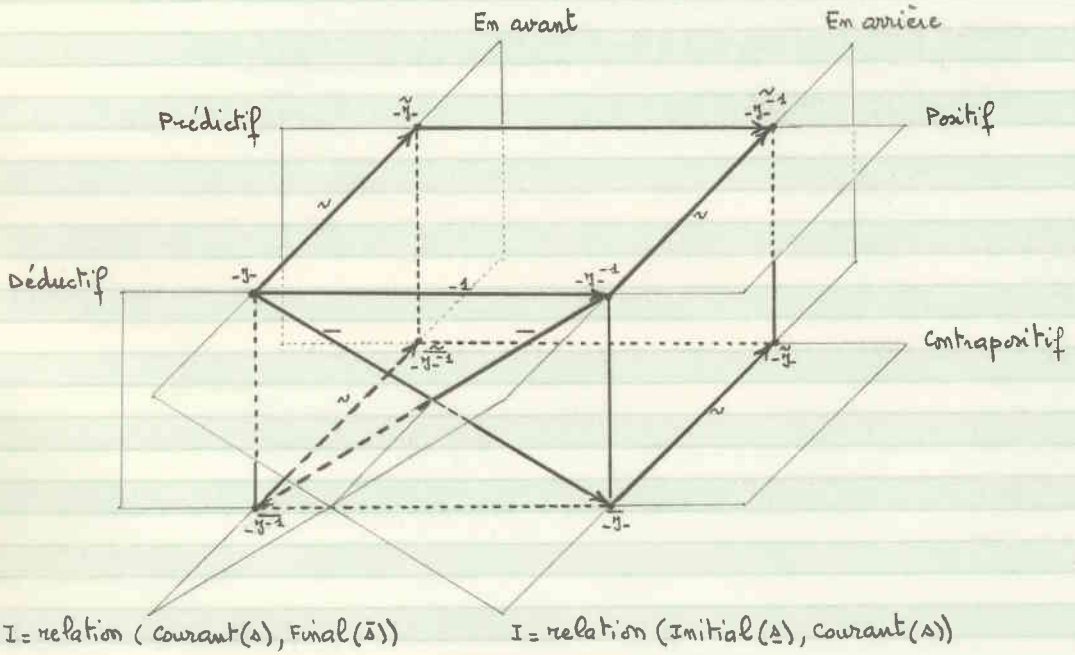
$$[\forall p \in \Sigma^{<S,A,t,tt>}, i \in |p| .\ [\varepsilon(p_0) \wedge \sigma(p_i)] \Rightarrow \psi(p_i)]$$

qui est à la base de la méthode de Floyd-Naur.

4.2.1.3 Principes d'induction dérivés par transformations

En partant du principe d'induction de base (-J-) qui est correct et sémantiquement complet, nous obtenons des principes d'induction dérivés en utilisant les trois transformations ~, -1 et ⁻ qui conservent la correction et la complétude sémantique, ce qui évite d'avoir à refaire les démonstrations pour chaque principe d'induction dérivé.

Nous pouvons représenter ces dérivations comme suit :



La liste des principes d'induction dérivés est la suivante :

$$[\forall p \in \Sigma\langle S,A,t,\mathbb{tt}\rangle, i \in |p|.\ [\varepsilon(p_0) \wedge \sigma(p_i)] \Rightarrow \psi(p_0,p_i)]$$

$[\exists J \in (S\times S \to \{\mathbb{tt},\mathbb{ff}\}).\ \forall \underline{s},s,\bar{s} \in S,\ a \in A.$

$(\tilde{\mathcal{J}}\text{-})$ $\quad \varepsilon(\underline{s}) \Rightarrow I(\underline{s},\underline{s})$

$[\varepsilon(\underline{s}) \wedge I(\underline{s},s)] \Rightarrow \neg[\exists s' \in S.\ t_a(s,s') \wedge \neg I(\underline{s},s')]$

$[\varepsilon(\underline{s}) \wedge I(\underline{s},\bar{s}) \wedge \sigma(\bar{s})] \Rightarrow \psi(\underline{s},\bar{s})]$

$\sigma(\bar{s}) \Rightarrow J(\bar{s},\bar{s})$ $\quad (\tilde{\mathcal{J}}^{-1}\text{-})$

$[J(s,\bar{s}) \wedge \sigma(\bar{s})] \Rightarrow \neg[\exists s' \in S.\ \neg J(s',\bar{s}) \wedge t_a(s',s)]$

$[\varepsilon(\underline{s}) \wedge J(\underline{s},\bar{s}) \wedge \sigma(\bar{s})] \Rightarrow \psi(\underline{s},\bar{s})]$

$\Longleftrightarrow$

$[\exists I \in (S\times S \to \{\mathbb{tt},\mathbb{ff}\}).\ \forall \underline{s},s,\bar{s} \in S,\ a \in A.$

$(\mathcal{J}\text{-})$ $\quad \varepsilon(\underline{s}) \Rightarrow I(\underline{s},\underline{s})$

$\wedge\ [\exists s' \in S.\ \varepsilon(\underline{s}) \wedge I(\underline{s},s') \wedge t_a(s',s)] \Rightarrow I(\underline{s},s)$

$\wedge\ [\varepsilon(\underline{s}) \wedge I(\underline{s},\bar{s}) \wedge \sigma(\bar{s})] \Rightarrow \psi(\underline{s},\bar{s})]$

$[\exists J \in (S\times S \to \{\mathbb{tt},\mathbb{ff}\}).\ \forall \underline{s},s,\bar{s} \in S,\ a \in A.$

$\sigma(\bar{s}) \Rightarrow J(\bar{s},\bar{s})$ $\quad (\mathcal{J}^{-1}\text{-})$

$\wedge\ [\exists s' \in S.\ t_a(s,\ \ \wedge J(s',\bar{s}) \wedge \sigma(\bar{s})] \Rightarrow J(s,\bar{s})$

$\wedge\ [\varepsilon(\underline{s}) \wedge J(\underline{s},\bar{s}) \wedge \sigma(\bar{s})] \Rightarrow \psi(\underline{s},\bar{s})]$

$\Longleftrightarrow$

$[\exists \bar{J} \in (S\times S \to \{\mathbb{tt},\mathbb{ff}\}).\ \forall \underline{s},s,\bar{s} \in S,\ a \in A.$

$(\widetilde{\overline{\mathcal{J}^{-1}}}\text{-})$ $\quad [\varepsilon(\underline{s}) \wedge \neg\psi(\underline{s},\bar{s}) \wedge \sigma(\bar{s})] \Rightarrow \bar{J}(\underline{s},\bar{s})$

$\wedge\ [\bar{J}(s,\bar{s}) \wedge \sigma(\bar{s})] \Rightarrow \neg[\exists s' \in S.\ t_a(s,s') \wedge \neg\bar{J}(s',\bar{s})]$

$\wedge\ \sigma(\bar{s}) \Rightarrow \neg\bar{J}(\bar{s},\bar{s})]$

$[\exists \bar{I} \in (S\times S \to \{\mathbb{tt},\mathbb{ff}\}).\ \forall \underline{s},s,\bar{s} \in S,\ a \in A.$

$[\varepsilon(\underline{s}) \wedge \neg\psi(\underline{s},\bar{s}) \wedge \sigma(\bar{s})] \Rightarrow \bar{I}(\underline{s},\bar{s})$ $\quad (\tilde{\overline{\mathcal{J}}}\text{-})$

$[\varepsilon(\underline{s}) \wedge \bar{I}(\underline{s},s)] \Rightarrow \neg[\exists s' \in S.\ \neg\bar{I}(\underline{s},s') \wedge t_a(s',s)]$

$\varepsilon(\underline{s}) \Rightarrow \neg\bar{I}(\underline{s},\underline{s})]$

$\Longleftrightarrow$

$[\exists \bar{J} \in (S\times S \to \{\mathbb{tt},\mathbb{ff}\}).\ \forall \underline{s},s,\bar{s} \in S,\ a \in A.$

$(\overline{\mathcal{J}^{-1}}\text{-})$ $\quad [\varepsilon(\underline{s}) \wedge \neg\psi(\underline{s},\bar{s}) \wedge \sigma(\bar{s})] \Rightarrow \bar{J}(\underline{s},\bar{s})$

$[\exists s' \in S.\ \bar{J}(s',\bar{s}) \wedge t_a(s',s) \wedge \sigma(\bar{s})] \Rightarrow \bar{J}(s,\bar{s})$

$\sigma(\bar{s}) \Rightarrow \neg\bar{J}(\bar{s},\bar{s})]$

$[\exists \bar{I} \in (S\times S \to \{\mathbb{tt},\mathbb{ff}\}).\ \forall \underline{s},s,\bar{s} \in S,\ a \in A.$

$[\varepsilon(\underline{s}) \wedge \neg\psi(\underline{s},\bar{s}) \wedge \sigma(\bar{s})] \Rightarrow \bar{I}(\underline{s},\bar{s})$ $\quad (\bar{\mathcal{J}}\text{-})$

$\wedge\ [\exists s' \in S.\ \varepsilon(\underline{s}) \wedge t_a(s,s') \wedge \bar{I}(\underline{s},s')] \Rightarrow \bar{I}(\underline{s},s)$

$\wedge\ \varepsilon(\underline{s}) \Rightarrow \neg\bar{I}(\underline{s},\underline{s})]$

Des applications supplémentaires des transformations $\sim$, $^{-1}$ et $^{-}$ aux principes d'induction ci-dessus ne donnent pas de nouveaux principes d'induction. Plus généralement, le diagramme suivant commute

La transformation relation/assertion conduit aux principes d'induction suivants:

$$\forall p \in \Sigma^{<S,A,t,\mathbb{B}>}, i \in |p|.\ [\varepsilon(p_0) \wedge \sigma(p_i)] \Rightarrow \psi(p_i)$$

$$[\exists i \in (S \to \{\mathbb{B}, \mathbb{B}\!\!\mathbb{B}\}).\ \forall \underline{s}, s, \bar{s} \in S,\ a \in A.$$

$$(\text{-}i\text{-}) \qquad \varepsilon(\underline{s}) \Rightarrow i(\underline{s})$$
$$\wedge\ [\exists s' \in S.\ i(s') \wedge t_a(s', s)] \Rightarrow i(s)$$
$$\wedge\ [i(\bar{s}) \wedge \sigma(\bar{s})] \Rightarrow \psi(\bar{s})]$$

$$\Longleftrightarrow$$

$$[\exists i \in (S \to \{\mathbb{B}, \mathbb{B}\!\!\mathbb{B}\}).\ \forall \underline{s}, s, \bar{s} \in S,\ a \in A.$$

$$(\text{-}\tilde{i}\text{-}) \qquad \varepsilon(\underline{s}) \Rightarrow i(\underline{s})$$
$$\wedge\ i(s) \Rightarrow \neg[\exists s' \in S.\ t_a(s, s') \wedge \neg i(s')]$$
$$\wedge\ [i(\bar{s}) \wedge \sigma(\bar{s})] \Rightarrow \psi(\bar{s})]$$

$$\Longleftrightarrow$$

$$[\exists i \in (S \to \{\mathbb{B}, \mathbb{B}\!\!\mathbb{B}\}).\ \forall \underline{s}, s, \bar{s} \in S,\ a \in A.$$

$$(\text{-}\bar{i}\text{-}) \qquad [\neg\psi(\bar{s}) \wedge \sigma(\bar{s})] \Rightarrow i(\bar{s})$$
$$\wedge\ [\exists s' \in S.\ t_a(s, s') \wedge i(s')] \Rightarrow i(s)$$
$$\wedge\ \varepsilon(\underline{s}) \Rightarrow \neg i(\underline{s})]$$

$$\Longleftrightarrow$$

$$[\exists i \in (S \to \{\mathbb{B}, \mathbb{B}\!\!\mathbb{B}\}).\ \forall \underline{s}, s, \bar{s} \in S,\ a \in A.$$

$$(\text{-}\tilde{\bar{i}}\text{-}) \qquad [\neg\psi(\bar{s}) \wedge \sigma(\bar{s})] \Rightarrow i(\bar{s})$$

$$\forall p \in \Sigma^{<S,A,t,\mathbb{B}>}, i \in |p|.\ [\varepsilon(p_0) \wedge \sigma(p_i)] \Rightarrow \psi(p_0)$$

$$\Longleftrightarrow$$

$$[\exists i \in (S \to \{\mathbb{B}, \mathbb{B}\!\!\mathbb{B}\}).\ \forall \underline{s}, s, \bar{s} \in S,\ a \in A.$$

$$\sigma(\bar{s}) \Rightarrow i(\bar{s}) \qquad (\text{-}i^{-1}\text{-})$$
$$\wedge\ [\exists s' \in S.\ t_a(s, s') \wedge i(s')] \Rightarrow i(s)$$
$$\wedge\ [\varepsilon(\underline{s}) \wedge i(\underline{s})] \Rightarrow \psi(\underline{s})]$$

$$[\exists i \in (S \to \{\mathbb{B}, \mathbb{B}\!\!\mathbb{B}\}).\ \forall \underline{s}, s, \bar{s} \in S,\ a \in A.$$

$$\sigma(\bar{s}) \Rightarrow i(\bar{s}) \qquad (\text{-}\tilde{i^{-1}}\text{-})$$
$$\wedge\ i(s) \Rightarrow \neg[\exists s' \in S.\ \neg i(s') \wedge t_a(s', s)]$$
$$\wedge\ [\varepsilon(\underline{s}) \wedge i(\underline{s})] \Rightarrow \psi(\underline{s})]$$

$$[\exists i \in (S \to \{\mathbb{B}, \mathbb{B}\!\!\mathbb{B}\}).\ \forall \underline{s}, s, \bar{s} \in S,\ a \in A.$$

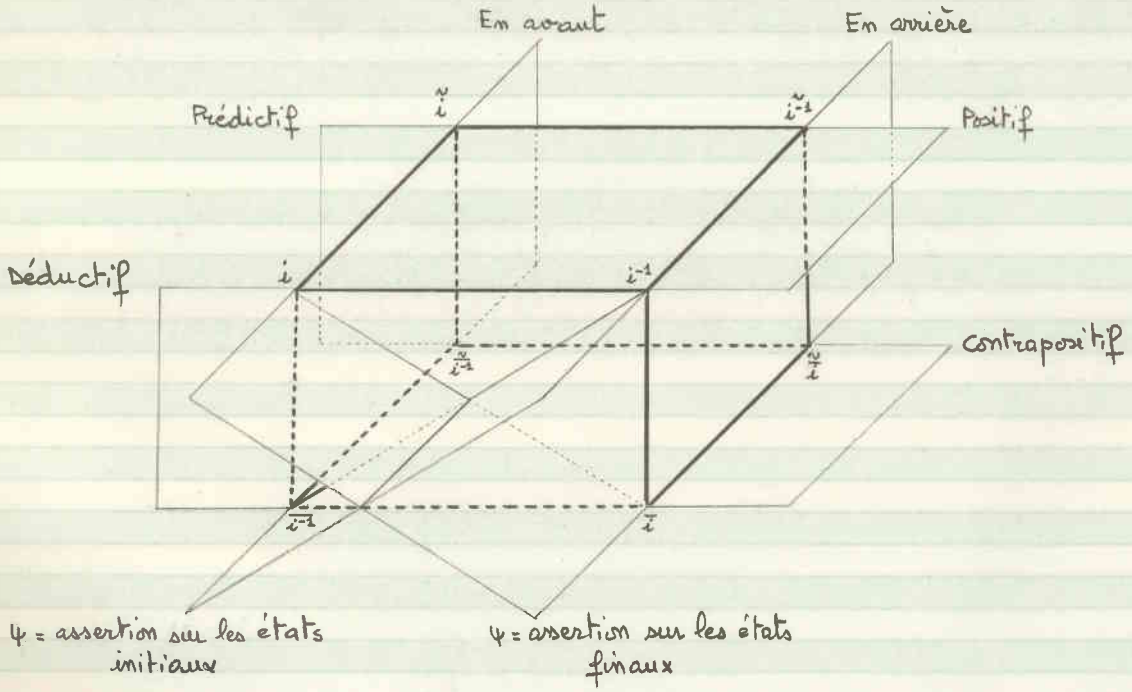
$$[\varepsilon(\underline{s}) \wedge \neg\psi(\underline{s})] \Rightarrow i(\underline{s}) \qquad (\text{-}\overline{i^{-1}}\text{-})$$
$$\wedge\ [\exists s' \in S.\ i(s') \wedge t_a(s', s)] \Rightarrow i(s)$$
$$\wedge\ \sigma(\bar{s}) \Rightarrow \neg i(\bar{s})]$$

$$[\exists i \in (S \to \{\mathbb{B}, \mathbb{B}\!\!\mathbb{B}\}).\ \forall \underline{s}, s, \bar{s} \in S,\ a \in A.$$

$$[\varepsilon(\underline{s}) \wedge \neg\psi(\underline{s})] \Rightarrow i(\underline{s}) \qquad (\text{-}\tilde{\overline{i^{-1}}}\text{-})$$
$$\wedge\ i(s) \Rightarrow \neg[\exists s' \in S.\ t_a(s, s') \wedge \neg i(s')]$$
$$\wedge\ \sigma(\bar{s}) \Rightarrow \neg i(\bar{s})]$$

Notons cependant que tous les principes d'induction relationnels sont équivalents tandis que pour les principes d'induction assertionnels, qui servent à démontrer différentes propositions, seuls ceux d'une même colonne sont équivalents.

Nous pouvons classer ces principes d'induction assertionnels comme suit :



En appliquant la transformation par distinction/confusion des états initiaux, des états finaux et des états initiaux et finaux, nous obtenons 64 autres principes d'induction.

Nous verrons dans la suite que chaque principe d'induction peut donner lieu à un grand nombre de méthodes de preuves.

4.2.1.4 Equivalence forte des principes d'induction dérivés

Tous les principes d'induction dans une même catégorie (c'est-à-dire permettant de démontrer une propriété d'invariance d'un certain type) sont corrects et sémantiquement complets. Par conséquent, s'il existe une preuve par une méthode $\mathcal{M}$ alors il existe également une preuve par toute autre méthode $\mathcal{M}'$ de la même catégorie.

En ce qui concerne les méthodes de Floyd [67] et Morris-Wegbreit [77], Dijkstra [82] a montré que toute preuve de correction partielle par "subgoal induction" peut se reécrire en une preuve par la méthode de Floyd (et conclut que la méthode de Morris-Wegbreit est inutile). Morris et Wegbreit avaient démontré la réciproque. Ce dernier résultat est généralisé dans Cousot-R [81] aux principes d'induction $(-\mathcal{Y}-)$ et $(-\mathcal{Y}-^{-1})$ comme suit :

Théorème 4.2.1.4~1

Les conditions .5, .i et .ε de $(-\mathcal{Y}-^{-1})$ et .ε, .i et .5 de $(-\mathcal{Y}-)$ sont équivalentes en définissant :

$$I(\underline{\lambda}, \lambda) = [\forall \bar{\lambda} \in S.\ (J(\lambda, \bar{\lambda}) \wedge 5(\bar{\lambda})) \Rightarrow \psi(\underline{\lambda}, \bar{\lambda})]$$

$$J(\lambda, \bar{\lambda}) = [\forall \underline{\lambda} \in S.\ (\varepsilon(\underline{\lambda}) \wedge I(\underline{\lambda}, \lambda)) \Rightarrow \psi(\underline{\lambda}, \bar{\lambda})]$$

et la remarque de Dijkstra se généralise de la même manière :

Théorème 4.2.1.4~2

Les conditions .5, .i et .ε de $(-\mathcal{Y}-^{-1})$ et .ε, .i et .5 de $(-\mathcal{Y}-)$ sont équivalentes en définissant :

$$I(\underline{\lambda}, \lambda) = [\varepsilon(\underline{\lambda}) \wedge \forall \bar{\lambda} \in S.\ (J(\lambda, \bar{\lambda}) \wedge 5(\bar{\lambda})) \Rightarrow J(\underline{\lambda}, \bar{\lambda})]$$

$$J(\lambda, \bar{\lambda}) = [5(\bar{\lambda}) \wedge \forall \underline{\lambda} \in S.\ (\varepsilon(\underline{\lambda}) \wedge I(\underline{\lambda}, \lambda)) \Rightarrow I(\underline{\lambda}, \bar{\lambda})]$$

Démonstration

- Si J satisfait les conditions .s, .i et .e de $(-\mathcal{Y}^{-1})$ alors nous avons :

$(-\mathcal{Y}.e)$ $\quad \varepsilon(\underline{a}) \Rightarrow [\varepsilon(\underline{a}) \wedge \forall \bar{a} \in S.((J(\underline{a},\bar{a}) \wedge s(\bar{a})) \Rightarrow J(\underline{a},\bar{a}))] \Rightarrow I(\underline{a},\underline{a})$

$(-\mathcal{Y}.i)$ $\quad [I(\underline{a},a') \wedge t_a(a',a)] \Rightarrow [\varepsilon(\underline{a}) \wedge \forall \bar{a} \in S.((J(a',\bar{a}) \wedge s(\bar{a})) \Rightarrow J(\underline{a},\bar{a}) \wedge t(a',a)] \Rightarrow$
$[\varepsilon(\underline{a}) \wedge \forall \bar{a} \in S.((J(a',\bar{a}) \wedge s(\bar{a})) \Rightarrow J(\underline{a},\bar{a})) \wedge \forall \bar{\bar{a}} \in S.((J(a,\bar{\bar{a}}) \wedge s(\bar{\bar{a}})) \Rightarrow J(a',\bar{\bar{a}}))]$ (car
$t_a(a',a) \Rightarrow [(J(a',\bar{\bar{a}}) \wedge s(\bar{\bar{a}})) \Rightarrow J(a',\bar{\bar{a}})]$ d'après $(-\mathcal{Y}^{-1}.i)$) $\Rightarrow$
$[\varepsilon(\underline{a}) \wedge \forall \bar{\bar{a}} \in S.((J(a,\bar{\bar{a}}) \wedge s(\bar{\bar{a}})) \Rightarrow J(\underline{a},\bar{\bar{a}}))] \Rightarrow I(\underline{a},a)$

$(-\mathcal{Y}.s)$ $\quad [I(\underline{a},\bar{a}) \wedge s(\bar{a})] \Rightarrow [\varepsilon(\underline{a}) \wedge \forall \bar{\bar{a}} \in S.((J(\bar{a},\bar{\bar{a}}) \wedge s(\bar{\bar{a}})) \Rightarrow J(\underline{a},\bar{\bar{a}}) \wedge J(\bar{a},\bar{\bar{a}}) \wedge s(\bar{a})]$
(d'après la définition de I et $(-\mathcal{Y}^{-1}.s)$) $\Rightarrow [\varepsilon(\underline{a}) \wedge J(\underline{a},\bar{a}) \wedge s(\bar{a}) \Rightarrow$
$\psi(\underline{a},\bar{a})$ (d'après $(-\mathcal{Y}^{-1}.e)$).

- Pour la réciproque, nous utilisons la transformation -1.

□

Ces résultats se généralisent aisément à tous les principes d'induction dérivés de $(-\mathcal{Y}-)$ par les transformations $\sim$, -1 et $-$:

Ceci signifie que si une preuve a été faite en utilisant une méthode M (c'est-à-dire que nous avons trouvé un invariant I satisfaisant les conditions de vérification CV) et M' est une autre méthode qui nécessite la découverte d'un invariant I' satisfaisant CV', alors nous pouvons définir formellement I' en fonction de I de sorte que CV(I) implique CV'(I') :

- Si $M' = \tilde{M}$ alors $I' = I$
- Si $M' = \bar{M}$ alors $I' = \neg I$
- Si $M' = M^{-1}$ et M est une méthode positive où I relie un état initial à un état courant alors $I'(a,\bar{a}) = [\forall \underline{a} \in S.\ (\varepsilon(\underline{a}) \wedge I(\underline{a},a)) \Rightarrow \psi(\underline{a},\bar{a})]$
- Si $M' = M^{-1}$ et M est une méthode contrapositive où I relie un état initial à un état courant alors $I'(a,\bar{a}) = [\exists \underline{a} \in S.\ \varepsilon(\underline{a}) \wedge I(\underline{a},a) \wedge \neg\psi(\underline{a},\bar{a})]$
- Si $M' = M^{-1}$ et M est une méthode positive où I relie un état courant à un état final alors $I'(\underline{a},a) = [\forall \bar{a} \in S.\ (I(a,\bar{a}) \wedge s(\bar{a})) \Rightarrow \psi(\underline{a},\bar{a})]$

- Si $M' = M^{-1}$ et M est une méthode contrapositive où I relie un état courant à un état final alors $I'(\underline{a}, a) = [\exists \bar{a} \in S.\ I(a, \bar{a}) \wedge \sigma(\bar{a}) \wedge \neg\psi(\underline{a}, \bar{a})]$

— De même quand ψ est une assertion, alors une preuve par un principe d'induction (M) peut également se faire par le principe d'induction (M) correspondant en choisissant

$I(\underline{a}, a) = [\varepsilon(\underline{a}) \wedge i(a)]$ (ou bien $I(a, \bar{a}) = [i(a) \wedge \sigma(\bar{a})]$)

La technique des variables auxiliaires (qui permet de reécrire une preuve d'invariance d'une relation par une méthode relationnelle en une preuve assertionnelle pour un programme transformé où les valeurs initiales des variables sont mémorisées dans des variables auxiliaires) se généralise comme suit :

Si $[\forall p \in \Sigma^{<S,A,t,\varepsilon>}, i \in |p|.\ \sigma(p_i) \Rightarrow \psi(p_0, p_i)]$ a été démontré par la méthode (M) utilisant l'invariant relationnel I, alors nous pourrons faire la même démonstration par la méthode (m) en posant :

- si I relie un état initial à un état courant,
  $S' = S \times S$, $A' = A$, $\varepsilon'(\langle \underline{a}, a \rangle) = [\varepsilon(\underline{a}) \wedge \underline{a} = a]$, $t'_a(\langle \underline{a}, a \rangle, \langle \underline{a}', a' \rangle) = [\underline{a} = \underline{a}' \wedge t_a(a, a')]$, $\sigma'(\langle \underline{a}, a \rangle) = \sigma(a)$ avec l'invariant unaire $i(\langle \underline{a}, a \rangle) = I(\underline{a}, a)$.
- si I relie un état courant à un état final,
  $S' = S \times S$, $A' = A$, $\sigma'(\langle a, \bar{a} \rangle) = [a = \bar{a} \wedge \sigma(\bar{a})]$, $t'_a(\langle a, \bar{a} \rangle, \langle a', \bar{a}' \rangle) = [t_a(a, a') \wedge \bar{a} = \bar{a}']$, $\varepsilon(\langle a, \bar{a} \rangle) = \varepsilon(a)$ avec l'invariant unaire $i(\langle a, \bar{a} \rangle) = I(a, \bar{a})$.

4.2.2 PRINCIPES D'INDUCTION POUR UNE SEMANTIQUE NON CLOSE FERMEE PAR FUSIONS

Dans le cas d'une sémantique $\langle S,A,\Sigma\rangle$ fermée par fusions (c'est-à-dire $Efus(\langle S,A,\Sigma\rangle) = \langle S,A,\Sigma\rangle$), l'invariance de ψ sous condition ϕ pour $\langle S,A,\Sigma\rangle$ est équivalente, d'après les théorèmes 4.1.3∨3.7 et 4.1.3∨4 à l'invariance de ψ sous condition ϕ pour $Rtran(\langle S,A,\Sigma\rangle)$. Ceci montre que dans ce cas il est toujours correct et sémantiquement complet de faire les preuves d'invariance en utilisant l'un quelconque des principes d'induction du paragraphe 4.2.1 pour le système de transition engendré par cette sémantique $\langle S,A,\Sigma\rangle$. C'est le cas en particulier (et d'après 4.1.3∨3.8 et 4.1.3∨3.9) pour le langage que nous avons considéré en 2.8.

4.2.3 PRINCIPES D'INDUCTION POUR UNE SEMANTIQUE (NON CLOSE ET) NON FERMEE PAR FUSIONS

4.2.3.1 Principes d'induction pour une sémantique non fermée par fusions définie par une condition sur les préfixes des traces engendées par un système de transition

Dans le cas d'une sémantique $\langle S,A,\Sigma\rangle$ non fermée par fusions, définie par une condition sur les préfixes des traces engendrées par un système de transition $\langle S,A,t,\varepsilon\rangle$:

$$\langle S,A,\Sigma\rangle = \langle S,A,\{p\in\Sigma' : \langle S,A,\Sigma'\rangle = Pref(\langle S,A,\Sigma\langle S,A,t,\varepsilon\rangle\rangle) \wedge Cp(p)\}\rangle$$

ψ est invariante sous condition ϕ pour $\langle S,A,\Sigma\langle S,A,t,\varepsilon\rangle\rangle$ si et seulement si ψ est invariante sous condition ϕ pour $Pref(\langle S,A,\Sigma\langle S,A,t,\varepsilon\rangle\rangle)$ (cf. 4.1.3∨3.1) et seulement si ψ est invariante sous condition ϕ pour $\langle S,A,\Sigma\rangle$ (cf. 4.1.3∨1).

Il est donc toujours correct de faire une preuve d'invariance relative à $\langle S, A, \Sigma \langle S, A, t, \varepsilon \rangle\rangle$ (pour laquelle nous pouvons toujours utiliser l'un quelconque des principes d'induction du paragraphes 4.2.1) pour conclure relativement à $\langle S, A, \Sigma \rangle$. Cette approche n'est pas toujours complète (comme en témoigne 4.1.3 n°1).

Pour être complet nous proposons d'utiliser le principe d'induction suivant (ou ses transformés comme en 4.2.1.2) qui utilise des variables auxiliaires (dans la preuve mais pas dans le programme) permettant de cumuler des histoires. Ce principe d'induction est directement inspiré d'un principe d'induction similaire proposé dans Cousot-Cousot [82] pour des preuves de fatalité. Il peut se motiver comme suit :

Pour démontrer que :

Ψ est invariante pour $\langle S, A, \Sigma \rangle$

il faut et il suffit en général de démontrer l'invariance d'une propriété plus forte :

$$\exists J \in (S \times S \to \{tt, ff\}). \; [J \text{ est invariante pour } \langle S, A, \Sigma \rangle \wedge J \Rightarrow \Psi]$$

D'après le théorème 4.1.3 n°2, il suffit de démontrer l'invariance pour les préfixes finis :

$$\exists J \in (S \times S \to \{tt, ff\}). \; [J \text{ est invariante pour } \underset{\sim}{Pref}^{<\omega} \langle S, A, \Sigma \rangle \wedge J \Rightarrow \Psi]$$

En écrivant $\forall p \in \underset{\sim}{Pref}^{<\omega} \langle S, A, \Sigma \rangle . Q(p)$ pour abréger $\forall p. [(\exists \Sigma'. [\langle S, A, \Sigma' \rangle = \underset{\sim}{Pref}^{<\omega} \langle S, A, \Sigma \rangle \wedge p \in \Sigma']) \Rightarrow Q(p)]$, ceci équivaut d'après 3.2.2 à :

$$\exists J \in (S \times S \to \{tt, ff\}). \; [\forall p \in \underset{\sim}{Pref}^{<\omega} \langle S, A, \Sigma \rangle, i \in |p| . J(p_0, p_i) \wedge J \Rightarrow \Psi]$$

En procédant par induction sur i, nous obtenons :

$$\exists J \in (S \times S \to \{tt, ff\}). \; \forall p \in \underset{\sim}{Pref}^{<\omega} \langle S, A, \Sigma \rangle .$$
$$J(p_0, p_0)$$
$$\wedge \; \forall i \in (|p| \setminus 0). \; J(p_0, p_{i-1}) \Rightarrow J(p_0, p_i)$$
$$\wedge \; J \Rightarrow \Psi$$

Il suffit de faire la preuve pour $i = |p|$ car si $p \in \underset{\sim}{Pref}^{<\omega}(\langle S, A, \Sigma \rangle)$ et $i \in |p|$ alors $p^{\leq i} \in \underset{\sim}{Pref}^{<\omega}(\langle S, A, \Sigma \rangle)$:

$$\exists J \in (S\times S \to \{tt, ff\}).\ \forall p \in \underset{\sim}{Pref}^{<\omega}(\langle S, A, \Sigma\rangle).$$

$$J(p_0, p_0)$$

$$\wedge\ [|p|>1 \wedge J(p_0, p_{|p|-1})] \Rightarrow J(p_0, p_{|p|})$$

$$\wedge\ J \Rightarrow \Psi$$

La condition $(p \in \underset{\sim}{Pref}^{<\omega}(\langle S,A,\Sigma\rangle) \wedge |p|>1)$ équivaut à $(\exists p' \in \underset{\sim}{Pref}^{<\omega}(\langle S,A,\Sigma\rangle), a\in A, s\in S.\ p = p' \xrightarrow{a} \langle s\rangle \ \wedge\ p \in \underset{\sim}{Pref}^{<\omega}(\langle S,A,\Sigma\rangle))$, ce qui donne :

$$\exists J \in (S\times S \to \{tt, ff\}).$$

$$\forall p \in \underset{\sim}{Pref}^{<\omega}(\langle S,A,\Sigma\rangle).\ J(p_0,p_0)$$

$$\wedge\ \forall p' \in \underset{\sim}{Pref}^{<\omega}(\langle S,A,\Sigma\rangle),\ a\in A,\ s\in S.$$

$$\left[J(p'_0, p'_{|p'|-1}) \ \wedge\ p' \xrightarrow{a} \langle s\rangle \in \underset{\sim}{Pref}^{<\omega}(\langle S,A,\Sigma\rangle)\right] \Rightarrow J(p'_0, s)$$

$$\wedge\ J \Rightarrow \Psi$$

En introduisant le système de transition $\langle S, A, t, \varepsilon\rangle$ engendré par $\langle S, A, \Sigma\rangle$, ceci peut s'écrire :

$$\exists J \in (S\times S \to \{tt, ff\}).$$

$$\forall \underline{s} \in S.\ \varepsilon(\underline{s}) \Rightarrow J(\underline{s}, \underline{s})$$

$$\wedge\ \forall p \in \underset{\sim}{Pref}^{<\omega}(\langle S,A,\Sigma\rangle),\ a\in A,\ s\in S.$$

$$\left[J(p_0, p_{|p|}) \wedge t_a(p_{|p|}, s) \wedge p \xrightarrow{a} \langle s\rangle \in \underset{\sim}{Pref}^{<\omega}(\langle S,A,\Sigma\rangle)\right] \Rightarrow J(p_0, s)$$

$$\wedge\ J \Rightarrow \Psi$$

ou encore

$$\exists J \in (S\times S \to \{tt, ff\}).$$

$$\forall \underline{s}, s', s, \bar{s} \in S,\ p', \bar{p} \in \underset{\sim}{Pref}^{<\omega}(\langle S,A,\Sigma\rangle),\ a\in A.$$

$$\varepsilon(\underline{s}) \Rightarrow \left[\exists p \in \underset{\sim}{Pref}^{<\omega}(\langle S,A,\Sigma\rangle).\ [p_0 = \underline{s} \wedge J(\underline{s},\underline{s})]\right]$$

$$\wedge\ \left[[p'_0 = \underline{s} \wedge J(\underline{s}, s')] \wedge t_a(s', s) \wedge [s' = p'_{|p'|} \wedge p' \xrightarrow{a} \langle s\rangle \in \underset{\sim}{Pref}^{<\omega}(\langle S,A,\Sigma\rangle)]\right]$$

$$\Rightarrow [p'_0 = \underline{s} \wedge J(\underline{s}, s)]$$

$$\wedge\ [\bar{p}_0 = \underline{s} \wedge J(\underline{s}, \bar{s})] \Rightarrow \Psi(\underline{s}, \bar{s})$$

En posant

$H = \underset{\sim}{Pref}^{<\omega}(\langle S, A, \Sigma\rangle)$

$F(\underline{s}, p) = [p_0 = \underline{s}]$

$R(s', p', a, s) = [s' = p'_{|p'|} \wedge p' \xrightarrow{a} \langle s\rangle \in \underset{\sim}{Pref}^{<\omega}(\langle S, A, \Sigma\rangle)]$

$N(s', p', a, s, p) = [s' = p'_{|p'|} \wedge p = p' \xrightarrow{a} \langle s\rangle]$

$I(\underline{s}, s, p) = [p_0 = \underline{s} \wedge J(\underline{s}, s)]$

nous obtenons

$$\exists I \in (S \times S \times H \to \{tt, ff\}).\ \forall \underline{s}, s, s', \bar{s} \in S,\ p', \bar{p} \in H,\ a \in A.$$

$$\varepsilon(\underline{s}) \Rightarrow [\exists p \in H.\ F(\underline{s}, p) \wedge I(\underline{s}, \underline{s}, p)]$$

$$\wedge\ [I(\underline{s}, s', p') \wedge t_a(s', s) \wedge R(s', p', a, s)] \Rightarrow [\exists p \in H.\ N(s', p', a, s, p) \wedge I(\underline{s}, s, p)]$$

$$\wedge\ I(\underline{s}, \bar{s}, \bar{p}) \Rightarrow \Psi(\underline{s}, \bar{s})$$

En pratique, nous évitons de raisonner sur les (préfixes des) traces de Σ en utilisant des "histoires" que nous interprétons comme des fonctions des préfixes des traces cumulant l'essentiel de l'information contenue dans ces préfixes. En comprenant H comme un ensemble d'histoires, $F(\underline{s}, h)$ comme l'assertion que l'histoire h commence en l'état $\underline{s}$, $R(s', h', a, s)$ comme l'assertion qu'au terme de l'histoire h' l'état s' a un successeur s par l'action a et $N(s', h', a, s, h)$ comme l'assertion que l'information, qu'au terme de l'histoire h' l'état s' a un successeur s par l'action a, se cumule dans l'histoire h, nous obtenons le principe d'induction suivant :

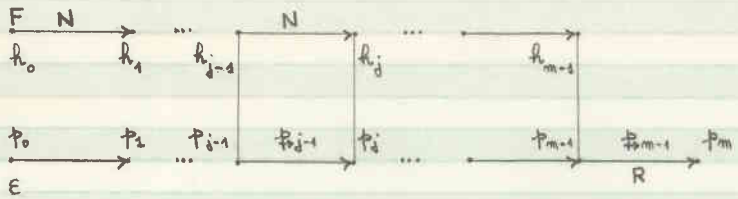
Théorème 4.2.3.1 ∿1

$$
\begin{aligned}
&[[\,\exists\, H,\ F\in(S\times H\to\{tt,ff\}),\ R\in(S\times H\times A\times S\to\{tt,ff\}),\\
&\quad N\in(S\times H\times A\times S\times H\to\{tt,ff\}).\\
&\qquad \text{Hüt}\langle S,A,\Sigma\rangle(H,F,N,R)\\
&\qquad\wedge\\
&\qquad(\exists I\in(S\times S\times H\to\{tt,ff\}).\ \forall \underline{s},s',s,\bar{s}\in S,\ h',\bar{h}\in H,\ a\in A.\\
&\qquad\qquad \mathcal{E}(\underline{s})\Rightarrow[\exists \underline{h}\in H.\ F(\underline{s},\underline{h})\wedge I(\underline{s},\underline{s},\underline{h})] \qquad (\text{-↑J})\\
&\qquad\qquad\wedge\\
&\qquad\qquad [I(\underline{s},s',h')\wedge t_a(s',s)\wedge R(s',h',a,s)]\\
&\qquad\qquad\qquad\Rightarrow[\exists h\in H.\ N(s',h',a,s,h)\wedge I(\underline{s},s,h)]\\
&\qquad\qquad\wedge\\
&\qquad\qquad I(\underline{s},\bar{s},\bar{h})\Rightarrow\Psi(\underline{s},\bar{s}))]\\
&\iff\\
&[\forall p\in\Sigma,\ i\in|p|.\ \Psi(p_0,p_i)]
\end{aligned}
$$

où

$$
\begin{aligned}
&\text{Hüt}\langle S,A,\Sigma\rangle(H,F,N,R)=\\
&\quad[\forall p\in\Sigma,\ m\in\omega.\ (1\leqslant m<|p|)\Rightarrow\\
&\qquad(\forall h\in(m\to H).\ [F(p_0,h_0)\wedge\forall j\in(m\setminus 0).\ N(p_{j-1},h_{j-1},p_{j-1},p_j,h_j)]\\
&\qquad\qquad\Rightarrow R(p_{m-1},h_{m-1},p_{m-1},p_m))]
\end{aligned}
$$

une formule qui se comprend mieux à l'aide du schéma suivant :



Démonstration

($\Rightarrow$) Soit $p \in \Sigma$. Nous démontrons par induction sur $m \in (|p| \vee 0)$ qu'il existe $h \in (m \to H)$ tel que $\forall j \in m.\ I(p_0, p_j, h_j) \wedge F(p_0, h_0) \wedge \forall j \in (m \vee 0).\ N(p_{j-1}, h_{j-1}, \beta_{j-1}, p_j, h_j)$. En effet pour $m=1$, nous avons $\mathcal{E}(p_0) \Rightarrow \exists h_0 \in H.\ F(p_0, h_0) \wedge I(p_0, p_0, h_0)$. Si par hypothèse d'induction le lemme est vrai pour $m-1 \geq 1$ et $m \in |p|$ alors nous avons $I(p_0, p_{m-2}, h_{m-2})$ et d'après la condition Hist, $R(p_{m-2}, h_{m-2}, \beta_{m-2}, p_{m-1})$ ce qui avec $\tau_{\beta_{m-2}}(p_{m-2}, p_{m-1})$ implique que $\exists h_{m-1} \in H.\ N(p_{m-2}, h_{m-2}, \beta_{m-2}, p_{m-1}, h_{m-1}) \wedge I(p_0, p_{m-1}, h_{m-1})$. Nous en déduisons que $\forall j \in |p|.\ I(p_0, p_j, h_j)$ et donc $\Psi(p_0, p_j)$.

($\Leftarrow$) Si $\forall p \in \Sigma, i \in |p|.\ \Psi(p_0, p_i)$ alors nous pouvons choisir :
$H = 2^{\Sigma} \times \omega$, $F(\Delta, \langle T, m \rangle) = [T = \{p \in \Sigma : p_0 = \Delta\} \neq \phi \wedge m = 0]$, $N(\Delta', \langle T', m' \rangle, a, \Delta, \langle T, m \rangle) =$
$[m = m'+1 \wedge T = \{p \in T' : m \in |p| \wedge p_{m'} = \Delta' \wedge \beta_{m'} = a \wedge p_m = \Delta\} \neq \phi]$, $R(\Delta', \langle T', m' \rangle, a, \Delta) =$
$[\exists p \in T'.\ (m'+1) \in |p| \wedge p_{m'} = \Delta' \wedge \beta_{m'} = a \wedge p_{m'+1} = \Delta]$ et $I(\underline{\Delta}, \Delta, \langle T, m \rangle) =$
$[T = \{p \in \Sigma : m \in |p| \wedge p_0 = \underline{\Delta} \wedge p_m = \Delta\} \neq \phi]$.
□

Observons que $(\text{-}\uparrow\mathcal{J})$ est une généralisation de $(\text{-}\mathcal{J})$ que nous retrouvons en choisissant $F = tt$, $R = tt$, $N = tt$ et $H = 1$.

4.2.3.2 Principes d'induction pour une sémantique non fermée par fusions définie par concordance avec une sémantique close

Il est toujours possible de définir une sémantique non close $\langle S, A, \Sigma \rangle$ par concordance avec une sémantique close (engendrée par un système de transition $\langle S^\#, A^\#, t^\#, \varepsilon^\# \rangle$) à une fonction $f_\Delta^\# \in (S^\# \to S)$ entre états près (cf. 2.7.2.2 ∾1) :

$$\langle S, A, \Sigma \rangle = \simeq \langle f_\Delta^\# \rangle (\langle S^\#, A^\#, \Sigma \langle S^\#, A^\#, t^\#, \varepsilon^\# \rangle \rangle)$$

Par conséquent, pour démontrer que $\psi \in (S \times S \to \{tt, ff\})$ est invariante pour $\langle S, A, \Sigma \rangle$ il est toujours correct et possible (d'après 4.1.3 v 3 et 4.1.3 v 4) d'utiliser l'un quelconque des principes d'induction du paragraphe 4.2.1 pour $\langle S^\#, A^\#, t^\#, \varepsilon^\# \rangle$ et $\psi^\#$ définie par $\psi^\#(a_0, a_1) = \psi(f_a^\#(a_0), f_a^\#(a_1))$.

Par exemple, en utilisant le principe d'induction (-J), nous obtenons :

$$
\begin{array}{l}
[\, \exists S^\#, A^\#, t^\# \in (A^\# \to (S^\# \times S^\# \to \{tt, ff\})),\ \varepsilon^\# \in (S^\# \to \{tt, ff\}),\ e^\# \in (S^\# \to S). \\
\quad \exists I \in (S^\# \times S^\# \to \{tt, ff\}).\ \forall \underline{s}, s, \bar{s} \in S^\#,\ a \in A^\#. \\
\qquad \langle S, A, \Sigma \rangle = \mathcal{U} \langle f_a^\# \rangle (\langle S^\#, A^\#, \Sigma \langle S^\#, A^\#, t^\#, \varepsilon^\# \rangle \rangle) \\
\qquad \wedge \\
\qquad \varepsilon^\#(\underline{s}) \Rightarrow I(\underline{s}, \underline{s}) \qquad\qquad (\text{-J}^\#) \\
\qquad \wedge \\
\qquad [\exists s' \in S^\#.\ I(\underline{s}, s') \wedge t_a^\#(s', s)] \Rightarrow I(\underline{s}, s) \\
\qquad \wedge \\
\qquad I(\underline{s}, \bar{s}) \Rightarrow \psi(f_a^\#(\underline{s}), f_a^\#(\bar{s}))\,] \\
\Longleftrightarrow \\
[\forall p \in \Sigma, i \in |p|.\ \psi(p_0, p_i)\,]
\end{array}
$$

4.2.3.3 Equivalence forte des deux principes d'induction (-↑J) et (-J#)

Théorème 4.2.3.3 v 1

Toute preuve d'invariance de ψ pour une sémantique quelconque $\langle S, A, \Sigma \rangle$ utilisant le principe d'induction (-J#) peut se recrire en une preuve utilisant (-↑J), et réciproquement.

Démonstration

- Ayant trouvé $S^\#$, $A^\#$, $t^\#$, $\varepsilon^\#$, $f_\Delta^\#$ et $I^\#$ satisfaisant les conditions données en $(\neg\mathcal{Y}^\#)$, nous pouvons toujours reécrire cette preuve d'invariance en une preuve utilisant le principe d'induction $(\neg\uparrow\mathcal{Y})$ en posant :

$H = S^\# \times S^\#$

$F(\underline{\Delta}, <\underline{\Delta}^\#, \Delta^\#>) = [\varepsilon^\#(\underline{\Delta}^\#) \wedge \Delta^\# = \underline{\Delta}^\# \wedge \underline{\Delta} = f_\Delta^\#(\underline{\Delta}^\#)]$

$R(\Delta', <\underline{\Delta}^{\#\prime}, \Delta^{\#\prime}>, a, \Delta) = [\Delta' = f_\Delta^\#(\Delta^{\#\prime}) \wedge \exists \Delta^\# \in S^\#.(t_a^\#(\Delta^{\#\prime}, \Delta^\#) \wedge \Delta = f_\Delta^\#(\Delta^\#))]$

$N(\Delta', <\underline{\Delta}^{\#\prime}, \Delta^{\#\prime}>, a, \Delta, <\underline{\Delta}^\#, \Delta^\#>) = [\Delta' = f_\Delta^\#(\Delta^{\#\prime}) \wedge t_a^\#(\Delta^{\#\prime}, \Delta^\#) \wedge \Delta = f_\Delta^\#(\Delta^\#)]$

$I(\underline{\Delta}, \Delta, <\underline{\Delta}^\#, \Delta^\#>) = [I^\#(\underline{\Delta}^\#, \Delta^\#) \wedge \underline{\Delta} = f_\Delta^\#(\underline{\Delta}^\#) \wedge \Delta = f_\Delta^\#(\Delta^\#)]$

En effet, nous avons bien :

- $\underline{Hist}<S, A, \Sigma>(H, F, N, R)$ car si nous avons $p \in \Sigma$, $m \in \omega$, $\forall i \in m.<\underline{\Delta}_i^\#, \Delta_i^\#> \in H$ tels que $1 < m < |p|$, $F(p_0, <\underline{\Delta}_0^\#, \Delta_0^\#>)$ et $\forall j \in (m \vee 0).N(p_{j-1}, <\underline{\Delta}_{j-1}^\#, \Delta_{j-1}^\#>, \mathfrak{p}_{j-1}, p_j, <\underline{\Delta}_j^\#, \Delta_j^\#>)$ alors la séquence $\Delta_0^\# \xrightarrow{\mathfrak{p}_0} \dots \xrightarrow{\mathfrak{p}_{m-2}} \Delta_{m-1}^\#$ est préfixe d'une trace $p^\# \in \Sigma<S^\#, A^\#, t^\#, \varepsilon^\#>$ telle que $p = f_\Delta^\#(p^\#)$. Par conséquent comme $m < |p| = |p^\#|$, nous avons $p_{m-1} = f_\Delta^\#(p_{m-1}^\#) = f_\Delta^\#(\Delta_{m-1}^\#)$ et $t_{\mathfrak{p}_{m-1}^\#}^\#(p_{m-1}^\#, p_m^\#)$ donc $t_{\mathfrak{p}_{m-1}^\#}^\#(\Delta_{m-1}^\#, p_m^\#)$ et $p_m = f_\Delta^\#(p_m^\#)$ soit $R(p_{m-1}, <\underline{\Delta}_{m-1}^\#, \Delta_{m-1}^\#>, \mathfrak{p}_{m-1}, p_m)$.

- Si $\varepsilon(\underline{\Delta})$ est vrai, alors $\exists \underline{\Delta}^\# \in S^\#.(\varepsilon^\#(\underline{\Delta}^\#) \wedge \underline{\Delta} = f_\Delta^\#(\underline{\Delta}^\#))$ donc $F(\underline{\Delta}, <\underline{\Delta}^\#, \underline{\Delta}^\#>)$ et $I^\#(\underline{\Delta}^\#, \underline{\Delta}^\#)$ donc $F(\underline{\Delta}, <\underline{\Delta}^\#, \underline{\Delta}^\#>) \wedge I(\underline{\Delta}, \Delta, <\underline{\Delta}^\#, \underline{\Delta}^\#>)$.

- Si $I(\underline{\Delta}, \Delta', <\underline{\Delta}^\#, \Delta^{\#\prime}>) \wedge t_a(\Delta', \Delta) \wedge R(\Delta', <\underline{\Delta}^\#, \Delta^{\#\prime}>, a, \Delta)$ est vrai, alors $\underline{\Delta} = f_\Delta^\#(\underline{\Delta}^\#) \wedge I^\#(\underline{\Delta}^\#, \Delta^{\#\prime}) \wedge t_a(\Delta^{\#\prime}, \Delta^\#) \wedge \Delta = f_\Delta^\#(\Delta^\#)$ impliquent $N(\Delta', <\underline{\Delta}^\#, \Delta^{\#\prime}>, a, \Delta, <\underline{\Delta}^\#, \Delta^\#>)$ et $I^\#(\underline{\Delta}^\#, \Delta^\#)$ donc $I(\underline{\Delta}, \Delta, <\underline{\Delta}^\#, \Delta^\#>)$.

- Si $I(\underline{\Delta}, \Delta, <\underline{\Delta}^\#, \Delta^\#>)$ est vrai alors nous avons $I^\#(\underline{\Delta}^\#, \Delta^\#)$ qui implique $\Psi(f_\Delta^\#(\underline{\Delta}^\#), f_\Delta^\#(\Delta^\#)) = \Psi(\underline{\Delta}, \Delta)$.

- Réciproquement, ayant trouvé E, H, F, R, N et I satisfaisant les conditions de $(\neg\uparrow\mathcal{Y})$, nous pouvons toujours reécrire cette preuve d'invariance en une preuve utilisant le principe d'induction $(\neg\mathcal{Y}^\#)$ comme suit :

Etant donnée $p \in \Sigma$, nous construisons $p^\# \in \Sigma\langle S \times H, A\rangle$ comme suit : Comme $E(p_0) \Rightarrow \exists h_0 \in H.\ F(p_0, h_0) \wedge I(p_0, p_0, h_0)$, nous choisissons $p_0^\# = \langle p_0, h_0\rangle$. Nous avons aussi $R(p_0, h_0, \pi_0, p_1)$ et $I(p_0, p_0, h_0) \wedge t_{\pi_0}(p_0, p_1)$ qui impliquent $\exists h_1 \in H.\ N(p_0, h_0, \pi_0, p_1, h_1) \wedge I(p_0, p_1, h_1)$. Nous posons alors $\pi_0^\# = \pi_0$ et $p_1^\# = \langle p_1, h_1\rangle$. Ayant construit $\pi_j^\#$ pour $j = 0, \dots, m-2$ et $p_j^\# = \langle p_j, h_j\rangle$ pour $j = 0, \dots, m-1$ avec $1 < m < |p|$ tel que $F(p_0, h_0) \wedge \forall j \in (m \vee 0).\ N(p_{j-1}, h_{j-1}, \pi_{j-1}, p_j, h_j) \wedge \forall j \in m.\ I(p_0, p_j, h_j)$, nous avons $t_{\pi_{m-1}}(p_{m-1}, p_m)$ et $R(p_{m-1}, h_{m-1}, \pi_{m-1}, p_m)$ et donc $\exists h_m \in H.\ N(p_{m-1}, h_{m-1}, \pi_{m-1}, p_m, h_m) \wedge I(p_0, p_m, h_m)$. Nous posons alors $\pi_{m-1}^\# = \pi_{m-1}$ et $p_m^\# = \langle p_m, h_m\rangle$.

Posons maintenant :

$\Sigma^\# = \{p^\# : p \in \Sigma\}$

$S^\# = S \times 2^{\Sigma^\#} \times \omega$

$A^\# = A$

$\varepsilon^\#(\langle A, T, m\rangle) = [T = \{p^\# \in \Sigma^\# : p_0^\#(0) = A\} \neq \phi \wedge m = 0]$

$t_a^\#(\langle A', T', m'\rangle, \langle A, T, m\rangle) = [m = m'+1 \wedge T = \{p^\# \in T' : m \in |p^\#| \wedge p_{m'}^\#(0) = A' \wedge \pi_{m'}^\# = a \wedge p_m^\#(0) = A\} \neq \phi]$

$I^\#(\langle \underline{A}, \underline{T}, \underline{m}\rangle, \langle A, T, m\rangle) = [T = \{p^\# \in \Sigma^\# : m \in |p^\#| \wedge p_0^\#(0) = \underline{A} \wedge p_m^\#(0) = A\} \neq \phi]$

$f_A^\#(\langle A, T, m\rangle) = A$

alors, nous avons bien :

- $\langle S, A, \Sigma\rangle = \simeq \langle f_A^\#\rangle(\langle S^\#, A^\#, \Sigma\langle S^\#, A^\#, t^\#, \varepsilon^\#\rangle\rangle)$ par construction.
- $\varepsilon^\#(\langle \underline{A}, \underline{T}, \underline{m}\rangle) = [T = \{p^\# \in \Sigma^\# : p_0^\#(0) = \underline{A}\} \neq \phi \wedge \underline{m} = 0] \Rightarrow I^\#(\langle \underline{A}, \underline{T}, \underline{m}\rangle, \langle \underline{A}, \underline{T}, \underline{m}\rangle)$.
- $\exists \langle A', T', m'\rangle \in S^\#.\ (I^\#(\langle \underline{A}, \underline{T}, \underline{m}\rangle, \langle A', T', m'\rangle) \wedge t_a^\#(\langle A', T', m'\rangle, \langle A, T, m\rangle)) \Leftrightarrow$

$\exists \langle A', T', m'\rangle \in S^\#.\ [T' = \{p^\# \in \Sigma^\# : m' \in |p^\#| \wedge p_0^\#(0) = \underline{A} \wedge p_{m'}^\#(0) = A'\} \neq \phi \wedge m = m'+1 \wedge T = \{p^\# \in T' : m \in |p^\#| \wedge p_{m'}^\#(0) = A' \wedge \pi_{m'}^\# = a \wedge p_m^\#(0) = A\} \neq \phi] \Rightarrow [T = \{p^\# \in \Sigma : m \in |p^\#| \wedge p_0^\#(0) = \underline{A} \wedge p_m^\#(0) = A\} \neq \phi] = I^\#(\langle \underline{A}, \underline{T}, \underline{m}\rangle, \langle A, T, m\rangle)$.

- $I^\#(\langle \underline{A}, \underline{T}, \underline{m}\rangle, \langle \bar{A}, \bar{T}, \bar{m}\rangle) = [T = \{p^\# \in \Sigma^\# : \bar{m} \in |p^\#| \wedge p_0^\#(0) = \underline{A} \wedge p_{\bar{m}}^\#(0) = \bar{A}\} \neq \phi] \Rightarrow$

$\{p \in \Sigma : \bar{m} \in |p| \wedge p_0 = \underline{A} \wedge p_{\bar{m}} = \bar{A}\} \neq \phi \Rightarrow \psi(\underline{A}, \bar{A}) \Rightarrow \psi(f_A^\#(\langle \underline{A}, \underline{T}, \underline{m}\rangle), f_A^\#(\langle \bar{A}, \bar{T}, \bar{m}\rangle))$.

□

4.3 CONSTRUCTION D'UNE METHODE DE PREUVE D'INVARIANCE A PARTIR D'UNE SEMANTIQUE OPERATIONNELLE ET D'UN PRINCIPE D'INDUCTION PAR DECOMPOSITION DE L'INVARIANT GLOBAL EN INVARIANTS LOCAUX

4.3.1 FORMALISATION DE LA CONSTRUCTION

Nous montrons comment construire formellement une méthode de preuve pour un langage de programmation étant donnés une sémantique opérationnelle, un principe d'induction, un langage d'assertions et sémantique.

4.3.1.1 Définition de la sémantique opérationnelle

Pour construire formellement une méthode de preuve de propriétés d'invariance pour un langage de programmation $\mathcal{P}r$ il faut commencer par en définir la sémantique opérationnelle,

$$Pr \in \mathcal{P}r \longrightarrow \langle S[\![Pr]\!], A[\![Pr]\!], \Sigma[\![Pr]\!] \rangle$$

ce qui donne $\varepsilon[\![Pr]\!]$ et $t[\![Pr]\!]$ (cf. 2.3), ou bien

$$Pr \in \mathcal{P}r \longrightarrow \langle S[\![Pr]\!], A[\![Pr]\!], t[\![Pr]\!], \varepsilon[\![Pr]\!] \rangle$$

ce qui donne $\Sigma[\![Pr]\!]$ (cf. 2.4).

4.3.1.2 Définition de la propriété invariante à démontrer

Il faut ensuite définir la propriété d'invariance d'intérêt en définissant un ensemble $\mathcal{P}$ de propriétés et une application

$$\langle Pr \in \mathcal{P}r,\ P \in \mathcal{P} \rangle \longrightarrow \psi[\![Pr, P]\!] \in (S[\![Pr]\!] \times S[\![Pr]\!] \to \{tt, ff\})$$

de sorte que la spécification "Pr a la propriété P" signifie :

$$\forall p \in \Sigma[\![Pr]\!],\ i \in |p|.\ \psi[\![Pr, P]\!](p_0, p_i)$$

4.3.1.3 Choix d'un principe d'induction

Il faut ensuite choisir un principe d'induction qui, de manière générale a la forme :

$$[\exists I \in As[\![S]\!].\ Cv[\![S, A, \Sigma, t, \varepsilon, \psi]\!](I)$$

En remplaçant $S, A, \Sigma, t, \varepsilon$ et ψ par leurs définitions dans ce principe d'induction, nous obtenons les conditions de vérification d'une propriété P pour un programme Pr quelconque. Il s'agit de déterminer l'application :

$$\langle Pr, P \rangle \longrightarrow \langle \tilde{As}[\![Pr, P]\!],\ \tilde{Cv}[\![Pr, P]\!] \rangle$$

de sorte qu'une preuve ait la forme :

$$[\exists \tilde{I} \in \tilde{As}[\![Pr, P]\!].\ \tilde{Cv}[\![Pr, P]\!](\tilde{I})]$$

et consiste donc à inventer un invariant $\tilde{I}$ (qui se présente généralement sous forme d'une conjonction d'invariants locaux associés à certains points du programme) puis à démontrer qu'il satisfait certaines conditions de vérification $\tilde{Cv}[\![Pr, P]\!]$.

On peut choisir $\tilde{As}[\![Pr, P]\!] = As[\![S[\![Pr]\!]]\!]$ et $\tilde{Cv}[\![Pr, P]\!] = Cv[\![S[\![Pr]\!], A[\![Pr]\!], \Sigma[\![Pr]\!], t[\![Pr]\!], \varepsilon[\![Pr]\!], \psi[\![Pr]\!]]\!]$ mais ce choix ne permet pas de faire toutes les simplifications désirables.

C'est pourquoi nous avons proposé une autre démarche (Cousot-Cousot [80c], Cousot-R [81], Cousot-Cousot [82a], Cousot-Cousot [84]) qui se justifie comme suit :

La méthode de preuve doit être correcte :

$$[\exists \tilde{I} \in \tilde{A_s}[\![Pr,P]\!].\ \tilde{C_v}[\![Pr,P]\!](\tilde{I})] \Longrightarrow [\forall p \in \Sigma[\![Pr]\!], i \in |p|.\ \psi[\![Pr,P]\!](p_0,p_i)]$$

et comme le principe d'induction choisi est correct et (sémantiquement) complet, ceci équivaut à :

$$[\exists \tilde{I} \in \tilde{A_s}[\![Pr,P]\!].\ \tilde{C_v}[\![Pr,P]\!](\tilde{I})] \Longrightarrow [\exists I \in A_s[\![S]\!].\ C_v[\![S,A,\Sigma,t,\varepsilon,\psi]\!](I)]$$

ou encore :

$$\forall \tilde{I} \in \tilde{A_s}[\![Pr,P]\!].\ \tilde{C_v}[\![Pr,P]\!](\tilde{I}) \Longrightarrow \exists I \in A_s[\![S]\!].\ C_v[\![S,A,\Sigma,t,\varepsilon,\psi]\!](I)$$

d'où nous déduisons qu'il doit exister une fonction $\gamma \in (\tilde{A_s}[\![Pr,P]\!] \to A_s[\![S[\![Pr]\!]]\!])$ telle que :

$$\tilde{C_v}[\![Pr,P]\!] \Longrightarrow C_v[\![S,A,\Sigma,t,\varepsilon,\psi]\!] \circ \gamma$$

De façon similaire, la méthode de preuve doit être (sémantiquement) complète, d'où nous déduisons qu'il doit exister une fonction $\alpha \in (A_s[\![S[\![Pr]\!]]\!] \to \tilde{A_s}[\![Pr,P]\!])$ telle que :

$$C_v[\![S,A,\Sigma,t,\varepsilon,\psi]\!] \Longrightarrow \tilde{C_v}[\![Pr,P]\!] \circ \alpha$$

La fonction $\gamma[\![Pr,P]\!]$ donne la signification $\gamma[\![Pr,P]\!](\tilde{I})$ des invariants "locaux" $\tilde{I} \in \tilde{A_s}[\![Pr,P]\!]$ en terme des invariants "globaux" de $A_s[\![S[\![Pr]\!]]\!]$, tandis que la fonction $\alpha[\![Pr,P]\!]$ donne la représentation $\alpha[\![Pr,P]\!](I)$ de l'invariant "global" $I \in A_s[\![S[\![Pr]\!]]\!]$ par des invariants "locaux" de $\tilde{A_s}[\![Pr,P]\!]$. Ces éléments conduisent à poursuivre la construction de la méthode de preuve comme suit :

4.3.1.4 Choix d'un langage pour exprimer les invariants locaux

En pratique l'invariant utilisé pour faire la preuve n'est pas un élément de $As[\![S[\![Pr]\!]]\!]$ (ce qui donnerait un invariant global pour tout le programme) mais il s'exprime généralement de manière équivalente mais plus aisée comme élément d'un certain ensemble $A\tilde{s}[\![Pr, P]\!]$ (que nous appelerons conventionnellement "langage"), le plus souvent sous forme d'invariants locaux associés à divers points du programme.

Exemple 4.3.1.4-1

Considérons l'exemple 4.2.1.1-1, nous avons pour le programme Pr :

```
1:
    Q := 0;
2:
    while X ≥ Y do
3:
        Q := Q+1;
4:
        X := X-Y;
5:
    od;
6:
```

$C = \{1, \ldots, 6\}$ états de contrôle

$\mathcal{V} = \{X, Y, Q\}$ variables

$\mathcal{M} = (\mathcal{V} \to \mathbb{Z})$ états mémoire

$S = C \times \mathcal{M}$ états

Un invariant $I \in As[\![S]\!] = (S \times S \to \{tt, ff\})$ est exprimé comme un vecteur $\tilde{I} = \langle P_1, \ldots, P_6 \rangle$ où $P_i \in (\mathbb{Z}^5 \to \{tt, ff\})$, $i = 1, \ldots, 6$. Par conséquent $A\tilde{s}[\![Pr, \psi]\!] = \prod_{c \in C} (\mathbb{Z}^5 \to \{tt, ff\})$

□

4.3.1.5 Définition de la sémantique du langage exprimant les invariants locaux

La sémantique du langage $\tilde{A_S}[\![P_r, P]\!]$ exprimant les invariants locaux se définit en terme de $A_S[\![S[\![P_r]\!]]\!]$ au moyen de deux fonctions :

$$\alpha[\![P_r, P]\!] \in (A_S[\![S[\![P_r]\!]]\!] \to \tilde{A_S}[\![P_r, P]\!])$$
$$\gamma[\![P_r, P]\!] \in (\tilde{A_S}[\![P_r, P]\!] \to A_S[\![S[\![P_r]\!]]\!])$$

Exemple 4.3.1.5-1

La signification de $\tilde{I} = \langle P_1, \ldots, P_6 \rangle$ (cf. exemple 4.2.1.1-1) est $I = \delta(\tilde{I})$ tel que $I(\underline{\Delta}, \Delta) = [\exists i \in C, \underline{x}, \underline{y}, \underline{q}, x, y, q \in \mathbb{Z}. \ \underline{\Delta} = \langle 1, \langle \underline{x}, \underline{y}, \underline{q} \rangle\rangle \wedge \Delta = \langle i, \langle x, y, q \rangle\rangle \wedge P_i(\underline{x}, \underline{y}, x, y, q)]$. Ceci formalise le fait que $P_i(\underline{x}, \underline{y}, x, y, q)$ est vrai entre les valeurs initiales $\underline{x}, \underline{y}$ et les valeurs courantes x, y, q des variables X, Y, Q du programme quand le contrôle est au point i.

Réciproquement, un invariant $I \in (S \times S \to \{tt, ff\})$ peut être représenté par $\alpha(I) = \langle P_1, \ldots, P_6 \rangle$ tel que pour tout $i = 1, \ldots, 6$, $P_i(\underline{x}, \underline{y}, x, y, q) = [\exists \underline{q} \in \mathbb{Z} . \ I(\langle 1, \langle \underline{x}, \underline{y}, \underline{q} \rangle\rangle, \langle i, \langle x, y, q \rangle\rangle)]$.

□

4.3.1.6 Propriétés du langage exprimant les invariants locaux et sa sémantique

4.3.1.6.1 Treillis complets des invariants locaux

Dans les principes d'induction que nous avons considérés au paragraphe 4.2, $As[\![S]\!]$ était de la forme $(s \to \{tt, ff\})$, $(s \times s \to \{tt, ff\})$, etc., etc., donc toujours un treillis complet booléen $\langle As[\![S]\!], \Rightarrow, \vee, \wedge, tt, ff, \neg \rangle$.

Quand on choisit le langage $\tilde{As}[\![Pr, P]\!]$ pour exprimer les invariants locaux, il faut pouvoir exprimer qu'un invariant est plus fort ou plus faible qu'un autre de façon à pouvoir traduire dans $\tilde{Cv}$ l'implication qui figure dans Cv. Il est donc nécessaire de disposer d'un ordre partiel $\sqsubseteq$ sur $\tilde{As}$ correspondant à $\Rightarrow$ sur As. Ceci nous conduira à choisir $\langle \tilde{As}[\![Pr, P]\!], \sqsubseteq \rangle$ comme un ensemble partiellement ordonné.

Il arrive souvent que $\langle \tilde{As}[\![Pr, P]\!], \sqsubseteq \rangle$ soit un treillis complet. En effet, la propriété que la conjonction d'invariants est invariante doit également être conservée pour $\tilde{As}[\![Pr, P]\!]$ car elle permet de combiner des preuves indépendantes en une seule sans aucune vérification supplémentaire. Ceci signifie que $\tilde{As}[\![Pr, P]\!]$ doit être muni d'une opération borne inférieure $\prod_{i \in \Delta} \tilde{I}_i$ de $\{\tilde{I}_i : i \in \Delta\}$ pour $\sqsubseteq$. Enfin l'invariant tt de $As[\![S[\![Pr]\!]]\!]$ exprime qu'on ne sait rien sur le programme Pr. Nous pourrons toujours ajouter l'équivalent à $\tilde{As}[\![Pr, P]\!]$ sous la forme d'un supremum $\top$ tel que $\forall \tilde{I} \in \tilde{As}[\![Pr, P]\!]. \tilde{I} \sqsubseteq \top$. Avec ces hypothèses $\langle \tilde{As}[\![Pr, P]\!], \sqsubseteq, \sqcap, \top \rangle$ est un inf-demi-treillis complet avec supremum et par conséquent c'est un treillis complet $\langle \tilde{As}[\![Pr, P]\!], \sqsubseteq, \sqcap, \sqcup, \top, \bot \rangle$.

Quand nous utiliserons les principes d'induction contrapositifs qui utilisent la négation, nous serons amenés à choisir $\langle \tilde{As}[\![Pr, P]\!], \sqsubseteq, \sqcup, \sqcap, \top, \bot, \neg \rangle$ comme étant un treillis complet booléen.

Exemple 4.3.1.6.1-1

Dans l'exemple 4.3.1.4-1, nous avons :

$S = C \times (\mathcal{V} \to \mathbb{Z})$ où $C = \{1, \dots, 6\}$ et $\mathcal{V} = \{X, Y, Q\}$

$A_S[\![S]\!] = (S \times S \to \{tt, ff\})$ qui est un treillis complet booléen pour $\Rightarrow, \vee, \wedge, tt, ff, \neg$.

$\tilde{A_S}[\![P_x, \psi]\!] = \prod_{c \in C} (\mathbb{Z}^S \to \{tt, ff\})$ qui est un treillis complet booléen pour $\sqsubseteq, \sqcup, \sqcap, \top, \bot, \neg$ définis par :

$\langle P_1, \dots, P_6 \rangle \sqsubseteq \langle Q_1, \dots, Q_6 \rangle$ si et seulement si $\bigwedge_{i=1}^{6} P_i \Rightarrow Q_i$

$\bigsqcup_{i \in \Delta} \langle P_1^i, \dots, P_6^i \rangle = \langle \bigvee_{i \in \Delta} P_1^i, \dots, \bigvee_{i \in \Delta} P_6^i \rangle$ et de même pour $\sqcap$ et $\wedge$

$\bot = \langle ff, \dots, ff \rangle$ et de même pour $\top$ et tt

$\neg \langle P_1, \dots, P_6 \rangle = \langle \neg P_1, \dots, \neg P_6 \rangle$

□

4.3.1.6.2 Correspondance entre invariants locaux et globaux

La paire (α, γ) de fonctions $\alpha \in (A_S \to \tilde{A_S})$ et $\gamma \in (\tilde{A_S} \to A_S)$ entre invariants globaux A_S et invariants locaux $\tilde{A_S}$ a souvent des propriétés intéressantes qui peuvent être exploitées pour construire la méthode de preuve. Ces propriétés sont les suivantes :

4.3.1.6.2.1 Correspondance monotone

Le fait que l'ordre $\sqsubseteq$ sur $\tilde{A_S}$ corresponde à l'implication $\Rightarrow$ sur A_S s'exprime par la monotonie de α et de γ :

(a) $\forall I_1, I_2 \in A_S . \; [(I_1 \Rightarrow I_2) \Rightarrow (\alpha(I_1) \sqsubseteq \alpha(I_2))]$

(b) $\forall \tilde{I}_1, \tilde{I}_2 \in \tilde{A_S} . \; [(\tilde{I}_1 \sqsubseteq \tilde{I}_2) \Rightarrow (\gamma(\tilde{I}_1) \Rightarrow \gamma(\tilde{I}_2))]$

4.3.1.6.2.2 Demi-correspondance de Galois

Supposant α et γ monotones (et donc que $\sqsubseteq$ correspond à $\Rightarrow$), il se peut qu'on ait $\neg(I \Rightarrow \gamma(\tilde{I}))$ et $\alpha(I) \sqsubseteq \tilde{I}$. Supposons par exemple que I soit le plus fort invariant pour une sémantique $\langle S, A, \Sigma\rangle$ (c'est-à-dire que $I(\underline{\Delta}, \Delta) = [\exists p \in \Sigma, i \in |p| \cdot p_0 = \underline{\Delta} \wedge p_i = \Delta]$). Alors (le sens de) $\tilde{I} \in \tilde{A}_{\tilde{\Delta}}$ est invariant si et seulement si $I \Rightarrow \gamma(\tilde{I})$. Nous voudrions pouvoir transposer ce raisonnement dans $\tilde{A}_{\tilde{\Delta}}$ c'est-à-dire que $\alpha(I) \sqsubseteq \tilde{I}$. Ceci nous amène à supposer que α et γ sont monotones et que

$\forall I \in A_\Delta, \tilde{I} \in \tilde{A}_{\tilde{\Delta}} \cdot (\alpha(I) \sqsubseteq \tilde{I}) \Rightarrow (I \Rightarrow \gamma(\tilde{I}))$

ou encore, de manière équivalente

(a) $\alpha \in (A_\Delta \to \tilde{A}_{\tilde{\Delta}})$ est monotone

(b) $\gamma \in (\tilde{A}_{\tilde{\Delta}} \to A_\Delta)$ est monotone

(c) $\gamma \circ \alpha$ est extensive

4.3.1.6.2.3 Quasi-correspondance de Galois

Quand (α, γ) est une demi-correspondance de Galois, il se peut que $\gamma \circ \alpha(I) \not\Rightarrow J$ même s'il existe $\tilde{I}$ tel que $I \Rightarrow \gamma(\tilde{I}) \Rightarrow J$. Autrement dit la représentation $\alpha(I)$ de I dans $\tilde{A}_{\tilde{\Delta}}$ donne moins d'informations que si on avait choisi $\alpha(I) = \tilde{I}$. Pour que $\alpha(I)$ soit la meilleure représentation possible de $I \in A_\Delta$ dans $\tilde{A}_{\tilde{\Delta}}$, il faut supposer que

$$\gamma \circ \alpha(I) \Rightarrow \wedge\{\gamma(\tilde{I}) : I \Rightarrow \gamma(\tilde{I})\}$$

pour éviter que $\gamma \circ \alpha(I) \not\Rightarrow J$ lorsque $I \Rightarrow \gamma(\tilde{I}) = J$. Comme $\gamma \circ \alpha$ est extensive cette condition revient à supposer que $\gamma \circ \alpha(I) = \wedge\{\gamma(\tilde{I}) : I \Rightarrow \gamma(\tilde{I})\}$, soit

(a) $\alpha \in (A_\Delta \to \tilde{A}_{\tilde{\Delta}})$ est monotone

(b) $\gamma \in (\tilde{A}_{\tilde{\Delta}} \to A_\Delta)$ est monotone

(c) $\gamma \circ \alpha$ est extensive

(d) $\forall I \in A_\Delta \cdot [\gamma \circ \alpha(I) = \wedge\{\gamma(\tilde{I}) : I \Rightarrow \gamma(\tilde{I})\}]$

Observons que les propriétés (c) et (d) sont indépendantes et que $\gamma \circ \alpha$ est l'unique opérateur de fermeture supérieure $\varrho \in (A_S \to A_S)$ tel que $\varrho[A_S] = \gamma[\tilde{A_S}]$. Autrement dit, à la représentation près, seuls les éléments de A_S qui appartiennent à $\varrho[A_S]$ sont disponibles quand on raisonne dans $\tilde{A_S}$ au lieu de A_S. Tout invariant I devra être approché par un invariant plus faible, le meilleur étant $\varrho(I)$. En général $\varrho(I) \neq I$ et cette perte d'information peut evidemment conduire à des problèmes d'incomplétude mais ce n'est pas toujours le cas.

4.3.1.6.2.4 Correspondance de Galois

Bien que (α, γ) soit une quasi-correspondance de Galois, α peut ne pas être un morphisme complet pour les disjonctions. Comme le plus fort invariant pour les principes d'induction (positifs) s'exprime comme une disjonction (en général infinie) ceci peut avoir pour conséquence que la représentation du plus fort invariant par α dans $\tilde{A_S}$ conduise à une perte d'information qui est source d'incomplétude. La propriété que α est un morphisme complet pour la disjonction et donc de manière équivalente que la paire (α, γ) est une correspondance de Galois peut donc être utile :

(a) $\alpha \in (A_S \to \tilde{A_S})$ est monotone

(b) $\gamma \in (\tilde{A_S} \to A_S)$ est monotone

(c) $\gamma \circ \alpha$ est extensive

(e) $\alpha \circ \gamma$ est réductive

(on trouvera des définitions équivalentes en annexe II).

4.3.1.6.2.5 Correspondance de Galois surjective

Les éléments de $\tilde{A}_S \setminus \alpha[A_S]$ ne servent à représenter aucune information de A_S et peuvent être éliminés en choisissant $\tilde{A}_S' = \alpha[A_S]$. Dans ces conditions, nous avons les propriétés :

(a) $\alpha \in (A_S \to \tilde{A}_S)$ est monotone

(b) $\gamma \in (\tilde{A}_S \to A_S)$ est monotone

(c) $\gamma \circ \alpha$ est extensive

(e) $\alpha \circ \gamma$ est réductive

(f) α est surjective

que nous avons fréquemment utilisées (cf. par exemple à Cousot-Cousot [76], Cousot-Cousot [77a]).

4.3.1.6.2.6 Correspondance de Galois injective

Si $I_1, I_2 \in A_S$ sont différents et ont la même représentation $\alpha(I_1) = \alpha(I_2)$ dans $\tilde{A}_S$, il y a une perte d'information en passant de A_S à $\tilde{A}_S$ par α puisque des invariants différents ne peuvent plus être distingués. Ceci peut conduire à des problèmes d'incomplétude sémantique puisque la propriété $I_1 \neq I_2$ ne peut pas s'exprimer dans $\tilde{A}_S$. Ceci nous amène aux conditions suivantes :

(a) $\alpha \in (A_S \to \tilde{A}_S)$ est monotone

(b) $\gamma \in (\tilde{A}_S \to A_S)$ est monotone

(c) $\gamma \circ \alpha$ est extensive

(e) $\alpha \circ \gamma$ est réductive

(g) α est injective

4.3.1.6.2.7 Isomorphisme complet

Enfin, dans le cas où les raisonnements dans A_S ou $\tilde{A_S}$ sont équivalents, α est un isomorphisme complet et γ est son inverse de sorte que les hypothèses suivantes sont satisfaites :

(R) $\quad \alpha\left(\bigvee_{i\in\Delta} I_i\right) = \bigcup_{i\in\Delta} \alpha(I_i)$

(i) $\quad \alpha\left(\bigwedge_{i\in\Delta} I_i\right) = \prod_{i\in\Delta} \alpha(I_i)$

(j) $\quad \alpha$ est bijective

(k) $\quad \gamma = \alpha^{-1}$

Exemple 4.3.1.6.2.7-1

Dans l'exemple 4.3.1.4-1 poursuivi en 4.3.1.5-1 et 4.3.1.6.1-1, nous avons :

$$\alpha_1(I)_i\ (\underline{x},\underline{y},x,y,q) = [\exists \underline{q}\in\mathbb{Z}.\ I(\langle 1,\langle \underline{x},\underline{y},\underline{q}\rangle\rangle, \langle i,\langle x,y,q\rangle\rangle)]$$

$$\gamma_1(\tilde{I})(\underline{\Delta},\Delta) = [\exists i\in C,\ \underline{x},\underline{y},\underline{q},\ x,y,q\in\mathbb{Z}.\ \underline{\Delta}=\langle 1,\langle \underline{x},\underline{y},\underline{q}\rangle\rangle \wedge \Delta=\langle i,\langle x,y,q\rangle\rangle \wedge \tilde{I}_i(\underline{x},\underline{y},x,y,q)]$$

de sorte que les conditions (a) et (b) sont satisfaites mais pas la condition (c). Nous pouvons également choisir :

$$\alpha_2(I)_i\ (\underline{x},\underline{y},x,y,q) = [\exists j\in C,\ q\in\mathbb{Z}.\ I(\langle j,\langle x,y,q\rangle\rangle, \langle i,\langle x,y,q\rangle\rangle]$$

$$\gamma_2(\tilde{I})(\underline{\Delta},\Delta) = [\exists j,i\in C,\ \underline{x},\underline{y},\underline{q},\ x,y,q\in\mathbb{Z}.\ \underline{\Delta}=\langle j,\langle \underline{x},\underline{y},\underline{q}\rangle\rangle \wedge \Delta=\langle i,\langle x,y,q\rangle\rangle \wedge \tilde{I}_i(\underline{x},\underline{y},x,y,q)]$$

qui satisfont les conditions (a), (b), (c), (d), (e), (f), (k) mais pas (g), (i), (j) et (k). (Avec ce deuxième choix et contrairement au premier, il n'est pas supposé que $I(\underline{\Delta},\Delta) \Rightarrow \varepsilon(\underline{\Delta})$).

□

4.3.1.7 Dérivation de conditions de vérificationn correctes

Etant donnés un principe d'induction correct et sémantiquement complet

$[\exists I \in As.\ Cv(I)]$

et une correspondance $\gamma \in (\tilde{As} \to As)$ entre $\tilde{As}$ et As, toute preuve

$[\exists \tilde{I} \in \tilde{As}.\ \tilde{Cv}(\tilde{I})]$

telle que

$\tilde{Cv} \Rightarrow Cv \circ \gamma$

est correcte. On peut donc choisir $\tilde{Cv}[\![Pr,P]\!]$ comme étant $Cv[\![S[\![Pr]\!], A[\![Pr]\!], \Sigma[\![Pr]\!], t[\![Pr]\!], \varepsilon[\![Pr]\!], \psi[\![Pr,P]\!]]\!] \circ \gamma[\![Pr,P]\!]$. Par diverses manipulations algébriques on cherchera à exprimer cette condition sous forme d'une conjonction de conditions plus simples correspondant chacune à une commande élémentaire du programme Pr. Des simplifications sont possibles puisque c'est une implication et non pas une égalité qui est réquise. La méthode de preuve obtenue de cette manière est correcte par construction. Pourque le résultat soit valable non pas pour un programme Pr particulier mais pour le langage Pr considéré il faut procéder par induction sur la syntaxe du langage.

Exemple 4.3.1.7-1

Dans le cas des principes d'induction pour l'invariance, la condition de vérification

$[\exists I \in As.\ Cv(I)]$

peut s'écrire sous forme d'une conjonction :

$[\exists I \in As.\ Cv_\varepsilon(I) \wedge Cv_i(I) \wedge Cv_\delta(I)]$

Par exemple pour le principe d'induction (-Y), nous avons :

$$Cv_\varepsilon(I) = [\forall \underline{\lambda} \in S.\ \varepsilon(\underline{\lambda}) \Rightarrow I(\underline{\lambda}, \underline{\lambda})]$$

$$Cv_i(I) = [\forall \underline{\lambda}, \lambda \in S.\ [\exists \lambda' \in S, a \in A.\ \varepsilon(\underline{\lambda}) \wedge I(\underline{\lambda}, \lambda') \wedge t_a(\lambda', \lambda)] \Rightarrow I(\underline{\lambda}, \lambda)]$$

$$Cv_\delta(I) = [\forall \underline{\lambda}, \bar{\lambda} \in S.\ [\varepsilon(\underline{\lambda}) \wedge I(\underline{\lambda}, \bar{\lambda})] \Rightarrow \psi(\underline{\lambda}, \bar{\lambda})]$$

Dans ce cas, on choisira le plus souvent

$$\tilde{Cv}(\tilde{I}) = \tilde{Cv}_\varepsilon(\tilde{I}) \wedge \tilde{Cv}_i(\tilde{I}) \wedge \tilde{Cv}_\delta(\tilde{I})$$

satisfaisant :

$$\tilde{Cv}_\varepsilon(\tilde{I}) \Rightarrow Cv_\varepsilon(\gamma(\tilde{I}))$$

$$\tilde{Cv}_i(\tilde{I}) \Rightarrow Cv_i(\gamma(\tilde{I}))$$

$$\tilde{Cv}_\delta(\tilde{I}) \Rightarrow Cv_\delta(\gamma(\tilde{I}))$$

De plus dans le cas des principes d'induction pour l'invariance le terme $Cv_i(I)$ est de la forme $F(I) \Rightarrow I$ (ou dualement $I \Rightarrow F(I)$).
Par exemple, dans le cas du principe d'induction (-Y), nous avons

$$Cv_i(I) = [F(I) \Rightarrow I]$$

où

$$F(I)(\underline{\lambda}, \lambda) = [\exists \lambda' \in S, a \in A.\ \varepsilon(\underline{\lambda}) \wedge I(\underline{\lambda}, \lambda') \wedge t_a(\lambda', \lambda)]$$

ou bien, nous pourrons intégrer le terme $Cv_\varepsilon(I)$ dans $Cv_i(I)$ ce qui donne

$$F(I)(\underline{\lambda}, \lambda) = [\exists \lambda' \in S, a \in A.\ \varepsilon(\underline{\lambda}) \wedge ([\lambda = \underline{\lambda}] \vee [I(\underline{\lambda}, \lambda') \wedge t_a(\lambda', \lambda)])]$$

Dans les deux cas la condition $\tilde{Cv}_i(\tilde{I}) \Rightarrow Cv_i(\gamma(\tilde{I}))$ s'écrit :

$$\tilde{Cv}_i(\tilde{I}) \Rightarrow [F(\gamma(\tilde{I})) \Rightarrow \gamma(\tilde{I})]$$

quand (α, γ) est une demi-correspondance de Galois (cf. 4.3.1.6.2.2), on est amené à choisir

$$\tilde{Cv}_i(\tilde{I}) \Rightarrow [\alpha \circ F \circ \gamma(\tilde{I}) \sqsubseteq \tilde{I}]$$

car $\alpha \circ F \circ \gamma(\tilde{I}) \sqsubseteq \tilde{I}$ implique que $F \circ \gamma(\tilde{I}) \Rightarrow \gamma(\tilde{I})$. Posons $\underset{\sim}{F} = \alpha \circ F \circ \gamma$. A la suite de Cousot-Cousot [80c], observons que si (α, γ) est une correspondance de Galois injective (cf. 4.3.1.6.2.7) alors $\alpha \circ F = \underset{\sim}{F} \circ \alpha$, (en effet, α étant injective nous avons $\gamma \circ \alpha = 1_{\underline{\underline{m}}}$ et donc $\alpha \circ F = \alpha \circ F \circ \gamma \circ \alpha = \underset{\sim}{F} \circ \alpha$).

Supposons maintenant que (α n'étant pas injective), il existe un opérateur $\tilde{F}$ sur $A\tilde{s}$ tel que $\alpha \circ F = \tilde{F} \circ \alpha$. Nous avons alors

[$\underset{\sim}{F} \sqsubseteq \tilde{F}$, l'inégalité peut être stricte et il y a égalité si (mais pas seulement si) α est surjective]

(En effet $\underset{\sim}{F} = \alpha \circ F \circ \gamma = \tilde{F} \circ \alpha \circ \gamma \sqsubseteq \tilde{F}$ car $\alpha \circ \gamma$ est réductive. L'inégalité peut être stricte comme le montre l'exemple suivant : $As = A\tilde{s} = \{a, b\}$, $a \sqsubset b$, $\alpha(a) = \alpha(b) = a$, $\gamma(a) = \gamma(b) = b$, $F(x) = x$, $\tilde{F}(\tilde{x}) = \tilde{x}$, $\underset{\sim}{F}(a) = \underset{\sim}{F}(b) = a$. Si α est surjective alors $\alpha \circ \gamma$ est l'identité et donc $\underset{\sim}{F} = \tilde{F}$ mais la réciproque n'est pas vraie comme le montre l'exemple suivant : $As = A\tilde{s} = \{a, b\}$, $a \sqsubset b$, $F(x) = \underset{\sim}{F}(x) = \tilde{F}(x) = \alpha(x) = a$, $\gamma(x) = b$.)

Dans ces conditions, on peut choisir

$C\tilde{o}\underset{\sim}{v}(\tilde{I}) = [\bar{F}(\tilde{I}) \sqsubseteq \tilde{I}]$

où $\bar{F}$ est un opérateur sur $A\tilde{s}$ tel que $\underset{\sim}{F} \sqsubseteq \bar{F} \sqsubseteq \tilde{F}$ (puisque $\bar{F}(\tilde{I}) \sqsubseteq \tilde{I}$ implique $\underset{\sim}{F}(\tilde{I}) \sqsubseteq \tilde{I}$ donc $\alpha \circ F \circ \gamma(\tilde{I}) \sqsubseteq \tilde{I}$ soit $F \circ \gamma(\tilde{I}) \Rightarrow \gamma(\tilde{I})$ et donc $C\tilde{o}\underset{\sim}{v}(\gamma(\tilde{I}))$). Si α n'est pas surjective le treillis complet des $\bar{F} \in (A\tilde{s} \to A\tilde{s})$ tels que $\underset{\sim}{F} \sqsubseteq \bar{F} \sqsubseteq \tilde{F}$ n'est pas réduit à un seul élément de sorte qu'il est possible d'envisager diverses conditions de vérifications.

Parmi ces conditions de vérification, $\tilde{F}$ peut nécessiter de trouver un invariant $\tilde{I}$ plus fort que pour $\underset{\sim}{F}$ puisque $\tilde{F}(\tilde{I}) \sqsubseteq \tilde{I}$ implique $\underset{\sim}{F}(\tilde{I}) \sqsubseteq \tilde{I}$ mais la réciproque n'est pas vraie. Cet argument en faveur du choix de $\underset{\sim}{F}$ doit être modulé par le fait qu'en pratique $\tilde{F}$ conduit à des conditions de vérification plus simples. En pratique, un équilibre doit être trouvé entre des invariants plus faibles et des conditions de vérification plus compliquées ou des invariants plus forts et des conditions de vérification plus simples.

□

4.3.1.8 Vérification de la complétude sémantique

La vérification de la complétude sémantique consiste à montrer que :

$$[\exists I \in A_S.\ C_v(I)] \Rightarrow [\exists \tilde{I} \in A_{\tilde{S}}.\ C_{\tilde{v}}(\tilde{I})]$$

Si nous choisissons $\tilde{I}$ comme étant la représentation de I par $\alpha \in (A_S \to A_{\tilde{S}})$, nous obtenons la condition suffisante de complétude :

$$C_v \Rightarrow C_{\tilde{v}} \circ \alpha$$

(celle-ci étant d'ailleurs nécésaire pour un α convenablement choisi).

Exemple 4.3.1.8-1

Dans le cas particulier que nous rencontrerons assez souvent où $\gamma \circ \alpha = \underline{1}$ (cf. 4.3.1.6.2.C, 4.3.1.6.2.7) et $C_{\tilde{v}} = C_v \circ \gamma$ auquel cas la condition suffisante de correction $C_{\tilde{v}} \Rightarrow C_v \circ \gamma$ est trivialement satisfaite, nous avons $C_{\tilde{v}} \circ \alpha = C_v \circ \gamma \circ \alpha = C_v$ et la condition suffisante de complétude $C_v \Rightarrow C_{\tilde{v}} \circ \alpha$ est également trivialement satisfaite.

□

Exemple 4.3.1.8-2

Pour poursuivre l'exemple 4.3.1.7-1, dans le cas où

$$C_v(I) = [F(I) \Rightarrow I \ \wedge\ I \Rightarrow \psi]$$

$$C_{\tilde{v}}(\tilde{I}) = [\bar{F}(\tilde{I}) \sqsubseteq \tilde{I} \ \wedge\ \tilde{I} \sqsubseteq \alpha(\psi)]$$

où α est monotone, $\bar{F} \sqsubseteq \tilde{F}$ et $\alpha \circ F = \tilde{F} \circ \alpha$, il faut vérifier que :

$$[F(I) \Rightarrow I \ \wedge\ I \Rightarrow \psi] \Rightarrow [\bar{F}(\alpha(I)) \sqsubseteq \alpha(I) \ \wedge\ \alpha(I) \sqsubseteq \alpha(\psi)]$$

Nous avons bien $(I \Rightarrow \psi)$ qui implique $\alpha(I) \sqsubseteq \alpha(\psi)$ par monotonie de sorte qu'il suffit de vérifier que

$$[F(I) \Rightarrow I] \Rightarrow [\bar{F}(\alpha(I)) \sqsubseteq \alpha(I)]$$

ce qui est évident car $[F(I) \Rightarrow I] \Rightarrow [\alpha \circ F(I) \sqsubseteq \alpha(I)]$ (par monotonie) $\Rightarrow [\tilde{F}(\alpha(I)) \sqsubseteq \alpha(I)]$ (car $\alpha \circ F = \tilde{F} \circ \alpha$) $\Rightarrow [\bar{F}(\alpha(I)) \sqsubseteq \alpha(I)]$ (car $\bar{F} \sqsubseteq \tilde{F}$).

□

4.3.2 EXEMPLES DE CONSTRUCTIONS

4.3.2.1 Construction d'une méthode de preuve de non-terminaison, d'absence d'erreurs à l'exécution et d'invariance globale de programmes séquentiels par l'absurde

4.3.2.1.1 La non-terminaison est une propriété d'invariance

Soit $Ps \in \mathcal{P}s$ un programme séquentiel et $\phi \in ((\mathcal{V} \to \mathcal{D}) \to \{tt, ff\})$ une caractérisation des valeurs possibles des variables à l'entrée du programme. Le programme Ps ne se termine pas normalement si aucun état de sortie ne peut être atteint durant l'exécution :

$$\forall p \in \Sigma\langle S[\![Ps]\!], A[\![Ps]\!], t[\![Ps]\!], \varepsilon\rangle, i \in |p|. \; [\sigma(p_i) \Rightarrow \psi(p_i)]$$

où

$$\varepsilon(\langle L, M\rangle) = [(Ps \equiv L:p) \wedge \phi(M)]$$

$$\sigma(\langle L, M\rangle) = tt$$

$$\psi(\langle L, M\rangle) = [\neg(Ps \equiv pL:)]$$

4.3.2.1.2 Choix d'un principe d'induction

Puisque ψ est une assertion sur les états finaux, d'après le paragraphe 4.2.1.3, nous pourrons utiliser les principes d'induction $(\text{-}i\text{-})$, $(\text{-}\tilde{i}\text{-})$, $(\text{-}\bar{i}\text{-})$, $(\text{-}\tilde{\bar{i}}\text{-})$. Nous choisirons $(\text{-}\tilde{\bar{i}}\text{-})$ comme exemple car ce principe d'induction n'est pas du tout conventionnel. $(\text{-}\tilde{I}\text{-})$ est de la forme :

$$[\exists I \in A_D[\![Ps]\!]. \; C_V[\![Ps]\!](\varepsilon, \sigma)(\psi)(I)]$$

où

$$As[\![Ps]\!] = (S[\![Ps]\!] \to \{tt, ff\})$$

$$Cv[\![Ps]\!](\varepsilon,\sigma)(\psi)(I) = Cv_\sigma[\![Ps]\!](\psi,\sigma)(I) \wedge Cv_\iota[\![Ps]\!](I) \wedge Cv_\varepsilon[\![Ps]\!](\varepsilon)(I)$$

$$Cv_\sigma[\![Ps]\!](\psi,\sigma)(I) = [\forall \bar{\Delta} \in S[\![Ps]\!] . [\neg\psi(\bar{\Delta}) \wedge \sigma(\bar{\Delta})] \Rightarrow I(\bar{\Delta})]$$

$$Cv_\iota[\![Ps]\!](I) = [\forall \Delta \in S[\![Ps]\!], a \in A[\![Ps]\!] . I(\Delta) \Rightarrow \neg[\exists \Delta' \in S[\![Pt]\!] . \neg I(\Delta') \wedge t[\![Ps]\!]_a(\Delta',\Delta)]]$$

$$Cv_\varepsilon[\![Ps]\!](\varepsilon)(I) = [\forall \underline{\Delta} \in S[\![Ps]\!] . \varepsilon(\underline{\Delta}) \Rightarrow \neg I(\underline{\Delta})]$$

4.3.2.1.3 Choix d'un langage pour exprimer les invariants locaux

Nous décidons de représenter un invariant global $I \in As[\![Ps]\!]$ par un vecteur $\tilde{I}$ d'invariants locaux sur les états mémoires et associés à chaque point du programme. Alors

$$\tilde{As}[\![Ps]\!] = \prod_{L \in C[\![Ps]\!]} (M[\![Ps]\!] \to \{tt, ff\})$$

où

$$C[\![Ps]\!] = \{L \in \mathcal{L} . Ps \equiv \beta L : \delta\}$$

$$M[\![Ps]\!] = (\{X \in \mathcal{V} : Ps \equiv \alpha X \beta\} \to \mathcal{D})$$

La signification de ce vecteur d'invariants locaux est que l'invariant $\tilde{I}(L)(M)$ associé au point L est vrai pour l'état mémoire M de tout état $\langle L, M\rangle$ du programme ayant L comme état de contrôle. Plus formellement

$$\chi[\![Ps]\!] \in (\tilde{As}[\![Ps]\!] \to As[\![Ps]\!])$$

$$\chi[\![Ps]\!](\tilde{I})(\langle L, M\rangle) = \tilde{I}(L)(M)$$

Réciproquement, un invariant global $I \in As[\![Ps]\!]$ est représenté par $\alpha[\![Ps]\!](I) \in \tilde{As}[\![Ps]\!]$, c'est-à-dire un vecteur $\tilde{I}$ d'invariants locaux sur les états mémoire M tels que :

$$\forall L \in C[\![Ps]\!] . \tilde{I}(L)(M) = I(\langle L, M\rangle)$$

4.3.2.1.4 Dérivation de conditions de vérification correctes

Nous avons à construire $C\tilde{v}[\![Ps]\!]$ tel que :

$$C\tilde{v}[\![Ps]\!](\varepsilon,\sigma)(\psi)(\tilde{I}) \Rightarrow Cv[\![Ps]\!](\varepsilon,\sigma)(\gamma[\![Ps]\!](\tilde{I}))$$

Puisque $Cv[\![Ps]\!]$ est une conjonction de trois conditions de vérification, nous choisissons $C\tilde{v}[\![Ps]\!]$ de la même forme, c'est-à-dire :

$$C\tilde{v}[\![Ps]\!](\varepsilon,\sigma)(\psi)(\tilde{I}) = C\tilde{v}_\sigma[\![Ps]\!](\psi,\sigma)(\tilde{I}) \wedge C\tilde{v}_i[\![Ps]\!](\tilde{I}) \wedge C\tilde{v}_\varepsilon[\![Ps]\!](\varepsilon)(\tilde{I})$$

Pour garantir la condition de correction ci-dessus, nous choisirons simplement :

$$C\tilde{v}_\sigma[\![Ps]\!](\psi,\sigma)(\tilde{I}) = Cv_\sigma[\![Ps]\!](\psi,\sigma)(\gamma[\![Ps]\!](\tilde{I}))$$

$$C\tilde{v}_i[\![Ps]\!](\tilde{I}) = Cv_i[\![Ps]\!](\gamma[\![Ps]\!](\tilde{I}))$$

$$C\tilde{v}_\varepsilon[\![Ps]\!](\varepsilon)(\tilde{I}) = Cv_\varepsilon[\![Ps]\!](\varepsilon)(\gamma[\![Ps]\!](\tilde{I}))$$

4.3.2.1.4.1 Base

$$C\tilde{v}_\sigma[\![Ps]\!](\psi,\sigma)(\tilde{I})$$

$$= Cv_\sigma[\![Ps]\!](\psi,\sigma)(\gamma[\![Ps]\!](\tilde{I}))$$

$$= [\forall \tilde{s} \in S[\![Ps]\!].\ [\neg\psi(\tilde{s}) \wedge \sigma(\tilde{s})] \Rightarrow \gamma[\![Ps]\!](\tilde{I})(\tilde{s})]$$

Puisque $S[\![Ps]\!] = C[\![Ps]\!] \times M[\![Ps]\!]$, nous avons :

$$= [\forall L \in C[\![Ps]\!], M \in M[\![Ps]\!].\ [\neg\psi(\langle L,M\rangle) \wedge \sigma(\langle L,M\rangle)] \Rightarrow \gamma[\![Ps]\!](\tilde{I})(\langle L,M\rangle)]$$

Remplaçons σ, ψ et γ par leurs définitions :

$$= [\forall L \in C[\![Ps]\!], M \in M[\![Ps]\!].\ (Ps \equiv \beta L:) \Rightarrow \tilde{I}(L)(M)]$$

Ps a un et un seul point de sortie d'où :

$$= [(Ps \equiv \beta L:) \wedge \forall M \in M[\![Ps]\!].\ \tilde{I}(L)(M)]$$

4.3.2.1.4.2 Induction

$$\tilde{Cv}_i[\![Ps]\!](\tilde{I})$$

$$= Cv_i[\![Ps]\!](\gamma[\![Ps]\!](\tilde{I}))$$

$$= [\forall \Delta \in S[\![Ps]\!], a \in A[\![Ps]\!].\ \gamma[\![Ps]\!](\tilde{I})(\Delta) \Rightarrow \neg[\exists \Delta' \in S[\![Ps]\!].\ \neg\gamma[\![Ps]\!](\tilde{I})(\Delta') \wedge t[\![Ps]\!]_a(\Delta', \Delta)]]$$

$$= [\forall \Delta \in S[\![Ps]\!], a \in A[\![Ps]\!].\ \gamma[\![Ps]\!](\tilde{I})(\Delta) \Rightarrow [\forall \Delta' \in S[\![Ps]\!].\ t[\![Ps]\!]_a(\Delta', \Delta) \Rightarrow \gamma[\![Ps]\!](\tilde{I})(\Delta')]]$$

$$= [\forall L \in C[\![Ps]\!], M \in M[\![Ps]\!], a \in A[\![Ps]\!].\ \tilde{I}(L)(M) \Rightarrow [\forall L' \in C[\![Ps]\!], M' \in M[\![Ps]\!].$$
$$t[\![Ps]\!]_a(\langle L', M'\rangle, \langle L, M\rangle) \Rightarrow \tilde{I}(L')(M')]]$$

$$= \bigwedge_{L \in C[\![Ps]\!]} [\forall M \in M[\![Ps]\!].\ \tilde{I}(L)(M) \Rightarrow [\forall L' \in C[\![Ps]\!], M' \in M[\![Ps]\!].$$
$$(\underset{\sim}{cond}[\![Ps]\!](L', L)(M') \wedge \underset{\sim}{succ}[\![Ps]\!](L')(M', M)) \Rightarrow \tilde{I}(L')(M')]]$$

Nous avons une conjonction de conditions de vérification, une pour chaque point du programme, et de la forme :

$$\tilde{I}(L)(M) \Rightarrow [\forall L' \in C[\![Ps]\!], M' \in M[\![Ps]\!].[\underset{\sim}{cond}[\![Ps]\!](L', L)(M') \wedge \underset{\sim}{succ}[\![Ps]\!](L')(M', M)] \Rightarrow \tilde{I}(L')(M')]$$

Cette condition peut se décomposer aux différents cas correspondant à la définition de $\underset{\sim}{cond}[\![Ps]\!]$:

(a) Si $(Ps \equiv L : \beta)$ alors L désigne le point d'entrée du programme, alors $\forall L' \in C[\![Ps]\!], M' \in M[\![Ps]\!].\ \neg \underset{\sim}{cond}[\![Ps]\!](L', L)(M')$ et la condition de vérification est donc vraie trivialement.

(b) Si $(Ps \equiv \beta L' : \underline{skip}\ ; L : \delta)$ ou $(Ps \equiv \beta L' : \underline{else}\ Ps'\ \underline{fi}\ ; L : \delta)$ ou $(Ps \equiv \beta L' : \underline{fi}\ ; L : \delta)$ alors $\underset{\sim}{cond}[\![Ps]\!](L', L)(M)$ est vrai et $\underset{\sim}{succ}[\![Ps]\!](L')(M', M)$ implique $M = M'$ de sorte que la condition de vérification peut être simplifié en :

$$\tilde{I}(L)(M) \Rightarrow \tilde{I}(L')(M)$$

(c) Si $(Ps \equiv \beta L' : V := E\ ; L : \delta)$ alors nous devons vérifier que

$$\tilde{I}(L)(M) \Rightarrow [\forall M' \in M[\![Ps]\!].\ [M' \in \underset{\sim}{dom}(\mathbb{E}[\![E]\!]) \wedge M = M'[V \leftarrow \mathbb{E}[\![E]\!](M')] \Rightarrow \tilde{I}(L')(M')]$$

Noter que M' doit être de la forme $M[V \leftarrow m]$ où $m \in \mathcal{D}$ est la valeur de V avant l'affectation. Alors cette condition peut se simplifier en :

$$\tilde{I}(L)(M) \Rightarrow [\forall m \in \mathcal{D}.\ [M[V \leftarrow m] \in \underset{\sim}{dom}(\mathbb{E}[\![E]\!]) \wedge M(V) = \mathbb{E}[\![E]\!](M[V \leftarrow m])] \Rightarrow \tilde{I}(L')(M[V \leftarrow m])]$$

Si $(Ps \equiv \beta L' : V := ?\ ; L : \delta)$, nous obtenons de même :

$$\tilde{I}(L)(M) \Rightarrow [\forall m \in \mathcal{D}.\ \tilde{I}(L')(M[V \leftarrow m])]$$

(d) Si ($Ps \equiv \beta L'$: *if* B *then* $L:\delta$) ou ($Ps \equiv \beta L'$: *while* B *do* $L:\delta$) ou ($Ps \equiv \beta$ *while* B *do* $L: C_0; \ldots; L_m: C_m; L'$: *od*; δ) nous obtenons

$$\tilde{I}(L)(M) \Rightarrow [(M \in \underset{\sim}{dom}(\mathbb{B}[\![B]\!]) \wedge \mathbb{B}[\![B]\!](M) = tt) \Rightarrow \tilde{I}(L')(M)]$$

(e) Si ($Ps \equiv \beta L'$: *if* B *then* Ps' *else* $L:\delta$) ou ($Ps \equiv \beta L'$: *while* B *do* Ps' *od*; $L:\delta$) ou ($Ps \equiv \alpha$ *while* B *do* $L_0: C_0; \ldots; L_m: C_m; L'$: *od*; $L:\delta$)

$$\tilde{I}(L)(M) \Rightarrow [(M \in \underset{\sim}{dom}(\mathbb{B}[\![B]\!]) \wedge \mathbb{B}[\![B]\!](M) = ff) \Rightarrow \tilde{I}(L')(M)]$$

4.3.2.1.4.3 Contradiction

$$\begin{aligned}
&C\tilde{v}_{\varepsilon}\,[\![Ps]\!](\varepsilon)(\tilde{I}) \\
&\quad = C v_{\varepsilon}\,[\![Ps]\!](\varepsilon)(\gamma[\![Ps]\!](\tilde{I})) \\
&\quad = [\forall \underline{\Delta} \in S[\![Ps]\!].\ \varepsilon(\underline{\Delta}) \Rightarrow \neg\gamma[\![Ps]\!](\tilde{I})(\underline{\Delta})] \\
&\quad = [\forall L \in C[\![Ps]\!], M \in M[\![Ps]\!].\ [(Ps \equiv L:\beta) \wedge \phi(M)] \Rightarrow \neg\tilde{I}(L)(M)] \\
&\quad = [(Ps \equiv L:\beta) \wedge (\forall M \in M[\![Ps]\!].\ \phi(M) \Rightarrow \neg\tilde{I}(L)(M))]
\end{aligned}$$

4.3.2.1.5 Résumé informel des conditions de vérification

En utilisant des notations mnémoniques, nous pourrons récapituler les conditions de vérification ci-dessus comme suit (P_i est l'invariant sur les variables du programme associé au point L_i) :

- Base

$\beta\ L_1$: P_1

- Induction

. Commande nulle

$\beta\ L_1$: skip ; L_2 : δ $\quad P_2 \Rightarrow P_1$

. Commande d'affectation

$\beta\ L_1$: $V := E$; L_2 : δ $\quad P_2 \Rightarrow [\forall m \in \mathcal{D}.\ V=E[V \leftarrow m] \Rightarrow P_1[V \leftarrow m]]$

$\beta\ L_1$: $V := ?$; L_2 : δ $\quad P_2 \Rightarrow [\forall m \in \mathcal{D}.\ P_1[V \leftarrow m]]$

. Commande conditionnelle

$\beta\ L_1$: if B then

L_2 : δ $\quad P_2 \Rightarrow [B \Rightarrow P_1]$

L_3 :

else

L_4 : δ' $\quad P_4 \Rightarrow [\neg B \Rightarrow P_1]$

L_5 :

fi ;

L_6 : β' $\quad P_6 \Rightarrow [P_3 \wedge P_5]$

. Commande itérative

$\beta\ L_1$: while B do

L_2 : δ $\quad P_2 \Rightarrow [B \Rightarrow (P_1 \wedge P_3)]$

L_3 :

od ;

L_4 : β' $\quad P_4 \Rightarrow [\neg B \Rightarrow (P_1 \wedge P_3)]$

- Contradiction

L_1 : β $\quad \phi \Rightarrow \neg P_1$

4.3.2.1.6 Exemple de preuve avec cette méthode

Soit à prouver que le programme suivant

```
1:
    q := 0 ;
2:
    while x ≥ y do
3:
        q := q+1 ;
4:
        x := x-y ;
5:
    od ;
6:
```

ne se termine pas normalement quand les valeurs initiales $\underline{x}, \underline{y}$ des variables x, y sont telle que $\underline{x} \geq 0$ et $\underline{y} = 0$. Nous supposons que le domaine $\mathcal{D}$ de valeurs de chaque variable est l'ensemble des entiers compris entre deux bornes min et max. Ces bornes sont supposées telles que $min < 0 < max$.

Les conditions de vérification (après des simplifications triviales) sont les suivantes (où $x, y, q \in [min, max]$) :

$P_6(x,y,q)$

$P_6(x,y,q) \Rightarrow [(x<y) \Rightarrow (P_2(x,y,q) \wedge P_5(x,y,q))]$

$P_5(x,y,q) \Rightarrow P_4(x+y, y, q)$

$P_4(x,y,q) \Rightarrow P_3(x,y,q-1)$

$P_3(x,y,q) \Rightarrow [(x \geq y) \Rightarrow (P_2(x,y,q) \wedge P_5(x,y,q))]$

$P_2(x,y,q) \Rightarrow [\forall q' \in [min, max].\ (q=0) \Rightarrow P_1(x,y,q')]$

$[(0 \leq x \leq max) \wedge (y=0)] \Rightarrow \neg P_1(x,y,q)$

Intuitivement, par l'absurde, si $y=0$ le programme ne peut se terminer que si $x<0$, en contradiction avec l'hypothèse sur la valeur initiale de x. Ceci peut se démontrer formellement en utilisant les invariants suivants :

$P_i(x,y,q) = [(y=0) \Rightarrow (x<0)]$ pour $i = 1, \ldots, 5$

$P_6(x,y,q) = tt$

4.3.2.1.7 Vérification de la complétude sémantique

Conformément au paragraph 7.5, nous devons vérifier que

$$\forall I \in As[\![Ps]\!].\ Cv[\![Ps]\!](\varepsilon,\sigma)(\psi)(I) \Rightarrow C\tilde{v}[\![Ps]\!](\varepsilon,\sigma)(\psi)(\alpha[\![Ps]\!](I))$$

En utilisant le fait que $C\tilde{v}[\![Ps]\!](\varepsilon,\sigma)(\psi)(\tilde{I}) = Cv[\![Ps]\!](\varepsilon,\sigma)(\psi)(\gamma[\![Ps]\!](\tilde{I}))$, il est suffisant de vérifier que $\gamma[\![Ps]\!](\alpha[\![Ps]\!](I)) = I$, qui est trivial car $\gamma[\![Ps]\!]$ est une bijection entre $\tilde{As}[\![Ps]\!]$ et $As[\![Ps]\!]$ (et $\alpha[\![Ps]\!]$ est son inverse).

4.3.2.1.8 Preuve d'absence d'erreurs à l'exécution, par l'absurde pour des programmes séquentiels

Soit $Ps \in \mathcal{P}s$ un programme séquentiel et $\phi \in ((\mathcal{V} \to \mathcal{D}) \to \{tt, ff\})$ une caractérisation des valeurs initiales possibles des variables du programme. L'exécution du programme Ps ne conduit pas à une erreur d'exécution si et seulement si tout état atteint durant l'exécution et qui n'est pas un état de sortie a un état successeur, c'est-à-dire

$$\forall p \in \Sigma^{<} s[\![Ps]\!], A[\![Ps]\!], t[\![Ps]\!], \varepsilon^{>},\ i \in |p|.\ [\sigma(p_i) \Rightarrow \psi(p_i)]$$

où

$$\varepsilon(\langle L, M\rangle) = [(Ps \equiv L:\beta) \wedge \phi(M)]$$

$$\sigma(\langle L, M\rangle) = [\neg(Ps \equiv \alpha L:)]$$

$$\psi(\langle L, M\rangle) = [\exists L' \in C[\![Ps]\!], M' \in M[\![Ps]\!], a \in A[\![Ps]\!].\ t[\![Ps]\!]_a(\langle L, M\rangle, \langle L', M'\rangle)]$$

avec $M[\![Ps]\!] = (\{x \in \mathcal{V}:\ Ps \equiv \alpha x \beta\} \to \mathcal{D})$, $C[\![Ps]\!] = \{L \in \mathcal{L}:\ Ps \equiv \beta L:\delta\}$

Noter que la seule différence avec le paragraphe 4.3.2.1.1 est σ et ψ. Alors, en choisissant le principe d'induction et le langage, pour exprimer les invariants locaux, considérés aux paragraphes 4.3.2.1.2 et 4.3.2.1.3, nous obtenons les mêmes conditions de vérification qu'en 4.3.2.1.4 excepté pour la base :

$$\tilde{Cv}_\sigma [\![Ps]\!](\psi,\sigma)(\tilde{I})$$

$$= Cv_\sigma [\![Ps]\!](\psi,\sigma)(\gamma[\![Ps]\!](\tilde{I}))$$

$$= [\forall \bar{s} \in S[\![Ps]\!] . [\neg\psi(\bar{s}) \wedge \sigma(\bar{s})] \Rightarrow \gamma[\![Ps]\!](\tilde{I})(\bar{s})]$$

$$= [\forall L \in C[\![Ps]\!],\ M \in M[\![Ps]\!] . [\neg(Ps \equiv \beta L:) \wedge (\forall L' \in C[\![Ps]\!], M' \in M[\![Ps]\!], a \in A[\![Ps]\!] .$$
$$\neg t[\![Ps]\!]_a(\langle L,M\rangle,\langle L',M'\rangle)] \Rightarrow \tilde{I}(L)(M)]$$

Alors pour tous les points L du programme Ps excepté l'état de sortie, nous devons prouver :

$$[\forall L' \in C[\![Ps]\!], M' \in M[\![Ps]\!], a \in A[\![Ps]\!] .\ \neg t[\![Ps]\!]_a(\langle L,M\rangle,\langle L',M'\rangle)] \Rightarrow \tilde{I}(L)(M)$$

Cette condition se décompose aux différents cas correspondant à $t[\![Ps]\!]_a$:

(a) Si $(Ps \equiv \beta L: \underline{skip}\ ; L':\delta)$ ou $(P \equiv \beta L: \underline{else}\ Ps'\ \underline{fi}\ ; L':\delta)$ ou $(Ps \equiv \beta L: \underline{fi}\ ; L':\delta)$ ou $(Ps \equiv \beta L: V:=?\ ; L':\delta)$ alors le membre de gauche est faux et la condition de vérification est identiquement vraie.

(b) Si $(Ps \equiv \beta L: V:=E\ ; L':\delta)$ alors nous obtenons :

$$[M \notin \underset{\sim}{dom}(\mathbb{E}[\![E]\!])] \Rightarrow \tilde{I}(L)(M)$$

(c) Si $(Ps \equiv \beta L: \underline{if}\ B\ \underline{then}\ \delta)$ ou $(Ps \equiv \beta L: \underline{while}\ B\ \underline{do}\ \delta)$ ou $(Ps \equiv \beta\ while\ B\ do\ L_0:C_0\ ;\ldots;\ L_m:C_m\ ;\ L:od\ ;\ \delta)$ alors nous obtenons

$$[M \notin \underset{\sim}{dom}(\mathbb{B}[\![B]\!])] \Rightarrow \tilde{I}(L)(M)$$

Nous pouvons récapatiler ces conditions de vérifications comme suit :

- Commande nulle

$\beta\ L_1 : \underline{skip}\ ;\ L_2 : \delta$ $\qquad P_2 \Rightarrow P_1$

- Commande d'affectation

$\beta\ L_1 : V := E\ ;\ L_2 : \delta$ $\qquad \neg\, \underline{dom}(E) \Rightarrow P_1$

$\qquad P_2 \Rightarrow [\forall m \in \mathcal{D} \,.\, V = E[V \leftarrow m] \Rightarrow P_1[V \leftarrow m]]$

$\beta\ L_1 : V := ?\ ;\ L_2 : \delta$ $\qquad P_2 \Rightarrow [\forall m \in \mathcal{D} \,.\, P_1[V \leftarrow m]]$

- Commande conditionnelle

$\beta\ L_1 :$ $\underline{if}\ B\ \underline{then}$ $\qquad \neg\, \underline{dom}(B) \Rightarrow P_1$

$\quad L_2 :$ $\qquad P_2 \Rightarrow [B \Rightarrow P_1]$

$\quad\ \delta$

$\quad L_3 :$

$\underline{else}$

$\quad L_4 :$ $\qquad P_4 \Rightarrow [\neg B \Rightarrow P_1]$

$\quad\ \delta'$

$\quad L_5 :$

$\underline{fi}$

$L_6 : \beta'$ $\qquad P_6 \Rightarrow [P_3 \wedge P_5]$

- Commande itérative

$\beta\ L_1 :$ $\underline{while}\ B\ \underline{do}$ $\qquad \neg\, \underline{dom}(B) \Rightarrow P_1$

$\quad L_2 :$ $\qquad P_2 \Rightarrow [B \Rightarrow (P_1 \wedge P_3)]$

$\quad\ \delta$

$\quad L_3 :$

$\underline{od}\ ;$

$L_4 : \beta'$ $\qquad P_4 \Rightarrow [\neg B \Rightarrow (P_1 \wedge P_3)]$

- Point d'entrée

$L_1 : \beta$ $\qquad \phi \Rightarrow \neg P_1$

Pour notre programme pris comme exemple

```
1:
    q := 0;
2:
    while x>=y do
3:
        q := q+1;
4:
        x := x-y;
5:
    od;
6:
```

nous pouvons montrer qu'il n'y a pas d'erreurs à l'exécution quand les valeurs initiales $\underline{x}, \underline{y}$ de x, y sont telles que $\underline{x} \geq 0 \wedge \underline{y} > 0$ et $\mathcal{D} = [min, max]$ avec $min < 0 < max$.

Les conditions de vérification (après des simplifications triviales) sont:

$[(q+1) > max] \Rightarrow P_3(x,y,q)$

$[((x-y) < min) \vee ((x-y) > max)] \Rightarrow P_4(x,y,q)$

$P_6(x,y,q) \Rightarrow [(x<y) \Rightarrow (P_2(x,y,q) \wedge P_5(x,y,q))]$

$P_5(x,y,q) \Rightarrow P_4(x+y, y, q)$

$P_4(x,y,q) \Rightarrow P_3(x,y,q-1)$

$P_3(x,y,q) \Rightarrow [(x \geq y) \Rightarrow (P_2(x,y,q) \wedge P_5(x,y,q))]$

$P_2(x,y,q) \Rightarrow [\forall q' \in [min, max]. (q=0) \Rightarrow P_1(x,y,q')]$

$[x \geq 0 \wedge y > 0] \Rightarrow \neg P_1(x,y,q)$

Intuitivement, si initialement $\underline{x} \geq 0$ et $\underline{y} > 0$ la seule erreur d'exécution possible est un débordement à la commande 3: q := q+1;. Puisque q reste positif ceci peut arriver seulement si la valeur initiale $\underline{x}$ de x (c'est-à-dire $q \times y + x$ en terme des valeurs courantes des variables) est supérieure à max, en contradiction avec le fait que $\underline{x} \in \mathcal{D} = [min, max]$.

Ce raisonnement est formalisé en montrant que les invariants locaux suivants satisfont les conditions de vérification (où $x, y, q \in [min, max]$):

$P_1(x,y,q) = [\neg(x \geq 0 \wedge y > 0)]$

$P_2(x,y,q) = P_5(x,y,q) = [(x \geq 0 \wedge y > 0 \wedge q \geq 0) \Rightarrow (q \times y + x > max)]$

$P_3(x,y,q) = [(x \geq y > 0 \wedge q \geq 0) \Rightarrow (q \times y + x > max)]$

$P_4(x,y,q) = [(x \geq y > 0 \wedge q \geq 1) \Rightarrow ((q-1) \times y + x > max)]$

$P_6(x,y,q) = ff$

4.3.2.1.9 Preuve d'invariance globale, par l'absurde pour des programmes séquentiels

Soit $Ps \in \mathcal{P}s$ un programme séquentiel. Un invariant global d'un programme Ps est une assertion $\delta \in ((\mathcal{V} \to \mathcal{D}) \to \{tt, ff\})$ sur les valeurs des variables qui est vraie tout le temps durant l'exécution du programme. C'est-à-dire

$$[\forall p \in \Sigma < S[\![Ps]\!], A[\![Ps]\!], t[\![Ps]\!], \varepsilon >, i \in |p|. [\sigma(p_i) \Rightarrow \psi(p_i)]$$

où

$\varepsilon(<L,M>) = [(Ps \equiv L:\beta) \wedge \phi(M)]$

$\sigma(<L,M>) = tt$

$\psi(<L,M>) = \delta(M)$

En choisissant le principe d'induction et le langage, pour exprimer les invariants locaux, considérés aux paragraphes 4.3.2.1 et 4.3.2.2, nous obtenons les conditions de vérifications du paragraphe 4.3.2.1.4 excepté pour la base :

$\tilde{Cv}_S[\![Ps]\!](\psi,\sigma)(\tilde{I})$

$= Cv_S[\![Ps]\!](\psi,\sigma)(\gamma[\![Ps]\!](\tilde{I}))$

$= [\forall \bar{s} \in S[\![Ps]\!]. [\neg\psi(\bar{s}) \wedge \sigma(\bar{s})] \Rightarrow \gamma[\![Ps]\!](\tilde{I})(\bar{s})]$

$= [\forall L \in C[\![Ps]\!], M \in M[\![Ps]\!]. \neg\delta(M) \Rightarrow \tilde{I}(L)(M)]$

qui s'écrit informellement (L_i désignant une étiquette quelconque du programme):

- Base

$\beta L_i : \beta'$ $\qquad\qquad$ $\neg\delta \Rightarrow P_i$

Pour notre programme pris comme exemple,

```
1:
    Q:=0;
2:
    while x≥y do
3:
        Q:= Q+1;
4:
        X := X-Y;
5:
    od;
6:
```

les conditions de vérification sont les suivantes (où $x,y,q \in [min, max]$) :

$$\neg\delta(x,y,q) \Rightarrow P_i(x,y,q) \qquad i=1,\dots,6$$

$$P_6(x,y,q) \Rightarrow [(x<y) \Rightarrow (P_2(x,y,q) \wedge P_5(x,y,q))]$$

$$P_5(x,y,q) \Rightarrow P_4(x+y,y,q)$$

$$P_4(x,y,q) \Rightarrow P_5(x,y,q-1)$$

$$P_3(x,y,q) \Rightarrow [(x\geq y) \Rightarrow (P_2(x,y,q) \wedge P_5(x,y,q))]$$

$$P_2(x,y,q) \Rightarrow [\forall q' \in [min,max].\ (q=0) \Rightarrow P_2(x,y,q')]$$

$$\phi(x,y,q) \Rightarrow \neg P_1(x,y,q)$$

Pour prouver que $\phi(x,y,q) = (x\geq 0 \wedge y>0 \wedge q\geq 0)$ est un invariant global de ce programme, nous pouvons choisir les invariants locaux suivants :

$$P_1(x,y,q) = P_2(x,y,q) = P_5(x,y,q) = [x<0 \vee y\leq 0 \vee q<0]$$

$$P_3(x,y,q) = P_4(x,y,q) = [(x\geq y) \Rightarrow (x<0 \vee y\leq 0 \vee q<0)]$$

$$P_6(x,y,q) = [(x<y) \Rightarrow (x<0 \vee y\leq 0 \vee q<0)]$$

4.3.2.2 Extension de la méthode de Morris-Wegbreit dite "Subgoal induction" aux programmes parallèles et généralisation à d'autres propriétés d'invariance

La méthode de Morris-Wegbreit [77] a été conçue pour démontrer la correction partielle de programmes séquentiels. Etant données des spécifications d'entrée $\phi \in ((\mathcal{V} \to \mathcal{D}) \to \{tt, ff\})$ et de sortie $\psi \in ((\mathcal{V} \to \mathcal{D})^2 \to \{tt, ff\})$ d'un programme Pr, il s'agit de démontrer la propriété d'invariance :

$$\forall p \in \Sigma^{<} \langle S[\![Pr]\!], A[\![Pr]\!], t[\![Pr]\!], \varepsilon \rangle, i \in |p| \cdot [\sigma(p_i) \Rightarrow \Psi(p_0, p_i)]$$

où

$$\varepsilon(\langle L, M \rangle) = [(Pr \equiv L : \beta) \wedge \phi(M)]$$

$$\sigma(\langle L, M \rangle) = [Pr \equiv \beta L :]$$

$$\Psi(\langle \underline{L}, \underline{M} \rangle, \langle \bar{L}, \bar{M} \rangle) = \psi(\underline{M}, \bar{M})$$

Nous allons montrer dans un premier temps que l'essence de la méthode de Morris-Wegbreit [77] consiste à appliquer le principe d'induction $(\mathcal{Y}^{-1})$, c'est-à-dire à une correspondance (α, γ) entre invariants près à démontrer que

$$[\exists I \in A_S[\![Pr]\!] \cdot Cv[\![Pr]\!](\varepsilon, \sigma)(\Psi)(I)]$$

où

$$A_S[\![Pr]\!] = (S[\![Pr]\!]^2 \to \{tt, ff\})$$

$$Cv[\![Pr]\!](\varepsilon, \sigma)(\Psi)(I) = [Cv_\sigma[\![Pr]\!](\sigma)(I) \wedge Cv_i[\![Pr]\!](\sigma)(I) \wedge Cv_\varepsilon[\![Pr]\!](\varepsilon, \sigma)(\Psi)(I)]$$

avec

$$Cv_\sigma[\![Pr]\!](\sigma)(I) = [\forall \bar{A} \in S[\![Pr]\!] \cdot \sigma(\bar{A}) \Rightarrow I(\bar{A}, \bar{A})]$$

$$Cv_i[\![Pr]\!](\sigma)(I) = [\forall A, A', \bar{A} \in S[\![Pr]\!], a \in A[\![Pr]\!] \cdot (t_a[\![Pr]\!](A, A') \wedge I(A', \bar{A}) \wedge \sigma(\bar{A})) \Rightarrow I(A, \bar{A})]$$

$$Cv_\varepsilon[\![Pr]\!](\varepsilon, \sigma)(\Psi)(I) = [\forall \underline{A}, \bar{A} \in S[\![Pr]\!] \cdot (\varepsilon(\underline{A}) \wedge I(\underline{A}, \bar{A}) \wedge \sigma(\bar{A})) \Rightarrow \Psi(\underline{A}, \bar{A})]$$

Ensuite nous généraliserons la méthode aux programmes parallèles et à d'autres propriétés d'invariance (Cousot-R [81]).

4.3.2.2.1 · Programmes séquentiels

Nous considérons des programmes séquentiels comme ils ont été définis au paragraphe 2.8.1.

4.3.2.2.1.1 Choix d'un langage pour exprimer les invariants locaux et sa sémantique

Nous choisissons

$$\tilde{A_S}[\![Ps]\!] = \prod_{L \in C[\![Ps]\!]} (M[\![Ps]\!] \xrightarrow{\bullet} \{tt, ff\})$$

où

$$C[\![Ps]\!] = \{L \in \mathcal{L} : Ps \equiv \beta L : \delta\}$$

$$M[\![Ps]\!] = (\{x \in \mathcal{V} : Ps \equiv \alpha x \beta\} \to \mathcal{D})$$

pour pouvoir associer une relation sur les états mémoires à chaque point du programme. La signification de ce vecteur d'invariants locaux est définie par la fonction sémantique

$$\gamma[\![Ps]\!] \in (\tilde{A_S}[\![Ps]\!] \to A_S[\![Ps]\!])$$

$$\gamma[\![Ps]\!](\tilde{I})(\langle L, M\rangle, \langle \bar{L}, \bar{M}\rangle) = \tilde{I}(L)(M, \bar{M})$$

Intuitivement, quand le contrôle est en L, la relation $\tilde{I}(L)$ est vraie entre l'état mémoire courant M et l'état mémoire (final) $\bar{M}$ (qui correspond au point $\bar{L}$ de sortie de Ps).

4.3.2.2.1.2 Dérivation de conditions de vérification correctes

Nous avons à construire $C\tilde{v}[\![Ps]\!]$ tel que

$$C\tilde{v}[\![Ps]\!](\varepsilon, \sigma)(\Psi)(\tilde{I}) \Rightarrow Cv[\![Ps]\!](\varepsilon, \sigma)(\Psi)(\gamma[\![Ps]\!](\tilde{I}))$$

$Cv[\![Ps]\!]$ étant une conjonction de trois conditions, nous choisissons $C\tilde{v}[\![Ps]\!]$ également comme conjonction de trois conditions, chacune satisfaisant le critère de correction.

4.3.2.2.1.2.1 Finalisation

$$
\begin{aligned}
&Cv_{\sigma}[\![Ps]\!](\sigma)(\gamma[\![Ps]\!](\tilde{I})) \\
&\quad = [\forall \bar{\lambda} \in S[\![Ps]\!].\ \sigma(\bar{\lambda}) \Rightarrow \gamma[\![Ps]\!](\tilde{I})(\bar{\lambda},\bar{\lambda})] \\
&\quad = [\forall \bar{L} \in C[\![Ps]\!], \bar{M} \in M[\![Ps]\!].\ \sigma(\langle \bar{L},\bar{M}\rangle) \Rightarrow \gamma[\![Ps]\!](\tilde{I})(\langle \bar{L},\bar{M}\rangle,\langle \bar{L},\bar{M}\rangle)] \\
&\quad = [\forall \bar{L} \in C[\![Ps]\!], \bar{M} \in M[\![Ps]\!].\ (Ps \equiv \beta \bar{L}:) \Rightarrow \tilde{I}(\bar{L})(\bar{M},\bar{M})] \\
&\quad = [\forall \bar{M} \in M[\![Ps]\!].\ \tilde{I}(\bar{L})(\bar{M},\bar{M})] \quad \text{où} \quad Ps \equiv \alpha \bar{L}: \\
&\quad = \tilde{Cv}_{\sigma}[\![Ps]\!](\sigma)(\tilde{I})
\end{aligned}
$$

Intuitivement, cette condition de vérification établit que l'invariant $\tilde{I}(\bar{L})$ associé au point de sortie doit être vrai pour toute exécution qui se termine.

4.3.2.2.1.2.2 Induction

$$
\begin{aligned}
&Cv_{i}[\![Ps]\!](\sigma)(\gamma[\![Ps]\!](\tilde{I})) \\
&\quad = [\forall \lambda,\lambda',\bar{\lambda} \in S[\![Ps]\!], a \in A[\![Ps]\!].(t[\![Ps]\!]_{a}(\lambda,\lambda') \wedge \gamma[\![Ps]\!](\tilde{I})(\lambda',\bar{\lambda}) \wedge \sigma(\bar{\lambda})) \Rightarrow \gamma[\![Ps]\!](\tilde{I})(\lambda,\bar{\lambda})] \\
&\quad = [\forall L \in C[\![Ps]\!], M,\bar{M} \in M[\![Ps]\!]. \\
&\qquad [\exists L' \in C[\![Ps]\!], M' \in M[\![Ps]\!].\ \underset{\sim}{cond}[\![Ps]\!](L,L')(M) \wedge \underset{\sim}{succ}[\![Ps]\!](L)(M,M') \wedge \tilde{I}(L')(M',\bar{M})] \Rightarrow \tilde{I}(L)(M,\bar{M})] \\
&\quad = \tilde{Cv}_{i}[\![Ps]\!](\sigma)(\tilde{I})
\end{aligned}
$$

Intuitivement cette condition de vérification établit que si un pas atomique du programme peut conduire du point L où l'état mémoire est M à son successeur immédiat L' avec l'état mémoire M' tel que $\underset{\sim}{succ}[\![Ps]\!](L)(M,M')$ alors l'invariant local $\tilde{I}(L')(M',\bar{M})$ après ce pas doit impliquer l'invariant local $\tilde{I}(L)(M,\bar{M})$ avant ce pas.

Suivant la nature de la commande étiquetée par L et en utilisant les définitions de $cond[\![Ps]\!]$ et $\underset{\sim}{succ}[\![Ps]\!]$, nous pourrons la décomposer en sous-cas. Par exemple, si L désigne une boucle while, nous obtenons :

$$[\,(Ps \equiv \beta L:\ \underline{while}\ B\ \underline{do}\ L':\delta) \wedge M \in \underline{dom}(\mathbb{B}[\![B]\!]) \wedge \mathbb{B}[\![B]\!](M) = tt \wedge \tilde{I}(L')(M,M)\,]$$
$$\Rightarrow \tilde{I}(L)(M,\bar{M})$$

et

$$[\,(Ps \equiv \beta L:\ \underline{while}\ B\ \underline{do}\ Ps'\ \underline{od};\ L':\delta) \wedge M \in \underline{dom}(\mathbb{B}[\![B]\!]) \wedge \mathbb{B}[\![B]\!](M) = ff \wedge \tilde{I}(L')(M,\bar{M})\,]$$
$$\Rightarrow \tilde{I}(L)(M,\bar{M})$$

4.3.2.2.1.2.3 Initialisation

$$Cv_\varepsilon[\![Ps]\!](\varepsilon,\sigma)(\Psi)(\gamma[\![Ps]\!](\tilde{I}))$$
$$= [\forall \underline{\Delta}, \bar{\Delta} \in S[\![Ps]\!] .\ (\mathcal{E}(\underline{\Delta}) \wedge \gamma[\![Ps]\!](\tilde{I})(\underline{\Delta},\bar{\Delta}) \wedge \sigma(\bar{\Delta})) \Rightarrow \Psi(\underline{\Delta},\bar{\Delta})\,]$$
$$= [\forall \underline{L}, \bar{L} \in C[\![Ps]\!],\ \underline{M},\bar{M} \in M[\![Ps]\!].$$
$$(Ps \equiv \underline{L}:\beta \wedge \tilde{I}(\underline{L})(\underline{M},\bar{M}) \wedge Ps \equiv \beta\bar{L}:) \Rightarrow (\phi(\underline{M}) \Rightarrow \psi(\underline{M},\bar{M}))\,]$$
$$= [\,(\phi(\underline{M}) \wedge \tilde{I}(\underline{L})(\underline{M},\bar{M})) \Rightarrow \psi(\underline{M},\bar{M})\,] \quad \text{où } Ps \equiv \underline{L}:\alpha$$
$$= \tilde{Cv}_\varepsilon[\![Ps]\!](\varepsilon,\sigma)(\Psi)(\tilde{I})$$

Intuitivement, la spécification d'entrée et l'invariant local associé au point d'entrée doivent impliquer la spécification de sortie.

4.3.2.2.1.3 Vérification de la complétude sémantique

Définissons

$$\alpha[\![Ps]\!] \in (A_S[\![Ps]\!] \to \tilde{A}_S[\![Ps]\!])$$
$$\alpha[\![Ps]\!](I)(L)(M,\bar{M}) = [\forall \bar{L} \in C[\![Ps]\!] .\ (Ps \equiv \beta\bar{L}:) \Rightarrow I(\langle L,M\rangle, \langle \bar{L},\bar{M}\rangle)]$$

qui spécifie comment une hypothèse d'induction $I \in A_S[Ps]$ peut être codée par un vecteur d'invariants locaux $\tilde{I}(L)$ associés à chaque point L du programme Ps.

Ayant choisi $\tilde{Cv}[\![Ps]\!](\varepsilon,\sigma)(\Psi) = Cv[\![Ps]\!](\varepsilon,\sigma)(\Psi) \circ \gamma[\![Ps]\!]$, la vérification de la complétude sémantique est :

$\forall I \in A_0[\![Ps]\!].\ \tilde{C}o[\![Ps]\!](\varepsilon,\sigma)(\psi)(\alpha[\![Ps]\!](I)) = Co[\![Ps]\!](\varepsilon,\sigma)(\psi)(\gamma\circ\alpha[\![Ps]\!](I)) \Leftarrow Co[\![Ps]\!](\varepsilon,\sigma)(\Psi)(I)$

car $Co[\![Ps]\!](\varepsilon,\sigma)(\Psi)$ est monotone et $\gamma\circ\alpha$ est extensive.

Ce résultat réfute l'argument de Misra [78] que « subgoal induction is not guaranteed to prove a correct program correct ».

4.3.2.2.1.4 Résumé des conditions de vérification pour la preuve de correction partielle de programmes séquentiels par induction en arrière

Nous utiliserons des notations mnémoniques informelles. $\underline{x}$ et $\tilde{x}$ désignent les vecteurs des valeurs respectivement initiales et finales des variables du programme et P_i est l'invariant associé au point L_i.

- Finalisation

βL_1 : $\qquad \forall \bar{x}.\ P_1(\bar{x}, \bar{x})$

- Induction

. Commande nulle

βL_1 : skip ; L_2 : δ $\qquad P_2 \Rightarrow P_1$

. Commande d'affectation

βL_1 : $V := E$; L_2 : δ $\qquad P_2[V \leftarrow E] \Rightarrow P_1$

βL_1 : $V := ?$; L_2 : δ $\qquad \forall m \in \mathcal{D}.\ P_2[V \leftarrow m] \Rightarrow P_1$

. Commande conditionnelle

βL_1 : if B then

L_2 : δ

L_3 :

else $\qquad [(P_2 \wedge B) \vee (P_4 \wedge \neg B)] \Rightarrow P_1$

L_4 : δ' $\qquad P_6 \Rightarrow (P_3 \wedge P_5)$

L_5 :

fi ;

L_6 : β'

. Commande itérative

βL_1 : while B do

L_2 : δ $\qquad [(P_2 \wedge B) \vee (P_4 \wedge \neg B)] \Rightarrow (P_1 \wedge P_3)$

L_3 :

od ;

L_4 : β'

- Initialisation

L_1 : β $\qquad \forall \underline{x}, \bar{x}.\ [(\phi(\underline{x}) \wedge P_1(\underline{x}, \bar{x})) \Rightarrow \psi(\underline{x}, \bar{x})]$

4.3.2.2.1.5 Preuves d'autres propriétés d'invariance de programmes séquentiels par induction en arrière

Morris et Wegbreit affirment que "a drawback of subgoal induction is that it cannot be used to prove invariants about non-terminating programs". Le problème est de montrer que lorsque l'exécution d'un programme Ps commence dans un état mémoire initial $\underline{M}$ satisfaisant une spécification d'entrée ϕ et atteint un point L quelconque du programme avec l'état mémoire M, alors l'invariant $\psi(L)(M)$ doit être vrai. Formellement

$$\forall p \in \Sigma < S[\![Pr]\!], A[\![Pr]\!], t[\![Pr]\!], \varepsilon >, i \in |p| . \; [\, \sigma(p_i) \Rightarrow \Psi(p_0, p_i) \,]$$

où

$$\varepsilon(<L, M>) = [(Ps \equiv L : \beta) \wedge \phi(M)]$$

$$\sigma(<L, M>) = tt$$

$$\Psi(<\underline{L}, \underline{M}>, <\bar{L}, \bar{M}>) = \psi(\bar{L})(\bar{M})$$

Les définitions ci-dessus de σ et Ψ diffèrent du cas de la correction partielle et donc de "subgoal induction", de sorte que la remarque de Morris-Wegbreit est justifiée pour "subgoal induction" mais non pour toutes les méthodes de preuve par induction en arrière.

Pour être complet, construisons une méthode de preuve d'invariants par induction en arrière. Ψ est maintenant une assertion sur les états finaux (final signifiant quelconque dans ce cas). D'après 4.2.1.3, nous pourrions utiliser le principe d'induction (-$\bar{I}$-) contrapositif en arrière. La démarche reste la même et les conditions de vérification sont similaires à celles du paragraphe 4.3.2.2.1.2 pour le pas d'induction 4.3.2.2.1.2.2 tandis que 4.3.2.2.1.2.1 et 4.3.2.2.1.2.3 deviennent :

- Finalisation

$$\beta L_i : \delta \qquad\qquad \neg \psi_i \Rightarrow P_i$$

- Initialisation

$$L_1 : \beta \qquad\qquad \phi \Rightarrow \neg P_1$$

Exemple

Illustrons cette méthode par le (contre) exemple simple de Morris-Wegbreit [77], qui consiste à montrer que $x>0$ dans le programme suivant :

```
1:
     x := 1
2:
     while true do
3:
          x := x+1;
4:
     od;
5:
```

Nous choisissons $\phi(x) = tt$, $\psi_1(x) = tt$, $\psi_i(x) = [x>0]$, $i = 2,\dots,5$. Les conditions de vérification sont :

$\neg \psi_i \Rightarrow P_i \qquad i = 1,\dots,5$

$P_2[x \leftarrow 1] \Rightarrow P_1$

$[(P_3 \wedge \underline{true}) \vee (P_5 \wedge \neg \underline{true})] \Rightarrow (P_2 \wedge P_4)$

$P_4[x \leftarrow x+1] \Rightarrow P_3$

$\phi \Rightarrow \neg P_1$

Ces conditions de vérification sont trivialement satisfaites par $P_i = \neg \psi_i$, $i = 1,\dots,5$.

□

4.3.2.2.2 Programmes parallèles asynchrones

4.3.2.2.2.1 Choix d'un langage pour exprimer les invariants locaux et sa sémantique

Pour exprimer une relation entre états courants et finaux, nous associons une relation à chaque point du programme qui ne soit pas dans une section critique. Dans le prélude et le postlude, c'est une relation entre états mémoires courant et final. A chaque point d'un processus c'est une relation entre les valeurs courantes des états de contrôle des autres processus, l'état mémoire courant et l'état mémoire final. Ainsi pour un programme

$$Ppa \equiv Ps \; [\![\; Pra_0 \parallel \ldots \parallel Pra_i \parallel \ldots \parallel Pra_{m-1} \;]\!] \; ; Ps' \qquad (m>1)$$

nous choisissons

$$\begin{aligned}
\tilde{A}s[\![Ppa]\!] = \{ \tilde{I} : \; & [\exists \underline{L} \in C[\![Ps]\!] . \; \tilde{I}(\underline{L}) \in (M[\![Ppa]\!]^2 \to \{tt, ff\})] \\
& \vee [\exists i \in m, \; L \in C[\![Pra_i]\!] . \\
& \qquad \tilde{I}(i)(L) \in ((\underset{j \in (m \setminus i)}{\pi} C[\![Pra_j]\!] \times M[\![Ppa]\!]^2) \to \{tt, ff\})] \\
& \vee [\exists \bar{L} \in C[\![Ps']\!] . \; \tilde{I}(\bar{L}) \in (M[\![Ppa]\!]^2 \to \{tt, ff\})] \}
\end{aligned}$$

La signification d'un tel vecteur $\tilde{I}$ est définie formellement par la fonction sémantique $\gamma[\![Ppa]\!]$. Nous avons $\gamma[\![Ppa]\!](\tilde{I}) = I$ où, par cas :

- $I(\langle L, M\rangle, \langle \bar{L}, \bar{M}\rangle) = \tilde{I}(L)(M, \bar{M})$ quand $L \in (C[\![Ps]\!] \cup C[\![Ps']\!])$ et $Ppa \equiv p\bar{L}$:
- $I(\langle L_0, \ldots, L_{m-1}, M\rangle, \langle \bar{L}, \bar{M}\rangle) = \underset{i \in m}{\wedge} \tilde{I}(L_i)(L_0, \ldots, L_{i-1}, L_{i+1}, \ldots, L_{m-1}, M, \bar{M})$
  quand $L_i \in C[\![Pra_i]\!]$, $i \in m$ et $Ppa \equiv p\bar{L}$:

($\gamma[\![Ppa]\!](\tilde{I})(\Delta, \bar{\Delta})$ n'a pas besoin d'être définie quand $\bar{\Delta}$ n'est pas un état final).

4.3.2.2.2.2 Construction de conditions de vérification correctes

Les conditions de finalisation et d'initialisation sont similaires à 4.3.2.2.1.2.1 et 4.3.2.2.1.2.3 respectivement et ne présentent pas de difficultés. Le point principal consiste à trouver $\tilde{Cv}_i[\![Ppa]\!]$ tel que :

$$\tilde{Cv}_i[\![Ppa]\!](\sigma)(\tilde{I}) \Rightarrow Cv_i[\![Ppa]\!](\sigma)(\gamma[\![Ppa]\!](\tilde{I}))$$

où

$$Cv_i[\![Ppa]\!](\sigma)(\gamma[\![Ppa]\!](\tilde{I}))$$

$$= [\forall \Delta, \Delta' \in S[\![Ppa]\!], a \in A[\![Ppa]\!]. (t[\![Ppa]\!]_a(\Delta,\Delta') \wedge \gamma[\![Ppa]\!](\tilde{I})(\Delta') \wedge \sigma(\Delta)) \Rightarrow \gamma[\![Ppa]\!](\tilde{I})(\Delta)]$$

$$= [\forall \Delta, \Delta' \in S[\![Ppa]\!], \bar{M} \in M[\![Ppa]\!], a \in A[\![Ppa]\!].$$

$$(t[\![Ppa]\!]_a(\Delta,\Delta') \wedge \gamma[\![Ppa]\!](\tilde{I})(\Delta', \langle \bar{L}, \bar{M} \rangle) \wedge Ppa \equiv p\bar{L}:) \Rightarrow \gamma[\![Ppa]\!](\tilde{I})(\Delta, \langle \bar{L}, \bar{M} \rangle)]$$

Nous décomposons cette condition de vérification en sous-cas selon la forme de Δ définie par $S[\![Ppa]\!]$:

Cas 1 : $\Delta = \langle L, M \rangle$ où $L \in C[\![Ps]\!]$ et $M \in M[\![Ppa]\!]$

Cas 1.1 : $\neg (Ps \equiv pL:)$

Ce cas a été traité en 4.3.2.2.1 pour les programmes séquentiels.

Cas 1.2 : $(Ps \equiv pL:)$

D'après la définition 2.8.2.2.4 de $t[\![Ppa]\!]_a$, Δ' est nécessairement de la forme $\langle\langle L_0, \ldots, L_{m-1} \rangle, M \rangle$ où $Ppa \equiv pL:[\![L_0 : p_0 \| \ldots \| L_{m-1} : p_{m-1}]\!] p'$ et $a = p_i$ de sorte que par définition de $\gamma[\![Ppa]\!]$, la condition de vérification est dans ce cas

$$\bigwedge_{i \in m} \tilde{I}(L_i)(L_0, \ldots, L_{i-1}, L_{i+1}, \ldots, L_{m-1}, M, \bar{M}) \Rightarrow \tilde{I}(L)(M, \bar{M})$$

Intuitivement, la conjonction des invariants d'entrée de chaque processus doit impliquer l'invariant de sortie du prélude.

Cas 2 : $\Delta = \langle L, M \rangle$ où $L \in [\![Ps']\!]$ et $M \in M[\![Ppa]\!]$

Ce cas a été traité en 4.3.2.2.1 pour les programmes séquentiels.

Cas 3 : $s = \langle\langle L_0, \dots, L_{m-1}\rangle, M\rangle$

où $Ppa \equiv Ps \,[\![Pra_0 \parallel \dots \parallel Pra_{m-1}]\!]; Ps'$, $L_i \in C[\![Pra_i]\!]$ et $M \in M[\![Ppa]\!]$

Suivant la forme possible de s', nous distinguons deux cas :

Cas 3.1 : $s' = \langle L', M'\rangle$ où $L' \in (C[\![Ps]\!] \cup C[\![Ps']\!])$ et $M' \in (\mathcal{V} \to \mathcal{D})$

D'après la définition 2.8.2.2.4 de $t[\![Ppa]\!]_a$ nous avons nécéssairement $M = M'$ et $Ppa \equiv \beta \,[\![\beta_0 L_0 : \parallel \dots \parallel \beta_{m-1} L_{m-1} :]\!]; L' : \beta'$ et $a = \beta'$ de sorte que par définition de $\tilde{\gamma}[\![Ppa]\!](\tilde{I})$, la condition de vérification est dans ce cas

$$\tilde{I}(L')(M, \bar{M}) \Longrightarrow \bigwedge_{i \in m} \tilde{I}(L_i)(L_0, \dots, L_{i-1}, L_{i+1}, \dots, L_{m-1}, M, \bar{M})$$

Intuitivement, l'invariant d'entrée du postlude doit impliquer la la conjonction des invariants de sortie de chaque processus.

Cas 3.2 : $s' = \langle\langle L'_0, \dots, L'_{m-1}\rangle, M'\rangle$ où $L'_i \in C[\![Pra_i]\!]$, $i \in m$ et $M' \in M[\![Ppa]\!]$

Par définition de $t[\![Ppa]\!]_a$, nous décomposons ce cas en deux sous-cas suivant que la transition de s à s' correspond ou non à l'exécution d'une section critique.

Cas 3.2.1 : Transition ne correspondant pas à une section critique

Par définition de $t[\![Ppa]\!]_a$ et $\gamma[\![Ppa]\!]$, la condition de vérification équivaut à :

$$\begin{aligned}
&[\,[\exists i \in m.\ Ppa \equiv \beta \,[\![Pra_0 \parallel \dots \parallel Pra_i \parallel \dots \parallel Pra_{m-1}]\!]\, \beta' \ \wedge (\forall j \in (m \setminus i).\ L'_j = L_j) \\
&\qquad \wedge\ t[\![Pra_i]\!]_{\underline{a}}(\langle L_i, M\rangle, \langle L'_i, M'\rangle) \ \wedge \bigwedge_{j \in m} \tilde{I}(L'_j)(L'_0, \dots, L'_{j-1}, L'_{j+1}, \dots, L'_{m-1}, M', \bar{M})] \\
&\Longrightarrow \bigwedge_{k \in m} \tilde{I}(L_k)(L_0, \dots, L_{k-1}, L_{k+1}, \dots, L_{m-1}, M, \bar{M})\,]
\end{aligned}$$

Cette condition se décompose en une conjonction de conditions de vérification correspondant aux processus Pra_i, $i \in m$:

$$\begin{aligned}
&[\,[\underline{cond}\,[\![Pra_i]\!]\,(L_i, L'_i)(M) \ \wedge\ \underline{succ}\,[\![Pra_i]\!](L_i)(M, M') \\
&\quad \wedge\ \tilde{I}(L'_i)(L_1, \dots, L_{i-1}, L_{i+1}, \dots, L_{m-1}, M', \bar{M}) \\
&\quad \wedge \bigwedge_{j \in (m \setminus i)} \tilde{I}(L_j)(L_0, \dots, L_{i-1}, L'_i, L_{i+1}, \dots, L_{j-1}, L_{j+1}, \dots, L_{m-1}, M', \bar{M})] \\
&\quad \Longrightarrow \bigwedge_{k \in m} \tilde{I}(L_k)(L_0, \dots, L_{k-1}, L_{k+1}, \dots, L_{m-1}, M, \bar{M})\,]
\end{aligned}$$

De nouveau, cette condition se décompose en sous-cas suivant k :

Cas 3.2.1.1 : Preuve séquentielle $(k = i)$

$$[[\ \underline{cond}\ [\![Pra_i]\!](L_i, L'_i)(M) \wedge \underline{succ}\ [\![Pra_i]\!](L_i)(M, M')$$
$$\wedge\ \tilde{\tilde{I}}(L'_i)(L_0, \ldots, L_{i-1}, L_{i+1}, \ldots, L_{m-1}, M', \bar{M}) \quad \wedge\ \underline{context}\ (i,i)\ ]$$
$$\Rightarrow\ \tilde{\tilde{I}}(L_i)(L_0, \ldots, L_{i-1}, L_{i+1}, \ldots, L_{m-1}, M, \bar{M})\ ]$$

où

$$\underline{context}\ (i,k) = \bigwedge_{j \in (m - \{i,k\})} \tilde{\tilde{I}}(L_j)(L_0, \ldots, L_{i-1}, L'_i, L_{i+1}, \ldots, L_{j-1}, L_{j+1}, \ldots, L_{m-1}, M', \bar{M})$$

Cette condition de vérification correspond au cas des preuves séquentielles (cf. 4.3.2.2.1) excepté pour le terme $\underline{context}\ (i,i)$. Elle signifie que si l'invariant $\tilde{\tilde{I}}(L'_i)$ est vrai après la transition du point L_i au point L'_i, alors l'invariant $\tilde{\tilde{I}}(L_i)$ doit être vrai avant cette transition. De plus, le terme $\underline{context}\ (i,i)$ établit que nous pouvons utiliser dans la preuve toute l'information disponible sur les autres processus Pra_j, $j \neq i$ avant la transition. Pour avoir des preuves séquentielles similaires dans les cas de programmes séquentiels ou parallèles, nous négligeons le terme $\underline{context}(i,i)$ et choisissons la condition de vérification plus forte :

$$[\ \underline{cond}\ [\![Pra_i]\!](L_i, L'_i)(M) \wedge \underline{succ}\ [\![Pra_i]\!](L_i)(M, M') \wedge$$
$$\tilde{\tilde{I}}(L'_i)(L_0, \ldots, L_{i-1}, L_{i+1}, \ldots, L_{m-1}, M', \bar{M})\ ]$$
$$\Rightarrow \tilde{\tilde{I}}(L_i)(L_0, \ldots, L_{i-1}, L_{i+1}, \ldots, L_{m-1}, M, \bar{M})$$

Notons que cette simplification est correcte puisque la condition de vérification ci-dessus implique l'originale. C'est également sémantiquement complet car, intuitivement, l'information donnée par $\underline{context}\ (i,i)$ peut si nécessaire être incorporée en $\tilde{\tilde{I}}(L'_i)$ en chaque point L'_i de chaque processus Pra_i.

Cas 3.2.1.2 : Absence d'interférences $(k \neq i)$

Pour $k \in (m \vee i)$, nous devons montrer :

$$[[\ \underline{cond}\ [\![Pra_i]\!](L_i, L'_i)(M) \wedge \underline{succ}\ [\![Pra_i]\!](L_i)(M, M')$$
$$\wedge\ \tilde{I}(L'_i)(L_0, \ldots, L_{i-1}, L_{i+1}, \ldots, L_{m-1}, M', \bar{M})$$
$$\wedge\ \tilde{I}(L_k)(L_0, \ldots, L_{i-1}, L'_i, L_{i+1}, \ldots, L_{k-1}, L_{k+1}, \ldots, M', \bar{M}) \wedge \underline{context}(i,k)\ ]$$
$$\Rightarrow \tilde{I}(L_k)(L_0, \ldots, L_{k-1}, L_{k+1}, \ldots, L_{m-1}, M, \bar{M})\ ]$$

Intuitivement, les invariants $\tilde{I}(L_k)$ dans un processus Pra_k ne doivent pas être invalidés par l'exécution de commandes dans d'autres processus Pra_i, $i \neq k$. Comme ci-dessus, il est correct (et complet) de négliger le terme $\underline{context}(i,k)$.

Nous devrions maintenant détailler ces conditions de vérification suivant la nature de la commande désignée par L_i. Ceci est simple et nous donnerons seulement les résultats en 4.3.2.2.8.4.

Cas 3.2.2 Transition correspondant à une section critique

Lorsque $Ppa \equiv Ps\ [\![Pra_0 \| \ldots \| Pra_i \| \ldots \| Pra_{m-1}]\!]\ ; Ps'$, $Pra_i \equiv \beta L_i : \langle Ps'' \rangle ; L'_i : \beta'$ $Ps'' \equiv \underline{L}_1 : \delta \bar{L}_2 :$, nous devons, d'après la condition de vérification, montrer que :

$$\underline{cs\text{-}proof}\ (\ t[\![Ps'']\!]^*(\langle \underline{L}_1, M\rangle, \langle \bar{L}_2, M'\rangle))$$

où

$$\underline{cs\text{-}proof}\ (P) =$$
$$[[\ P \wedge \tilde{I}(L'_i)(L_0, \ldots, L_{i-1}, L_{i+1}, \ldots, L_{m-1}, M', \bar{M}) \wedge$$
$$\hat{\wedge}_{j \in (m \vee i)}\ \tilde{I}(L_j)(L_0, \ldots, L_{i-1}, L'_i, L_{i+1}, \ldots, L_{j-1}, L_{j+1}, \ldots, L_{m-1}, M', \bar{M})\ ]$$
$$\Rightarrow \hat{\wedge}_{k \in m}\ \tilde{I}(L_k)(L_0, \ldots, L_{k-1}, L_{k+1}, \ldots, L_{m-1}, M, \bar{M})\ ]$$

Comme c'est le cas dans toutes les preuves d'invariance, il n'est pas toujours nécessaire de caractériser exactement la relation $t[\![Ps'']\!]^*(\langle \underline{L}_1, M\rangle, \langle \bar{L}_2, M'\rangle)$ entre les états d'entrée et de sortie de la section critique. Une approximation ψ pourra être utilisée car la formule ci-dessus est égale à :

$$[\exists \psi \in (s[\![Ps''\!]\!]^2 \to \{tt, ff\}).$$

$$t[\![Ps'']\!]^*(\langle \underline{L}_1, M\rangle, \langle \bar{L}_2, M'\rangle) \Rightarrow \psi(\langle \underline{L}_1, M\rangle, \langle \bar{L}_2, M'\rangle)]$$

$$\wedge \ \underset{\sim}{cs\text{-}proof}(\psi(\langle \underline{L}_1, M\rangle, \langle \bar{L}_2, M'\rangle))]$$

Puisque nous avons à montrer que ψ est invariant, nous pourrons appliquer le principe d'induction ($-\forall^{-1}$). Nous obtenons :

$$[\exists \psi \in (s[\![Ps''\!]\!]^2 \to \{tt, ff\}).$$

$$[\exists J \in (s[\![Ps'']\!]^2 \to \{tt, ff\}).$$

$$(\forall M' \in (\mathcal{U} \to \mathcal{D}).\ J(\langle \bar{L}_2, M'\rangle, \langle \bar{L}_2, M'\rangle))$$

$$\wedge (\forall L_1, L_2 \in C[\![Ps'']\!],\ M_1, M_2, M' \in M[\![Ppa]\!].$$

$$(t[\![Ps'']\!](\langle L_1, M_1\rangle, \langle L_2, M_2\rangle) \wedge J(\langle L_2, M_2\rangle, \langle \bar{L}_2, M'\rangle)) \Rightarrow J(\langle L_1, M_1\rangle, \langle \bar{L}_2, M'\rangle))$$

$$\wedge (\forall M, M' \in (\mathcal{U} \to \mathcal{D}).\ J(\langle \underline{L}_1, M\rangle, \langle \bar{L}_2, M'\rangle) \Rightarrow \psi(\langle \underline{L}_1, M\rangle, \langle \bar{L}_2, M'\rangle))]$$

$$\wedge [\underset{\sim}{cs\text{-}proof}(\psi(\langle \underline{L}_1, M\rangle, \langle \bar{L}_2, M'\rangle))]]$$

Après simplification pour éliminer ψ, nous obtenons

$$[\exists J \in (s[\![Ps'']\!]^2 \to \{tt, ff\}).$$

$$\forall M \in (\mathcal{U} \to \mathcal{D}).\ J(\langle \bar{L}_2, M'\rangle, \langle \bar{L}_2, M'\rangle)$$

$$\wedge \forall L_1, L_2 \in C[\![Ps'']\!],\ M_1, M_2, M' \in M[\![Ppa]\!].$$

$$(t[\![Ps'']\!](\langle L_1, M_1\rangle, \langle L_2, M_2\rangle) \wedge J(\langle L_2, M_2\rangle, \langle \bar{L}_2, M'\rangle)) \Rightarrow J(\langle L_1, M_1\rangle, \langle \bar{L}_2, M'\rangle)$$

$$\wedge \underset{\sim}{cs\text{-}proof}(J(\langle \underline{L}_1, M\rangle, \langle \bar{L}_2, M'\rangle))]$$

Si nous appliquons 4.3.2.2.1.2.1 et 4.3.2.2.1.2.2, c'est équivalent à

$$[\exists \tilde{J} \in \prod_{L \in C[\![Ps'']\!]} (M[\![Ppa]\!]^2 \to \{tt, ff\}).$$

$$\forall M' \in (\mathcal{U} \to \mathcal{D}).\ \tilde{J}(\bar{L}_2)(M', M')$$

$$\wedge \forall L, L' \in C[\![Ps'']\!],\ M, M', M'' \in M[\![Ppa]\!].$$

$$(\underset{\sim}{cond}[\![Ps'']\!](L, L')(M) \wedge \underset{\sim}{succ}[\![Ps'']\!](L)(M, M'') \wedge \tilde{J}(L')(M'', M')) \Rightarrow \tilde{J}(L)(M, M')$$

$$\wedge \underset{\sim}{cs\text{-}proof}(\tilde{J}(\underline{L}_1)(M, M'))]$$

Ainsi la condition de vérification pour les sections critiques a été divisée en deux sous-problèmes. Premièrement, le corps Ps'' de la section critique doit être traité indépendemment de son contexte de sorte à inventer en chaque point L une relation $\tilde{J}(L)(M, M')$ entre

l'état mémoire courant M et l'état mémoire final M', et à montrer que $\tilde{J}$ est invariant en utilisant la méthode de preuve par induction en arrière récapitulée en 4.3.2.2.1.4. Puis faire la preuve cs-proof ($\tilde{J}(\underline{L}_{-1})(M,M')$) où la section critique est considérée comme atomique et sa sémantique définie par $\tilde{J}(\underline{L}_{-1})(M,M')$. Comme pour les autres commandes, cette preuve peut se décomposer en :

- une preuve séquentielle (en omettant context(i,i))

$$[\tilde{J}(\underline{L}_{-1})(M,M') \wedge \tilde{I}(L'_i)(L_0,\dots,L_{i-1},L_{i+1},\dots,L_{m-1},M',\bar{M})]$$
$$\Rightarrow \tilde{I}(L_i)(L_0,\dots,L_{i-1},L_{i+1},\dots,L_{m-1},M,\bar{M})$$

- une preuve d'absence d'interférences (en omettant context(i,k))

$$[\tilde{J}(\underline{L}_{-1})(M,M') \wedge \tilde{I}(L'_i)(L_0,\dots,L_{i-1},L_{i+1},\dots,L_{m-1},M',\bar{M}) \wedge$$
$$\tilde{I}(L_k)(L_0,\dots,L_{i-1},L'_i,L_{i+1},\dots,L_{k-1},L_{k+1},\dots,L_{m-1},M',\bar{M})]$$
$$\Rightarrow \tilde{I}(L_k)(L_0,\dots,L_{k-1},L_{k+1},\dots,L_{m-1},M,\bar{M})$$

4.3.2.2.2.3 Vérification de la complétude sémantique

Définissons $\alpha[\![Ppa]\!] \in (As[\![Ppa]\!] \to \tilde{A}s[\![Ppa]\!])$ qui spécifie comment une hypothèse d'induction $I \in As[\![Ppa]\!]$ peut être codée par un vecteur d'invariants $\tilde{I}(L)$ associés en chaque point L du programme Ppa

$$Ppa \equiv Ps\,[\![\,Pra_0 \| \dots \| Pra_i \| \dots \| Pra_{m-1}\,]\!]\,; Ps'$$

par cas :

- $\alpha[\![Ppa]\!](I)(L)(M,\bar{M}) = [\forall \bar{L} \in C[\![Ppa]\!].\ (Ppa \equiv \beta\bar{L}:) \Rightarrow I(\langle L,M\rangle,\langle\bar{L},\bar{M}\rangle)]$
  quand $L \in (C[\![Ps]\!] \cup C[\![Ps']\!])$
- $\forall i\,em,\ L \in C[\![Pra_i]\!]$,

$$\alpha[\![Ppa]\!](I)(i)(L)(L_0,\dots,L_{i-1},L_{i+1},\dots,L_{m-1},M,\bar{M}) =$$
$$[\forall \bar{L} \in C[\![Ppa]\!].\ (Ppa \equiv \beta\bar{L}:) \Rightarrow I(\langle\langle L_0,\dots,L_{i-1},L,L_{i+1},\dots,L_{m-1}\rangle,M\rangle,\langle\bar{L},\bar{M}\rangle)]$$

La preuve de complétude (sémantique) montre qu'il est complet d'omettre les termes context(i,i) dans la preuve séquentielle et context(i,k) dans la preuve d'absence d'interférences. Elle montre aussi que la preuve d'absence d'interférences peut être simplifiée en omettant le terme $\tilde{I}(L_R)$ à gauche de l'implication. (Cependant, en pratique ce terme est utile puisque $\tilde{I}(L'_i)$ pourra s'écrire plus simplement).

Nous devons montrer que :

$$\forall I \in As[\![Ppa]\!].\ Cv[\![Ppa]\!](\varepsilon,\sigma)(\psi)(I) \Rightarrow C\tilde{v}[\![Ppa]\!](\varepsilon,\sigma)(\psi)(\alpha[\![Ppa]\!](I))$$

La preuve suit les cas considérés en 4.3.2.2.2. Nous traitons uniquement le cas 3.2.1 (puisque le cas 3.2.2 est similaire tandis que les cas restants se traitent comme en 4.3.2.2.1.2).

Le terme correspondant à $C\tilde{v}[\![Ppa]\!](\varepsilon,\sigma)(\psi)(\alpha[\![Ppa]\!](I))$ dans le cas 3.2.1 est la conjonction d'une preuve séquentielle et d'une preuve d'absence d'interférences :

$$\begin{aligned}
&[\ [(\underline{cond}[\![Pra_i]\!](L_i,L'_i)(M) \wedge \underline{succ}[\![Pra_i]\!](L_i)(M,M') \wedge (\forall \bar{L} \in C[\![Ppa]\!].\\
&\qquad (Ppa \equiv p\bar{L}:) \Rightarrow I(\langle\langle L_0,\dots,L_{i-1},L'_i,L_{i+1},\dots,L_{m-1}\rangle,M'\rangle,\langle\bar{L},\bar{M}\rangle)))\\
&\quad \Rightarrow (\forall \bar{L} \in C[\![Ppa]\!].\ (Ppa \equiv p\bar{L}:) \Rightarrow I(\langle\langle L_0,\dots,L_{m-1}\rangle,M\rangle,\langle\bar{L},\bar{M}\rangle)\ ]\\
\wedge\\
&\bigwedge_{k\in(m\setminus i)} [\ (\underline{cond}[\![Pra_i]\!](L_i,L'_i)(M) \wedge \underline{succ}[\![Pra_i]\!](L_i)(M,M')\\
&\qquad \wedge (\forall \bar{L} \in C[\![Ppa]\!].\ (Ppa \equiv p\bar{L}:) \Rightarrow I(\langle\langle L_0,\dots,L_{i-1},L'_i,L_{i+1},\dots,L_{m-1},M'\rangle,\langle\bar{L},\bar{M}\rangle)))\\
&\qquad \Rightarrow (\forall \bar{L} \in C[\![Ppa]\!].\ (Ppa \equiv p\bar{L}:) \Rightarrow I(\langle\langle L_0,\dots,L_{m-1}\rangle,M\rangle,\langle\bar{L},\bar{M}\rangle))\ ]\ ]
\end{aligned}$$

qui est impliqué par :

$$\begin{aligned}
&[[\ \underline{cond}[\![Pra_i]\!](L_i,L'_i)(M) \wedge \underline{succ}[\![Pra_i]\!](L_i)(M,M')\\
&\quad \wedge (\forall \bar{L} \in C[\![Ppa]\!].\ (Ppa \equiv p\bar{L}:) \Rightarrow I(\langle\langle L_0,\dots,L_{i-1},L'_i,L_{i+1},\dots,L_{m-1}\rangle,M'\rangle,\langle\bar{L},\bar{M}\rangle))\ ]\\
&\quad \Rightarrow (\forall \bar{L} \in C[\![Ppa]\!].\ (Ppa \equiv p\bar{L}:) \Rightarrow I(\langle\langle L_0,\dots,L_{m-1}\rangle,M\rangle,\langle\bar{L},\bar{M}\rangle))\ ]
\end{aligned}$$

qui est lui-même impliqué par

$$\begin{aligned}
&[\forall \bar{L} \in C[\![Ppa]\!].\ (Ppa \equiv p\bar{L}:) \Rightarrow (\underline{cond}[\![Pra_i]\!](L_i,L'_i)(M) \wedge \underline{succ}[\![Pra_i]\!](L_i)(M,M') \wedge\\
&\quad I(\langle\langle L_0,\dots,L_{i-1},L'_i,L_{i+1},\dots,L_{m-1}\rangle,M'\rangle,\langle\bar{L},\bar{M}\rangle))] \Rightarrow I(\langle\langle L_0,\dots,L_{m-1}\rangle,M\rangle,\langle\bar{L},\bar{M}\rangle)\ ]
\end{aligned}$$

qui est le terme de $Cv[\![Ppa]\!](\varepsilon,\sigma)(\psi)(I)$ correspondant à la transition considérée au cas 3.2.1.

4.3.2.2.2.4 Résumé des conditions de vérification pour la preuve de correction partielle de programmes parallèles asynchrones par induction en arrière

$Ppa \equiv \alpha_0 \ell_0 : \beta_0 \; [\![Pra_0 \| \ldots \| Pra_{i-1} \| \alpha \ell_{ij} : \beta \| Pra_{i+1} \| \ldots \| Pra_{m-1}]\!] ; \alpha_1 \ell_1 : \beta_1 ; \bar{\ell} :$

Les invariants $P_0(x, \bar{x})$ associé au point ℓ_0 du prélude et $P_1(x, \bar{x})$ associé au point ℓ_1 du postlude relient la valeur courante x à la valeur finale $\bar{x}$ des variables (quand l'exécution est en $\bar{\ell}$). L'invariant de sortie $P_{\bar{\ell}}(\bar{x}, \bar{x})$ porte sur la valeur finale $\bar{x}$ des variables.
L'invariant $P_{ij}(c_0, \ldots, c_{i-1}, c_{i+1}, \ldots, c_{m-1}, x, \bar{x})$ associé au point j du processus i relie les états de contrôle $c_0, \ldots, c_{i-1}, c_{i+1}, \ldots, c_{m-1}$ des autres processus, les valeurs courantes x et les valeurs finales $\bar{x}$ des variables.

- Preuve séquentielle

Les conditions de vérification pour le prélude, le postlude et chaque processus Pra_i, $i \in m$ sont les mêmes que pour les programmes séquentiels (cf. 4.3.2.2.1.4) plus

. Finalisation du parallélisme

$\alpha \; [\![\beta_0 \ell_0 : \| \ldots \| \beta_{m-1} \ell_{m-1} :]\!] ; \ell_f : \beta$

$P_f(x, \bar{x}) \Rightarrow \bigwedge_{i \in m} P_i(\ell_0, \ldots, \ell_{i-1}, \ell_{i+1}, \ldots, \ell_{m-1}, x, \bar{x})$

. Section critique

$\alpha \ell_{i_1} : \not\downarrow$
$\ell_{i_2} : \beta$
$\ell_{i_3} : \not\uparrow ;$
$\ell_{i_4} : \gamma$

A chaque point ℓ_{ij} du corps $\ell_{i2} : \beta\, \ell_{i3}$, un invariant $P_{ij}(x, x')$ relie les valeurs courantes x en ℓ_{ij} aux valeurs finales x' en ℓ_{i3} des variables. Les conditions de vérification sont celles des programmes séquentiels (excepté pour l'initialisation) :

$$[\, P_{i2}(x, x') \wedge P_{i4}(c_0, \dots, c_{i-1}, c_{i+1}, \dots, c_{m-1}, x', \bar{x})]$$
$$\Rightarrow P_{i1}(c_0, \dots, c_{i-1}, c_{i+1}, \dots, c_{m-1}, x, \bar{x})$$

- Initialisation du parallélisme

$$\alpha\, \ell_d : [\![\, \ell_0 : \alpha_0 \,\|\, \dots \,\|\, \ell_{m-1} : \alpha_{m-1} \,]\!]\, \beta$$
$$[\, \bigwedge_{i \in m} P_i(\ell_0, \dots, \ell_{i-1}, \ell_{i+1}, \dots, \ell_{m-1}, x, \bar{x}) \,] \Rightarrow P_d(x, \bar{x})$$

- Preuve d'absence d'interférences

Pour tout point ℓ_{kj} de tout processus Pra_k, $k \in (m \setminus i)$

- Commande nulle

$$\alpha\, \ell_{i1} : \underline{\text{skip}}\ ;\ \ell_{i2} : \beta$$
$$(P_{i2}[c_k \leftarrow \ell_{kj}] \wedge P_{kj}[c_i \leftarrow \ell_{i2}]) \Rightarrow P_{kj}[c_i \leftarrow \ell_{i1}]$$

- Commande d'affectation

$$\alpha\, \ell_{i1} : V := E\ ;\ \ell_{i2} : \beta$$
$$(P_{i2}[c_k \leftarrow \ell_{kj}, V \leftarrow E] \wedge P_{kj}[c_i \leftarrow \ell_{i2}, V \leftarrow E]) \Rightarrow P_{kj}[c_i \leftarrow \ell_{i1}]$$

$$\alpha\, \ell_{i1} : V := ?\ ;\ \ell_{i2} : \beta$$
$$\forall m \in \mathcal{D}.\ (P_{i2}[c_k \leftarrow \ell_{kj}, V \leftarrow m] \wedge P_{kj}[c_i \leftarrow \ell_{i2}, V \leftarrow m]) \Rightarrow P_{kj}[c_i \leftarrow \ell_{i1}]$$

Commande conditionnelle

$\alpha\ \ell_{i1}$: if B then

ℓ_{i2} :
ℓ_{i3} : β

else

ℓ_{i4} :
ℓ_{i5} : γ

fi ;

ℓ_{i6} : δ

$$[(P_{i2}[c_k \leftarrow \ell_{kj}] \wedge B \wedge P_{kj}[c_i \leftarrow \ell_{i2}]) \vee (P_{i4}[c_k \leftarrow \ell_{kj}] \wedge \neg B \wedge P_{kj}[c_i \leftarrow \ell_{i4}])] \Rightarrow P_{kj}[c_i \leftarrow \ell_{i1}]$$

$$(P_{i6}[c_k \leftarrow \ell_{kj}] \wedge P_{kj}[c_i \leftarrow \ell_{i6}]) \Rightarrow (P_{kj}[c_i \leftarrow \ell_{i3}] \wedge P_{kj}[c_i \leftarrow \ell_{i5}])$$

Commande itérative

$\alpha\ \ell_{i1}$: while B do

ℓ_{i2} :
ℓ_{i3} : β

od ;

ℓ_{i4} : γ

$$[(P_{i2}[c_k \leftarrow \ell_{kj}] \wedge B \wedge P_{kj}[c_i \leftarrow \ell_{i2}]) \vee (P_{i4}[c_k \leftarrow \ell_{kj}] \wedge \neg B \wedge P_{kj}[c_i \leftarrow \ell_{i4}])] \Rightarrow [P_{kj}[c_i \leftarrow \ell_{i1}] \wedge P_{kj}[c_i \leftarrow \ell_{i3}]]$$

- Section critique

i_4

$$[P_{i2}(x, x') \wedge P_{i4}(c_0, \dots, c_{k-1}, \ell_{kj}, c_{k+1}, \dots, c_{i-1}, c_{i+1}, \dots, c_{m-1}, x', \bar{x})$$
$$\wedge P_{kj}(c_0, \dots, c_{k-1}, c_{k+1}, \dots, c_{i-1}, \ell_{i4}, c_{i+1}, \dots, c_{m-1}, x', \bar{x})$$
$$\Rightarrow P_{kj}(c_0, \dots, c_{k-1}, c_{k+1}, \dots, c_{i-1}, \ell_{i1}, c_{i+1}, \dots, c_{m-1}, x, \bar{x})$$

4.3.2.2.2.5 Exemples

Exemple 4.3.2.2.2.5-1

Un exemple très simple pris dans Owicki-Gries [76]. Nous devons montrer que le programme suivant :

```
0:
   [[
      11: <
            12:
               x := x+1 ;
            13:
         >;
      14:
   ||
      21: <
            22:
               x:=x+1;
            23:
         >;
      24:
   ]];
3:
```

est partiellement correct pour :

$\phi(\underline{x}) = (\underline{x} = 0)$

$\psi(\underline{x}, \bar{x}) = (\bar{x} = 2)$

Les conditions de vérification sont les suivantes :

- Finalisation :

  $P_3(\bar{x}, \bar{x})$

  $P_3(x, \bar{x}) \Rightarrow [P_{24}(14, x, \bar{x}) \wedge P_{14}(24, x, \bar{x})]$

- Preuve séquentielle pour le processus 2 :

  $P_{23}(x', x')$

  $P_{23}(x+1, x') \Rightarrow P_{22}(x, x')$

  $[P_{22}(x, x') \wedge P_{24}(c_1, x', \bar{x})] \Rightarrow P_{21}(c_1, x, \bar{x})$

- Absence d'interférences du processus 2 avec la preuve du processus 1 :

  $[P_{22}(x, x') \wedge P_{24}(11, x', \bar{x}) \wedge P_{11}(24, x', \bar{x})] \Rightarrow P_{11}(21, x, \bar{x})$

  $[P_{22}(x, x') \wedge P_{24}(14, x', \bar{x}) \wedge P_{14}(24, x', \bar{x})] \Rightarrow P_{14}(21, x, \bar{x})$

- De la même manière, nous avons une preuve séquentielle pour le processus 1 et une preuve d'absence d'interférences du processus 1 avec la preuve du processus 2.

- Initialisation :

$$[P_{11}(21, x, \bar{x}) \wedge P_{21}(11, x, \bar{x})] \Rightarrow P_0(x, \bar{x})$$

$$[\phi(x) \wedge P_0(x, \bar{x})] \Rightarrow \psi(x, \bar{x})$$

Esquisse de la preuve :

0: $\{x=\bar{x}-2\}$

[[

11: $\{(c_2=21 \wedge x=\bar{x}-2) \vee (c_2=24 \wedge x=\bar{x}-1)\}$

<

12: $\{x=x'-1\}$
x:=x+1;
13: $\{x=x'\}$

>;

14: $\{(c_2=21 \wedge x=\bar{x}-1) \vee (c_2=24 \wedge x=\bar{x})\}$

||

21: $\{(c_1=11 \wedge x=\bar{x}-2) \vee (c_1=14 \wedge x=\bar{x}-1)\}$

<

22: $\{x=x'-1\}$
x:=x+1;
23: $\{x=x'\}$

>;

24: $\{(c_1=11 \wedge x=\bar{x}-1) \vee (c_1=14 \wedge x=\bar{x})\}$

]];

3: $\{x=\bar{x}\}$

□

Exemple 4.3.2.2.2.5-2

Le programme parallèle asynchrone suivant calcule $f=m!$ quand $m>1$.

```
0:
1: n1:=1; n2:=n;
   [[        11:
             12: f1:=1;
             13: while (n1+2)<n2 do
             14: n1:=n1+1;
             15: f1:=f1×n1;
             16: od;

             21:
             22: f2:=n2;
             23: while (n1+2)<n2
             24: n2:=n2-1;
             25: f2:=f2×n2;
             26: od;
   ]];
3: if (n1+1)=n2 then f:=f1×f2; else f:=f1×f2×(n1+1); fi;
```

Les conditions de vérification sont les suivantes :

Finalisation :

$$\forall \bar{m}, \bar{m}_1, \bar{m}_2, \bar{f}_1, \bar{f}_2, \bar{f} \cdot P_4(\langle \bar{m}, \bar{m}_1, \bar{m}_2, \bar{f}_1, \bar{f}_2, \bar{f} \rangle, \langle \bar{m}, \bar{m}_1, \bar{m}_2, \bar{f}_1, \bar{f}_2, \bar{f} \rangle)$$

$$[(P_4[f \leftarrow f_1 \times f_2] \wedge (m_1+1)=m_2) \vee (P_4[f \leftarrow f_1 \times f_2 \times (m_1+1)] \wedge (m_1+1) \neq m_2)] \Rightarrow P_3$$

$$P_3 \Rightarrow (P_{26}[c_1 \leftarrow 16] \wedge P_{16}[c_2 \leftarrow 26])$$

. Preuve séquentielle du processus 2 :

$$[(P_{23} \wedge (m_1+2)<m_2) \vee (P_{26} \wedge (m_1+2) \geq m_2)] \Rightarrow [P_{22} \wedge P_{25}]$$

$$P_{25}[f_2 \leftarrow f_2 \times m_2] \Rightarrow P_{24}$$

$$P_{24}[m_2 \leftarrow m_2-1] \Rightarrow P_{23}$$

$$P_{22}[f_2 \leftarrow m_2] \Rightarrow P_{21}$$

. Preuve d'absence d'interférences du processus 2 avec la preuve du processus 1 :

Pour $j=1,\ldots,6$

$$[(P_{23}[c_1 \leftarrow 1j] \wedge (m_1+2)<m_2 \wedge P_{1j}[c_2 \leftarrow 23]) \vee (P_{26}[c_1 \leftarrow 1j] \wedge (m_1+2) \geq m_2 \wedge P_{1j}[c_2 \leftarrow 26])]$$

$$\Rightarrow (P_{1j}[c_2 \leftarrow 22] \wedge P_{1j}[c_2 \leftarrow 25])$$

$$(P_{25}[c_1 \leftarrow 1j, f_2 \leftarrow f_2 \times m_2] \wedge P_{1j}[c_2 \leftarrow 25, f_2 \leftarrow f_2 \times m_2]) \Rightarrow P_{1j}[c_2 \leftarrow 24]$$

$$(P_{24}[c_1 \leftarrow 1j, m_2 \leftarrow m_2-1] \wedge P_{1j}[c_2 \leftarrow 24, m_2 \leftarrow m_2-1]) \Rightarrow P_{1j}[c_2 \leftarrow 23]$$

$$(P_{22}[c_1 \leftarrow 1j, f_2 \leftarrow m_2] \wedge P_{1j}[c_2 \leftarrow 22, f_2 \leftarrow m_2]) \Rightarrow P_{1j}[c_2 \leftarrow 21]$$

. La preuve séquentielle du processus 1 et la preuve d'absence d'interférences du processus 1 avec la preuve du processus 2 sont similaires.

. Initialisation :

$(P_{11}[c_2 \leftarrow 21] \wedge P_{21}[c_1 \leftarrow 11]) \Rightarrow P_1$

$P_1[n_1 \leftarrow 1, n_2 \leftarrow n] \Rightarrow P_0$

$(n>1 \wedge P_0) \Rightarrow (\bar{f} = \bar{n}! \wedge \bar{n} = n)$

L'esquisse de la preuve est :

Nous posons $\pi(a,b) = (a \leq b \rightarrow a \times (a-1) \times \cdots \times b \mid 1)$ et

$I = (n=\bar{n} \wedge [(\bar{n}_1+1 = n_2 \wedge \bar{f} = \bar{f}_1 \times \bar{f}_2) \vee (\bar{n}_1+1 \neq \bar{n}_2 \wedge \bar{f} = \bar{f}_1 \times \bar{f}_2 \times (\bar{n}_1+1))])$.

0: $\{(n>1) \Rightarrow (\bar{f}=n! \wedge \bar{n}=n)\}$
$n_1:=1;\ n_2:=n;$
1: $\{(n_1<n_2) \Rightarrow (1 \leq \bar{n}_2 - \bar{n}_1 \leq 2 \wedge \bar{f}_1 = \pi(n_1+1, \bar{n}_1) \wedge \bar{f}_2 = \pi(\bar{n}_2, n_2) \wedge I)\}$
[[
11: $\{[(c_2 \in \{21,22\} \wedge n_1<n_2) \vee (c_2=23 \wedge n_1+2<n_2) \vee (c_2 \in \{24,25\} \wedge n_1+1<n_2) \vee (c_2=26 \wedge n_1<n_2 \leq n_1+2)] \Rightarrow [\bar{f}_1 = \pi(n_1+1, \bar{n}_1) \wedge \sup(n_1, \bar{n}_2-2) \leq \bar{n}_1 < \bar{n}_2 \wedge I]\}$
$f_1:=1;$
12: $\{[(c_2 \in \{21,22\} \wedge n_1<n_2) \vee (c_2=23 \wedge n_1+2<n_2) \vee (c_2 \in \{24,25\} \wedge n_1+1<n_2) \vee (c_2=26 \wedge n_1<n_2 \leq n_1+2)] \Rightarrow [\bar{f}_1 = f_1 \times \pi(n_1+1, \bar{n}_1) \wedge \sup(n_1, \bar{n}_2-2) \leq \bar{n}_1 < \bar{n}_2 \wedge I]\}$
while $(n_1+2)<n_2$ **do**
13: $\{[(c_2 \in \{21,22,23\} \wedge n_1+2<n_2) \vee (c_2 \in \{24,25,26\} \wedge n_1+1<n_2)] \Rightarrow [f_1 = f_1 \times \pi(n_1+1, \bar{n}_1) \wedge \sup(n_1+1, \bar{n}_2-2) \leq \bar{n}_1 < \bar{n}_2 \wedge I]\}$
$n_1:=n_1+1;$
14: $\{[(c_2 \in \{21,22,23\} \wedge n_1+1<n_2) \vee (c_2 \in \{24,25,26\} \wedge n_1<n_2)] \Rightarrow [f_1 = f_1 \times \pi(n_1, \bar{n}_1) \wedge \sup(n_1, \bar{n}_2-2) \leq \bar{n}_1 < \bar{n}_2 \wedge I]\}$
$f_1:=f_1*n_1;$
15: $\{[(c_2 \in \{21,22,23\} \wedge n_1+1<n_2) \vee (c_2 \in \{24,25,26\} \wedge n_1<n_2)] \Rightarrow [f_1 = f_1 \times \pi(n_1+1, \bar{n}_1) \wedge \sup(n_1, \bar{n}_2-2) \leq \bar{n}_1 < \bar{n}_2 \wedge I]\}$
od;
16: $\{[1 \leq n_2 - n_1 \leq 2] \Rightarrow [\bar{n}_2 - 2 \leq n_1 = \bar{n}_1 < \bar{n}_2 \wedge f_1 = \bar{f}_1 \wedge I]\}$
||
21: $\{[(c_1 \in \{11,12\} \wedge n_1<n_2) \vee (c_1=13 \wedge n_1+2<n_2) \vee (c_1 \in \{14,15\} \wedge n_1+1<n_2) \vee (c_1=16 \wedge n_1<n_2 \leq n_1+2)] \Rightarrow [\bar{f}_2 = \pi(\bar{n}_2, n_2) \wedge \bar{n}_1 < \bar{n}_2 \leq \inf(\bar{n}_1+2, n_2) \wedge I]\}$
$f_2:=n_2;$
22: $\{[(c_1 \in \{11,12\} \wedge n_1<n_2) \vee (c_1=13 \wedge n_1+2<n_2) \vee (c_1 \in \{14,15\} \wedge n_1+1<n_2) \vee (c_1=16 \wedge n_1<n_2 \leq n_1+2)] \Rightarrow [\bar{f}_2 = \pi(\bar{n}_2, n_2-1) \times f_2 \wedge \bar{n}_1 < \bar{n}_2 \leq \inf(\bar{n}_1+2, n_2) \wedge I]\}$
while $(n_1+2)<n_2$ **do**
23: $\{[(c_1 \in \{11,12,13\} \wedge n_1+2<n_2) \vee (c_1 \in \{14,15,16\} \wedge n_1+1<n_2)] \Rightarrow [\bar{f}_2 = \pi(\bar{n}_2, n_2-1) \times f_2 \wedge \bar{n}_1 < \bar{n}_2 \leq \inf(\bar{n}_1+2, n_2-1) \wedge I]\}$
$n_2:=n_2-1;$
24: $\{[(c_1 \in \{11,12,13\} \wedge n_1+1<n_2) \vee (c_1 \in \{14,15,16\} \wedge n_1<n_2)] \Rightarrow [\bar{f}_2 = \pi(\bar{n}_2, n_2) \times f_2 \wedge \bar{n}_1 < \bar{n}_2 \leq \inf(\bar{n}_1+2, n_2) \wedge I]\}$
$f_2:=f_2*n_2;$
25: $\{[(c_1 \in \{11,12,13\} \wedge n_1+1<n_2) \vee (c_1 \in \{14,15,16\} \wedge n_1<n_2)] \Rightarrow [\bar{f}_2 = \pi(\bar{n}_2, n_2-1) \times f_2 \wedge \bar{n}_1 < \bar{n}_2 \leq \inf(\bar{n}_1+2, n_2) \wedge I]\}$
od;
26: $\{[1 \leq n_2 - n_1 \leq 2] \Rightarrow [\bar{f}_2 = f_2 \wedge \bar{n}_1 < \bar{n}_2 = n_2 \leq \bar{n}_1 + 2 \wedge I]\}$
]];
3: $\{n_1=\bar{n}_1 \wedge n_2=\bar{n}_2 \wedge f_1=\bar{f}_1 \wedge f_2=\bar{f}_2 \wedge I\}$
if $(n_1+1)=n_2$ **then** $f:=f_1 \times f_2;$ **else** $f:=f_1 \times f_2 \times (n_1+1);$ **fi;**
4: $\{n=\bar{n} \wedge n_1=\bar{n}_1 \wedge n_2=\bar{n}_2 \wedge f_1=\bar{f}_1 \wedge f_2=\bar{f}_2 \wedge f=\bar{f}\}$

□

Remarque 4.3.2.2.2.5-3 (Un principe d'induction avant-arrière symétrique)

Les invariants associés en chaque point de chaque processus sont de la forme $A \Rightarrow R$ où A dépend des valeurs courantes des états de contrôle des autres processus et des valeurs courantes des variables et R est une relation entre les valeurs courantes et finales des variables. A décrit ce qui a été accompli jusqu'ici et R décrit ce qui reste à faire. Ceci aurait été encore plus clair si nous avions utilisé des invariants redondants qui décrivent plus précisément le comportement du programme, par exemple

4: $\{[m>1 \wedge f=m!] \Rightarrow [m=\bar{m} \wedge m_1=\bar{m}_1 \wedge m_2=\bar{m}_2 \wedge f_1=\bar{f}_1 \wedge f_2=\bar{f}_2 \wedge f=\bar{f}]\}$

23: $\{[[(c_1=11 \wedge m_1=1 \wedge m_1+2<m_2) \vee (c_1 \in \{12,13\} \wedge m_1>1 \wedge f_1=m_1! \wedge m_1+2<m_2) \vee$
$(c_1=14 \wedge m_1>1 \wedge f_1=(m_1-1)! \wedge m_1+1<m_2) \vee (c_1 \in \{15,16\} \wedge m_1>1 \wedge f_1=m_1! \wedge m_1+1<m_2)]$
$\wedge [m_2 \leq m \wedge f_2=\pi(m_2,m)]] \Rightarrow [\bar{f}_2=\pi(\bar{m}_2,m_2-1)\times f_2 \wedge \bar{m}_1<\bar{m}_2 \leq \underline{\inf}(\bar{m}_1+2,m_2) \wedge I]\}$

Cette possibilité offerte par l'induction en arrière devrait être contrastée avec l'induction en avant (Lamport [77], Owicki-Gries [76a]) qui permet de spécifier ce qui a été fait (i.e. A) mais non ce qui reste à faire (i.e. R). Alors pour comprendre le programme, il faut inventer ce qui reste à faire à partir de ce qui a été fait et de la spécification de sortie.

Cette constatation pour le programme "factorielle" est en fait générale. La preuve est que $J=(A \Rightarrow R)$ où $A(\underline{a}) = [\exists \underline{\alpha} \in S[\![Pr]\!] . \varepsilon(\underline{\alpha}) \wedge t[\![Pr]\!]^*(\underline{\alpha},\underline{a})]$ et $R(\underline{a},\bar{\underline{a}}) = [t[\![Pr]\!]^*(\underline{a},\bar{\underline{a}}) \wedge \delta(\bar{\underline{a}})]$, est toujours un invariant pour l'induction positive en arrière ($-\gamma^{-1}$). Noter que nous aurions pu démontrer la complétude sémantique pour l'induction assertionnelle en avant ($-i-$) en utilisant A. Alors en utilisant le principe d'induction en arrière ($-\gamma^{-1}$) on peut spécifier le maximum d'information A sur l'état courant du programme, qui pourrait se faire en utilisant le principe d'induction en avant ($-i-$), plus une certaine information R sur ce qui reste à faire. Ceci ne permet pas toutefois de spécifier de relation entre

l'état courant et l'état initial du programme (par exemple dans le programme 4.3,2,2,2,2.5-2, on peut exprimer que $n = \underline{n}$ à la ligne 0: et donc que les valeurs initiales $\underline{n}$ et finales $\bar{n}$ de n sont égales, mais on ne peut pas exprimer que $\underline{n} = n = \bar{n}$ en tout point du programme). Pour ce faire nous proposons d'utiliser un invariant $K(\underline{s}, s, \bar{s})$ qui est la conjonction de l'invariant $I(\underline{s}, s)$ utilisé dans le principe d'induction (-Y-) et de l'invariant $J(s, \bar{s})$ utilisé dans le principe d'induction $(-Y^{-1}-)$, ce qui conduit au principe d'induction suivant :

$$
\begin{aligned}
&[\exists K \in (S^3 \to \{tt, ff\}) \,.\, \forall \underline{s}, s, s', \bar{s} \in S,\ a \in A. \\
&\quad [\varepsilon(\underline{s}) \wedge \sigma(\underline{s})] \Rightarrow [K(\underline{s}, \underline{s}, \bar{s}) \wedge K(\underline{s}, \bar{s}, \bar{s})] \\
&\quad \wedge\ [\varepsilon(\underline{s}) \wedge K(\underline{s}, s', \bar{s}) \wedge t_a(s', s) \wedge \sigma(\bar{s})] \Leftrightarrow \\
&\qquad\qquad [\varepsilon(\underline{s}) \wedge t_a(s', s) \wedge K(\underline{s}, s, \bar{s}) \wedge \sigma(\bar{s})] \qquad (-YY^{-1}-) \\
&\quad \wedge\ [\varepsilon(\underline{s}) \wedge (K(\underline{s}, \underline{s}, \bar{s}) \vee K(\underline{s}, \bar{s}, \bar{s})) \wedge \sigma(\bar{s})] \Rightarrow \psi(\underline{s}, \bar{s})] \\
&\Leftrightarrow \\
&[\forall p \in \Sigma\langle S, A, t, \varepsilon\rangle, i \in |p| \,.\, (\sigma(p_i) \Rightarrow \psi(p_0, p_i))]
\end{aligned}
$$

Pour la preuve de correction, nous avons $[\forall p \in \Sigma\langle S, A, t, \varepsilon\rangle, i \in |p|, j \in i \,.\, (\sigma(p_i) \Rightarrow K(p_0, p_j, p_i))]$. Pour la preuve de complétude sémantique, il suffit de choisir $K(\underline{s}, s, \bar{s}) = [\exists p \in \Sigma\langle S, A, t, \varepsilon\rangle, i \in |p|, j \in i \,.\, p_0 = \underline{s} \wedge p_j = s \wedge p_i = \bar{s} \wedge \sigma(\bar{s})]$.

□

4.3.2.2.3 Construction d'une méthode d'absence d'interblocages dans les programmes parallèles asynchrones par induction en arrière

La méthode de preuve developpée au paragraphe 4.3.2.2.2 pour les programmes parallèles asynchrones se généralise sans difficultés pour les programmes parallèles synchrones définis en 2.8.5. Par exemple pour la sémantique libérale des sémaphores considérée en 2.8.5.3, nous obtenons les conditions de vérification suivantes :

- Preuve séquentielle
  - $\alpha\, \ell_{i_1} : \underline{P}(Se)\,;\ \ell_{i_2} : \beta$
    $(Se > 0 \wedge P_{i_2}[Se \leftarrow Se-1]) \Rightarrow P_{i_1}$
  - $\alpha\, \ell_{i_1} : \underline{V}(Se)\,;\ \ell_{i_2} : \beta$
    $P_{i_2}[Se \leftarrow Se+1] \Rightarrow P_{i_1}$

- Preuve d'absence d'interferences

  (pour chaque point ℓ_{kj} de chaque processus Pro_k, $k \in (nvi)$) :
  - $\alpha\, \ell_{i_1} : \underline{P}(Se)\,;\ \ell_{i_2} : \beta$
    $(Se > 0 \wedge P_{i_2}[c_k \leftarrow \ell_{kj},\ Se \leftarrow Se-1] \wedge P_{kj}[c_i \leftarrow \ell_{i_2},\ Se \leftarrow Se-1]) \Rightarrow P_{kj}[c_i \leftarrow \ell_{i_1}]$
  - $\alpha\, \ell_{i_1} : \underline{V}(Se)\,;\ \ell_{i_2} : \beta$
    $(P_{i_2}[c_k \leftarrow \ell_{kj},\ Se \leftarrow Se+1] \wedge P_{kj}[c_i \leftarrow \ell_{i_2},\ Se \leftarrow Se+1]) \Rightarrow P_{kj}[c_i \leftarrow \ell_{i_1}]$

L'exécution d'un programme synchrone est globalement bloquée si les processus n'ont pas tous terminé et si tous les processus dont l'exécution n'est pas terminée sont en attente pour prendre un sémaphore. Plus formellement, si

$Pps \equiv Ps \,[\![\, Pro_0 \,\|\, \ldots \,\|\, Pro_{n-1} \,]\!]\, ; Ps'$

alors Pps est (globalement) bloqué dans l'état Δ si et seulement si $\beta[\![Pps]\!](\Delta)$ est vrai, avec :

$$\beta[\![Pps]\!](\Delta) = [\exists L_0 \in C[\![Pro_0]\!], \ldots, L_{m-1} \in C[\![Pro_{m-1}]\!], M \in M[\![Pps]\!]. \quad \Delta = \ll L_0, \ldots, L_{m-1} >, M > \wedge$$
$$[[\forall i \in m. (Pro_i \equiv \alpha L_i : \not\vdash (Se); \delta \wedge M(Se) \leq 0) \vee (Pro_i \equiv \alpha L_i :)]$$
$$\wedge [\exists j \in m. \neg (Pro_j \equiv \alpha L_j :)]]]$$

Un programme Pps est exempt d'interblocage si aucune exécution de Pps ne conduit à un état où Pps est bloqué, c'est-à-dire :

$$\forall p \in \Sigma < S[\![Pps]\!], A[\![Pps]\!], t[\![Pps]\!], \varepsilon >, i \in |p|. \neg \beta[\![Pps]\!](p_i)$$

C'est une propriété d'invariance où $\varsigma(\bar{\Delta}) = tt$ et $\psi(\underline{\Delta}, \bar{\Delta}) = \neg \beta[\![Pps]\!](\bar{\Delta})$ ne dépend pas des états initiaux. D'après 4.2.1.3, l'absence d'interblocage peut être prouvée par induction en arrière, en utilisant le principe d'induction contrapositif $(\text{-}\bar{I})$.

Le choix d'un langage pour exprimer les invariants locaux et sa sémantique est similaire à celui de la correction partielle (cf. 4.3.2.2.2.1) excepté qu'au point L_i du processus i, nous avons un invariant local de la forme :

$$\tilde{I}(L_i)(L_0, \ldots, L_{i-1}, L_{i+1}, \ldots, L_{m-1}, M)$$

au lieu de

$$\tilde{I}(L_i)(L_0, \ldots, L_{i-1}, L_{i+1}, \ldots, L_{m-1}, M, \bar{M})$$

puisqu'il n'est plus nécessaire de relier les états courants et finaux.

La dérivation des conditions de vérification correspondant au pas d'induction de $(\text{-}\bar{I})$ est la même que dans les paragraphes 4.3.2.2.2.2 et 4.3.2.2.2.3 excepté que les états mémoires finaux sont omis. Les cas qui restent sont :

. Initialisation

$$[\forall \underline{\lambda} \in S[\![Pps]\!] .\ \varepsilon(\underline{\lambda}) \Rightarrow \neg\, \gamma[\![Pps]\!](\tilde{I})(\underline{\lambda})]$$

$$= [\forall \underline{L} \in C[\![Pps]\!],\ \underline{M} \in (\mathcal{V} \to \mathcal{N}) .\ (Pps \equiv \underline{L} : \alpha \ \wedge\ \phi(\underline{M})) \Rightarrow \neg \tilde{I}(\underline{L})(\underline{M})]$$

Finalisation

$$[\forall \bar{\lambda} \in S[\![Pps]\!] .\ \beta[\![Pps]\!](\bar{\lambda}) \Rightarrow \gamma[\![Pps]\!](\tilde{I})(\bar{\lambda})]$$

quand $\bar{\lambda}$ n'est pas de la forme $\ll L_0, \dots, L_{m-1} \gg, M \gg$, $\beta[\![Pps]\!](\bar{\lambda})$ est faux de sorte que la condition est trivialement vérifiée, sinon elle est équivalente à :

$$[\forall L_0 \in C[\![Prs_0]\!], \dots, L_{m-1} \in C[\![Prs_{m-1}]\!],\ M \in M[\![Pps]\!] .$$

$$[\, Pps \equiv Ps[\![Prs_0 \| \dots \| Prs_{m-1}]\!] ; Ps' \ \wedge\ (\exists j \in m .\ \neg(Prs_j \equiv \alpha L_j :)) \ \wedge$$

$$(\forall i \in m .\ (Prs_i \equiv \alpha L_i :) \vee (Prs_i \equiv \alpha L_i : \not\!\!\downarrow(se) ; \delta \ \wedge\ M(se) \leq 0))]$$

$$\Rightarrow \bigwedge_{k \in m} \tilde{I}(L_k)(L_0, \dots, L_{k-1}, L_{k+1}, \dots, L_{m-1}, M)]$$

Informellement, pour tout tuple $L_0, \dots, L_{m-1}$ d'étiquettes telles que

- L_i désigne une commande $L_i : \not\!\!\downarrow(se)$;
  (auquel cas $\underline{\text{bloqué}}[\![Pps]\!](i, L_i)(M) = [M(se) \leq 0]$)

ou bien

- L_i désigne le point de sortie du processus Prs_i
  (auquel cas $\underline{\text{bloqué}}[\![Pps]\!](i, L_i)(M) = tt$)

et telles qu'elles ne désignent pas toutes des points de sortie, nous devons montrer que :

$$\bigwedge_{i \in m} \underline{\text{bloqué}}[\![Pps]\!](i, L_i)(M) \Rightarrow \bigwedge_{i \in m} \tilde{I}(L_i)(L_0, \dots, L_{i-1}, L_{i+1}, \dots, L_{m-1}, M)$$

Exemple 4.3.2.2.3-1

Considérons le programme très simple suivant :

```
0:
   ||
      11: while true do 12: P(m) ; 13: V(m); 14: od ; 15:
   ||
      21: while true do 22: P(m) ; 23: V(m); 24: od ; 25:
   ||;
3:
```

Les conditions de vérification sont :

- Initialisation

  $\phi(m) \Rightarrow \neg P_0(m)$

- Induction

  . $[P_{11}(21, m) \wedge P_{21}(11, m)] \Rightarrow P_0(m)$

  . Preuve séquentielle du processus 1 :

  $P_{12}(c_2, m) \Rightarrow [P_{11}(c_2, m) \wedge P_{14}(c_2, m)]$

  $[m>0 \wedge P_{13}(c_2, m-1)] \Rightarrow P_{12}(c_2, m)$

  $P_{14}(c_2, m+1) \Rightarrow P_{13}(c_2, m)$

  . Absence d'interférences du processus 1 avec la preuve du processus 2 :

  Pour $j = 1, \ldots, 5$

  $[P_{12}(2_j, m) \wedge P_{2_j}(12, m)] \Rightarrow [P_{2_j}(11, m) \wedge P_{2_j}(14, m)]$

  $[m>0 \wedge P_{13}(2_j, m-1) \wedge P_{2_j}(13, m-1)] \Rightarrow P_{2_j}(12, m)$

  $[P_{14}(2_j, m+1) \wedge P_{2_j}(14, m+1)] \Rightarrow P_{2_j}(13, m)$

  . La preuve séquentielle du processus 2 et la preuve d'absence d'interférences du processus 2 avec la preuve du processus 1 est similaire.

  . $P_3(m) \Rightarrow [P_{15}(25, m) \wedge P_{25}(15, m)]$

- Finalisation

  . Interblocage possible en 12 et 22 :

  $m \leq 0 \Rightarrow [P_{12}(22, m) \wedge P_{22}(12, m)]$

. Interblocage possible en 12 et 25 :

$m \leq 0 \Rightarrow [P_{12}(25, m) \wedge P_{25}(12, m)]$

. Interblocage possible en 15 et 22 :

$m \leq 0 \Rightarrow [P_{15}(22, m) \wedge P_{22}(15, m)]$

L'esquisse de la preuve est :

```
0: {m<1}
   ||
        11: {[m≤0 ∧ c2∈{21,22,24}]∨[m<0 ∧ c2=23]∨[c2=25]}
            while true do
        12:     {[m≤0 ∧ c2∈{21,22,24}]∨[m<0 ∧ c2=23]∨[c2=25]}
                P(m);
        13:     {[m<0 ∧ c2∈{21,22,24}]∨[m<-1 ∧ c2=23]∨[c2=25]}
                V(m);
        14:     {[m≤0 ∧ c2∈{21,22,24}]∨[m<0 ∧ c2=23]∨[c2=25]}
            od;
        15: {tt}
   ||
        21: {[m≤0 ∧ c1∈{11,12,14}]∨[m<0 ∧ c1=13]∨[c1=15]}
            while true do
        22:     {[m≤0 ∧ c1∈{11,12,14}]∨[m<0 ∧ c1=13]∨[c1=15]}
                P(m);
        23:     {[m<0 ∧ c1∈{11,12,14}]∨[m<-1 ∧ c1=13]∨[c1=15]}
                V(m);
        24:     {[m≤0 ∧ c1∈{11,12,14}]∨[m<0 ∧ c1=13]∨[c1=15]}
            od;
        25: {tt}
   ||;
3: {tt}
```

Informellement, en chaque point du programme nous donnons une condition sur m qui est nécessaire (mais peut-être pas suffisante) pour que le programme soit bloqué plus tard. Puisque cette condition n'est pas satisfaite par les états d'entrée quand $m \geq 1$, le programme ne peut pas être bloqué.

□

4.3.2.2.4 Construction d'une méthode de preuve d'exclusion mutuelle dans les programmes parallèles asynchrones par induction en arrière

Deux sections d'un programme sont en exclusion mutuelle si elles ne contiennent pas de commandes qui peuvent s'exécuter en même temps. Supposons que cs_i (respectivement cs_j) est un prédicat qui caractérise l'ensemble des étiquettes appartenant à la section critique du processus Prs_i (respectivement Prs_j) du programme :

$Pps \equiv Ps[\![Prs_0 \| \ldots \| Prs_{m-1}]\!] ; Ps'$

Les sections critiques sont en exclusion mutuelle si et seulement si :

$\forall p \in \Sigma < S[\![Pps]\!], A[\![Pps]\!], t[\![Pps]\!], \varepsilon >, k \in |p| .\ \underset{\sim}{me}[\![Pps]\!](i,j)(cs_i, cs_j)(p_k)$

où

$\underset{\sim}{me}[\![Pps]\!](i,j)(cs_i, cs_j)(\bar{s}) = [\forall L_0 \in C[\![Prs_0]\!], \ldots, L_{m-1} \in C[\![Prs_{m-1}]\!], M \in M[\![Pps]\!].$

$\bar{s} = \langle\langle L_0, \ldots, L_{m-1} \rangle, M \rangle \Rightarrow \neg (cs_i(L_i) \wedge cs_j(L_j))]$

L'exclusion mutuelle est une propriété d'invariance qui peut être démontrée en utilisant le principe d'induction contrapositif en arrière ($\neg\tilde{I}$). Les conditions de vérification sont alors similaires à celles pour l'absence d'interblocages, excepté pour la finalisation qui est :

- Finalisation

$[\neg \underset{\sim}{me}[\![Pps]\!](i,j)(cs_i, cs_j)(\bar{s}) \Rightarrow \gamma[\![Pps]\!](\tilde{I})(\bar{s})]$

$= [(cs_i(L_i) \wedge cs_j(L_j)) \Rightarrow \bigwedge_{k \in m} \tilde{I}(L_k)(L_0, \ldots, L_{k-1}, L_{k+1}, \ldots, L_{m-1}, M)]$

Informellement, pour toutes les étiquettes L_i de Prs_i telles que $cs_i(L_i)$ et L_j de Prs_j telles que $cs_j(L_j)$,

$\tilde{I}(L_i)(c_0, \ldots, c_{j-1}, L_j, c_{j+1}, \ldots, c_{i-1}, c_{i+1}, \ldots, c_{m-1}, M)$

et

$\tilde{I}(L_j)(c_0, \ldots, c_{j-1}, c_{j+1}, \ldots, c_{i-1}, L_i, c_{i+1}, \ldots, c_{m-1}, M)$

doivent être vrais. De plus, pour toutes les étiquettes L_k du processus k, $k \in (m \vee \{i,j\})$ nous devons avoir :

$$[cs_i(c_i) \wedge cs_j(c_j)] \Rightarrow \tilde{I}(L_k)(c_0, \ldots, c_{k-1}, c_{k+1}, \ldots, c_{m-1}, M)$$

Exemple 4.3.2.2.4-1

Les points 13 et 23 du programme 4.3.2.2.3-1 sont en exclusion mutuelle quand $m \leq 1$. Les conditions de vérification sont celles de 4.3.2.2.3-1 excepté pour la finalisation qui est :

$P_{13}(23, m)$

$P_{23}(13, m)$

L'esquisse de la preuve est :

```
0: {m>1
   [[
        11: {[m>0 ∧ c2=23]v[m>1 ∧ c2∈{21,22,24}]}
            while true do
        12:     {[m>0 ∧ c2=23]v[m>1 ∧ c2∈{21,22,24}]}
                ℘(m);
        13:     {[c2=23]v[m>0 ∧ c2∈{21,22,24}]}
                𝔳(m);
        14:     {[m>0 ∧ c2=23]v[m>1 ∧ c2∈{21,22,24}]}
            od;
        15: {##}

        21: {[m>0 ∧ c1=13]v[m>1 ∧ c1∈{11,12,14}]}
            while true do
        22:     {[m>0 ∧ c1=13]v[m>1 ∧ c1∈{11,12,14}]}
                ℘(m);
        23:     {[c1=13]v[m>0 ∧ c1∈{11,12,14}]}
                𝔳(m);
        24:     {[m>0 ∧ c1=13]v[m>1 ∧ c1∈{11,12,14}]}
            od;
        25: {##}
   ]];
3: {##}
```

□

4.3.2.2.5 Construction d'une méthode de preuve de non-terminaison de programmes parallèles par induction en arrière

Un programme Pps ne se termine pas si et seulement si

$$\forall p \in \Sigma^{<} S[\![Pps]\!], A[\![Pps]\!], t[\![Pps]\!], \varepsilon^{>}, i \in |p|.\ \Psi(p_i)$$

où

$$\Psi(\bar{x}) = \neg [\exists \bar{L} \in C[\![Pps]\!], \bar{M} \in M[\![Pps]\!].\ \bar{x} = \langle \bar{L}, \bar{M} \rangle \wedge Pps \equiv \alpha \bar{L} :]$$

C'est une propriété que nous pouvons montrer par induction en arrière en utilisant $(\dot{-}\bar{i})$. Les conditions de vérification sont celles du paragraphe 4.3.2.2.4 excepté pour la finalisation qui est :

$$\forall \bar{M} \in M[\![Pps]\!].\ \check{I}(\bar{L})(\bar{M}) \qquad \text{où } Pps \equiv \alpha \bar{L} :$$

Exemple 4.3.2.2.5-1

Les conditions de vérification sont celles de l'exemple 4.3.2.2.4-1 excepté pour la finalisation qui est :

$P_3(m)$

Esquisse de la preuve :

0: {ff}
‖
11: {ff} while true do 12: {ff} $\underline{p}(m)$; 13: {ff} $\underline{v}(m)$; 14: {ff} od; 15: {tt}
‖
21: {ff} while true do 22: {ff} $\underline{p}(m)$; 23; {ff} $\underline{v}(m)$; 24: {ff} od; 25: {tt}
‖;
3: {tt}

□

4.3.2.2.6 Conclusion sur la preuve de propriétés d'invariance de programmes par induction en arrière

La méthode de Morris-Wegbreit [77] dite "subgoal induction" n'a jamais remporté le succès qu'a eu la méthode de Floyd [67]. Diverses raisons ont été avancées dont certaines sont incorrectes (comme l'incomplétude sémantique Misra [78], cf. 4.3.2.2.1.3) ou superficielles (comme les limitations concernant les programmes qui ne se terminent pas Morris-Wegbreit [77], cf. 4.3.2.2.1.5). Dijkstra [82, p.224-225] démontre un cas particulier du théorème 4.2.1.4 N2 à propos d'un programme séquentiel consistant en une boucle "while" et conclut (à la page xiii) que "we can ignore subgoal induction because it is nothing but the Invariance Theorem in a complicated disguise". Cette conclusion porte sur l'équivalence des méthodes de preuve et reste donc valable pour la méthode de preuve de correction partielle que nous avons introduite (en généralisant la méthode de Morris-Wegbreit dite "subgoal induction") quand on la compare par exemple aux méthodes de preuve d'invariance en avant à la Lamport [77], Owicki-Gries [76a] (qui généralisent la méthode de Floyd [67]). Cependant comme nous l'avons remarqué en 4.3.2.2.2.5-3, l'invariant A associé en tout point d'un programme pour une méthode de preuve en avant peut être utilisé pour une méthode de preuve en arrière dans la forme A⇒R. Quand la preuve est utilisée comme commentaires, ceci est utile pour le lecteur du programme puisque A décrit ce qui a été fait jusqu'à ce point et R ce qui reste à faire. R doit être inventé par le lecteur du programme quand les méthodes de preuve en avant sont utilisées.

Ces arguments nous semblent en fait peu convaincants. La véritable raison de l'insuccès de l'induction en arrière nous semble être que pour les méthodes de preuve en avant, les mêmes invariants peuvent être utilisés pour la correction partielle, l'absence d'interblocages,

l'exclusion mutuelle, l'absence d'erreurs à l'exécution, la non-terminaison, etc. Ceci parce que dans l'induction en avant, le même principe d'induction $(\text{-}\mathcal{Y})$ peut être utilisé pour la preuve de toutes ces propriétés d'invariance. Ce qui n'est pas le cas pour les méthodes de preuve en arrière pour lesquelles nous devons utiliser deux principes d'induction $(\text{-}\mathcal{Y}^{-1})$ et $(\text{-}\bar{\imath})$ qui conduisent à des conditions de vérifications différentes.

Un compromis heureux pourrait consister à utiliser une combinaison des principes d'induction en avant et en arrière comme nous l'avons proposé en 4.3.2.2.2.5-3. Pour les programmes parallèles, par exemple, il faut souvent démontrer une propriété de la forme $\forall p \in \Sigma\langle S,A,t,\varepsilon\rangle, i \in |p|.\ (\sigma(p_i) \Rightarrow \psi(p_0,p_i))$ (comme la correction partielle) et une propriété de la forme $\forall p \in \Sigma\langle S,A,t,\varepsilon\rangle, i\in|p|.\ \Gamma(p_i)$ (où Γ est une conjonction de conditions correspondant par exemple à l'absence d'interblocages globaux permanents, l'exclusion mutuelle de certaines sections critiques, etc.). On pourra alors choisir le principe d'induction suivant :

$$
\begin{aligned}
&[\exists K \in (S^3 \to \{tt, ff\}).\ \forall \underline{s}, s, s', \bar{s} \in S, a \in A \\
&\qquad [\varepsilon(\underline{s}) \wedge \sigma(\bar{s})] \Rightarrow [K(\underline{s},\underline{s},\bar{s}) \wedge K(\underline{s},\bar{s},\bar{s})] \\
&\quad \wedge\ [\varepsilon(\underline{s}) \wedge K(\underline{s},s,\bar{s}) \wedge t_a(s,s') \wedge \sigma(\bar{s})] \Leftrightarrow \\
&\qquad\qquad [\varepsilon(\underline{s}) \wedge t_a(s,s') \wedge K(\underline{s},s',\bar{s}) \wedge \sigma(\bar{s})] \\
&\quad \wedge\ [\varepsilon(\underline{s}) \wedge (K(\underline{s},\underline{s},\bar{s}) \vee K(\underline{s},\bar{s},\bar{s})) \wedge \sigma(\bar{s})] \Rightarrow \psi(\underline{s},\bar{s}) \\
&\quad \wedge\ [\varepsilon(\underline{s}) \wedge K(\underline{s},s,\bar{s})] \Rightarrow \Gamma(s)\ ] \\
&\Longleftrightarrow \\
&[\forall p \in \Sigma\langle S,A,t,\varepsilon\rangle, i \in |p|.\ [\forall j \in i. \neg\sigma(p_j) \wedge \sigma(p_i)] \Rightarrow [\forall j \in i. \Gamma(p_j) \wedge \psi(p_0,p_i)]]
\end{aligned}
$$

(La preuve de correction consiste essentiellement à démontrer que $\forall p \in \Sigma\langle S,A,t,\varepsilon\rangle, i\in|p|.\ [\forall j \in i.\ \neg\sigma(p_j) \wedge \sigma(p_i)] \Rightarrow [\forall j \in i.\ K(p_0,p_j,p_i)]$ et

la preuve de complétude se fait en choisissant $K(\underline{\Delta}, \Delta, \bar{\Delta}) =$
$[\exists p \in \Sigma^{<s,A,t,\varepsilon>}, i \in |p|, j \leq i \,.\, p_0 = \underline{\Delta} \wedge p_j = \Delta \wedge p_i = \bar{\Delta} \wedge \forall j \in i \,.\, \neg\sigma(p_j) \wedge \sigma(p_i)]$).

Ce principe d'induction est implicitement utilisé dans les preuves informelles que donnent Ricart-Agrawala [81].

4.3.2.3 Construction d'une méthode de preuve pour les programmes parallèles communicants

En appliquant la méthode de construction définie au paragraphe 4.3.1, nous avons proposé dans Cousot-Cousot [80a] une méthode de preuve pour un sous-ensemble de CSP dont l'originalité consistait à proposer une décomposition des preuves pour les programmes CSP en des preuves pour chaque processus (ne faisant référence qu'à l'état du processus), pour chaque canal (faisant référence à l'état de tous les processus au moment des communications (la référence aux seuls états des processus qui communiquent sur le canal étant incomplète)) et en des preuves d'absence d'interférence (entre les assertions associées aux canaux et l'exécution des processus). L'idée essentielle était de considérer qu'entre l'initialisation et l'attente du premier rendez-vous, ou entre deux attentes de rendez-vous ou entre l'attente du dernier rendez-vous et la terminaison, l'exécution d'un processus est indivisible. Cette idée n'est plus valable pour les programmes communicants considérés au paragraphe 2.8.3 pour la raison que les processus peuvent, entre deux communications par les canaux, interférer au moyen des variables globales partagées.

Soit

$$Ppc \equiv Ps \,[\![\, Prc_0 \parallel \ldots \parallel Prc_{n-1} \,]\!]\, ; Ps'$$

Pour construire une méthode de preuve basée sur le principe d'induction (Y), nous pouvons utiliser la décomposition définie par:

$$As[\![Ppc]\!] = (S[\![Ppc]\!]^2 \to \{tt, ff\})$$

$$\tilde{As}[\![Ppc]\!] = (\tilde{Prel}[\![Ppc]\!] \cup \tilde{Proc}[\![Ppc]\!] \cup \tilde{Postl}[\![Ppc]\!])$$

Une relation reliant les états mémoire courants et initiaux est associée à chaque point du prélude et du postlude :

$$\tilde{Prél}\,[\![Ppc]\!] = \{\tilde{I} : \exists \underline{L} \in C[\![Ps]\!].\ \tilde{I}(\underline{L}) \in (M[\![Ppc]\!]^2 \to \{tt, ff\})\}$$

$$\tilde{Post}\,[\![Ppc]\!] = \{\tilde{I} : \exists \underline{l} \in C[\![Ps']\!].\ \tilde{I}(\underline{l}) \in (M[\![Ppc]\!]^2 \to \{tt, ff\})\}$$

Une relation reliant l'état mémoire et contrôle courant à l'état mémoire initial est associée à chaque point de chaque processus :

$$\tilde{Proc}\,[\![Ppc]\!] = \{\tilde{I} : \exists i \in m, L \in C[\![Prc_i]\!].\ \tilde{I}(i)(L) \in (M[\![Ppc]\!] \times (\underset{j \in (m \setminus i)}{\pi} C[\![Prc_j]\!]) \times M[\![Ppc]\!] \to \{tt, ff\})\}$$

La signification d'un tel vecteur $\tilde{I}$ est définie formellement par la fonction sémantique $\gamma[\![Ppc]\!](\tilde{I}) = I$, où par cas :

$$I(\langle \underline{L}, \underline{M} \rangle, \langle L, M \rangle) = \tilde{I}(L)(\underline{M}, M) \quad \text{quand} \quad Ppc \equiv \underline{L} : P \ \wedge\ L \in (C[\![Ps]\!] \cup C[\![Ps']\!])$$

$$I(\langle \underline{L}, \underline{M} \rangle, \langle\langle L_0, \ldots, L_{m-1} \rangle, M \rangle) =$$

$$\bigwedge_{i \in m} \tilde{I}(L_i)(\underline{M}, L_0, \ldots, L_{i-1}, L_{i+1}, \ldots, L_{m-1}, M) \quad \text{quand} \quad Ppc \equiv \underline{L} : P$$

Nous ne donnerons pas le détail de la construction des conditions de vérification correspondant à cette décomposition, ni la vérification de la complétude sémantique. En résumé, les conditions de vérification sont les suivantes (si on exclut les commandes de communication, ce sont celles proposées par Lamport [76a] et sont similaires aux conditions de Owicki-Gries [77] (sauf pour l'usage de variables auxiliaires qui est remplacé par celui des compteurs ordinaux). Avec les commandes de communication elles sont similaires à celles proposées par Levin [79] (les variables auxiliaires étant remplacées par des compteurs ordinaux)) :

Les invariants $P_i(\underline{x}, x)$ associés aux points L_i des prélude et postlude relient l'état initial $\underline{x}$ à l'état courant x des variables quand l'exécution est en L_i.

Les invariants $P_{ij}(\underline{x}, c_0, \ldots, c_{i-1}, c_{i+1}, \ldots, c_{m-1}, x)$ associés aux points L_{ij} du processus Prc_i relient l'état initial $\underline{x}$ des variables à l'état

courant x des variables et l'état $c_0, \ldots, c_{i-1}, c_{i+1}, \ldots, c_{m-1}$ de contrôle des autres processus $Prc_0, \ldots, Prc_{i-1}, Prc_{i+1}, \ldots, Prc_{m-1}$ quand l'exécution est au point L_{ij} du processus Prc_i.

- Preuve séquentielle

(pour le prélude Ps, le postlude Ps' et chaque processus $Prc_0, \ldots, Prc_{m-1}$)

. Initialisation (prélude)

$L_1 : \alpha$ $\qquad\qquad$ $\forall \bar{x}.\ P_1(\bar{x}, \bar{x})$

. Commande nulle

$\alpha\ L_1 :$ skip ; $L_2 : \beta$ $\qquad\qquad$ $P_1 \Rightarrow P_2$

. Commande d'affectation

$\alpha\ L_1 :\ V := E\ ;\ L_2 : \beta$ $\qquad\qquad$ $P_1 \Rightarrow P_2[V \leftarrow E]$

$\alpha\ L_1 :\ V := ?\ ;\ L_2 : \beta$ $\qquad\qquad$ $\forall v \in \mathfrak{D}.\ P_1 \Rightarrow P_2[V \leftarrow v]$

. Commande conditionnelle

$\alpha\ L_1 :$ if B then

$L_2 :$ β $\qquad\qquad$ $[P_1 \wedge B] \Rightarrow P_2$

$L_3 :$

else $\qquad\qquad$ $[P_1 \wedge \neg B] \Rightarrow P_4$

$L_4 :$ γ $\qquad\qquad$ $[P_3 \vee P_5] \Rightarrow P_6$

$L_5 :$

fi ;

$L : \delta$

. Itération

$\alpha\ L_1 :$ while B do

$L_2 :$ β $\qquad\qquad$ $(P_1 \vee P_3) \Rightarrow [(B \Rightarrow P_2) \wedge (\neg B \Rightarrow P_4)]$

$L_3 :$

od ;

$L_4 : \delta$

. Section critique (dans le processus Prc_i)

α L_{i_1} : $\not$ L_{i_2} : β L_{i_3} : $\not$; L_{i_4} : γ

A chaque point L_{ij} du corps $L_{i_2} : \beta\, L_{i_3}$: de la section critique, une relation invariante $P_{ij}(x', x)$ relie les valeurs courantes x en L_{ij} aux valeurs initiales x' en L_{i_2} des variables dans la section critique. Les conditions de vérification correspondantes sont celles des programmes séquentiels avec $\forall x'.\ P_{i_2}(x', x')$. Il faut y ajouter

$$[\, P_{i_1}[x \leftarrow x'] \wedge P_{i_3}(x', x)\,] \Rightarrow P_{i_4}$$

. Sortie d'une commande alternative (dans le processus Prc_i)

α $\underline{\text{se}}$ β_0 ; L_{i_0} : $\underline{\text{or}}$ ... $\underline{\text{or}}$ $\beta_{\ell-1}$; $L_{i_{\ell-1}}$: $\underline{\text{es}}$; L_{i_ℓ} : β

$$\forall j \in \ell.\ P_{i_j} \Rightarrow P_{i_\ell}$$

- Initialisation / Finalisation du parallélisme

α L_m : $[\![\, L_0 : \beta_0 \parallel \ldots \parallel L_{m-1} : \beta_{m-1} \,]\!]$ β

$$P_m(\underline{x}, x) \Rightarrow [\, \bigwedge_{i \in m} P_i(\underline{x}, L_0, \ldots, L_{i-1}, L_{i+1}, \ldots, L_{m-1}, x)\,]$$

α $[\![\, \beta_0\, L_0 : \parallel \ldots \parallel \beta_{m-1}\, L_{m-1} : \,]\!]$; L_m : β

$$[\, \bigwedge_{i \in m} P_i(\underline{x}, L_0, \ldots, L_{i-1}, L_{i+1}, \ldots, L_{m-1}, x)\,] \Rightarrow P_m(\underline{x}, x)$$

- Preuve de correction des communications

  . Ch!E équivalent à ae true ; ch!E then skip ; es se traite comme ci-dessous

  . Ch?V équivalent à ae true ; ch?V then skip ; es se traite comme ci-dessous

  . Pour toutes paires de commandes alternatives dans des processus différents i et j,

$$\alpha L_{i_1} : \underline{ae} \ \dots \ \underline{or} \ B_1 \ ; \ ch!E \ \underline{then} \ L_{i_2} : \beta \ \underline{or} \ \dots \ \underline{es} \ \gamma$$

$$\alpha' L_{j_1} : \underline{ae} \ \dots \ \underline{or} \ B_2 \ ; \ ch?V \ \underline{then} \ L_{j_2} : \beta' \ \underline{or} \ \dots \ \underline{es} \ \gamma'$$

$$\left(P_{i_1}[c_j \leftarrow L_{j_1}] \wedge B_1 \wedge P_{j_1}[c_i \leftarrow L_{i_1}] \wedge B_2 \right)$$
$$\Rightarrow \left(P_{i_2}[c_j \leftarrow L_{j_2}, V \leftarrow E] \wedge P_{j_2}[c_i \leftarrow L_{i_2}, V \leftarrow E] \right)$$

- Preuve d'absence d'interférences

Pour tout point L_{kl} de tout processus $Proc_k$, $k \in (m \cup i)$,

  . Commande nulle

$$\alpha L_{i_1} : \underline{skip} \ ; \ L_{i_2} : \beta$$
$$\left(P_{i_1}[c_k \leftarrow L_{kl}] \wedge P_{kl}[c_i \leftarrow L_{i_1}] \right) \Rightarrow P_{kl}[c_i \leftarrow L_{i_2}]$$

  . Commande d'affectation

$$\alpha L_{i_1} : V := E \ ; \ L_{i_2} : \beta$$
$$\left(P_{i_1}[c_k \leftarrow L_{kl}] \wedge P_{kl}[c_i \leftarrow L_{i_1}] \right) \Rightarrow P_{kl}[c_i \leftarrow L_{i_2}, V \leftarrow E]$$

$$\alpha L_{i_1} : V := ? \ ; \ L_{i_2} : \beta$$
$$\left(P_{i_1}[c_k \leftarrow L_{kl}] \wedge P_{kl}[c_i \leftarrow L_{i_2}] \right) \Rightarrow \left(\forall v \in \mathcal{D}. \ P_{kl}[c_i \leftarrow L_{i_2}, V \leftarrow v] \right)$$

. Commande conditionnelle

α L_{i_1} : **if** B **then**

L_{i_2} :

β

L_{i_3} :

else

L_{i_4} :

γ

L_{i_5} :

fi ;

L_{i_6} : δ

$$(P_{i_1}[c_k \leftarrow L_{k\ell}] \wedge P_{k\ell}[c_i \leftarrow L_{i_1}]) \Rightarrow [(B \Rightarrow P_{k\ell}[c_i \leftarrow L_{i_2}]) \wedge (\neg B \Rightarrow P_{k\ell}[c_i \leftarrow L_{i_4}])]$$

$$[(P_{i_3}[c_k \leftarrow L_{k\ell}] \wedge P_{k\ell}[c_i \leftarrow L_{i_3}]) \vee (P_{i_5}[c_k \leftarrow L_{k\ell}] \wedge P_{k\ell}[c_i \leftarrow L_{i_5}])] \Rightarrow P_{k\ell}[c_i \leftarrow L_{i_6}]$$

. Itération

α L_{i_1} : **while** B **do**

L_{i_2} :

β

L_{i_3} :

od ;

L_{i_4} : γ

$$[(P_{i_1}[c_k \leftarrow L_{k\ell}] \wedge P_{k\ell}[c_i \leftarrow L_{i_1}]) \vee (P_{i_3}[c_k \leftarrow L_{k\ell}] \wedge P_{k\ell}[c_i \leftarrow L_{i_3}])]$$
$$\Rightarrow [(B \Rightarrow P_{k\ell}[c_i \leftarrow L_{i_2}]) \wedge (\neg B \Rightarrow P_{k\ell}[c_i \leftarrow L_{i_4}])]$$

. Section critique

α L_{i_1} : ⊀

L_{i_2} :

β

L_{i_3} :

⊁ ;

L_{i_4} : γ

$$[P_{i_1}[c_k \leftarrow L_{k\ell}, x \leftarrow x'] \wedge P_{k\ell}[c_i \leftarrow L_{i_3}, x \leftarrow x'] \wedge P_{i_3}(x', x)] \Rightarrow P_{k\ell}[c_i \leftarrow L_{i_4}]$$

. Communication $(i \neq k,\ j \neq k)$

$\alpha\, L_{i_1}$: **se** ... **or** B_1 ; ch!E **then** L_{i_2} : β **or** ... **es** γ

$\alpha'\, L_{j_1}$: **se** ... **or** B_2 ; ch?v **then** L_{j_2} : β' **or** ... **es** γ'

$$[\, P_{i_1}[c_j \leftarrow L_{j_2}, c_k \leftarrow L_{kl}] \wedge B_1 \wedge P_{j_2}[c_i \leftarrow L_{i_2}, c_k \leftarrow L_{kl}] \wedge B_2 \wedge P_{kl}[c_i \leftarrow L_{i_1}, c_j \leftarrow L_{j_1}]]$$
$$\Rightarrow \quad P_{kl}[c_i \leftarrow L_{i_2},\ c_j \leftarrow L_{j_2},\ v \leftarrow E\,]$$

4.3.2.4 Comparaison des méthodes de preuve pour les programmes parallèles connues dans la littérature

La comparaison des méthodes de preuve de propriétés d'invariance de programmes parallèles (comme Ashcroft [77], Hoare [75], Howard [76], Keller [76], Lamport [77], Mazurkiewicz [77], Newton [75], Owicki-Gries [76a, 76b], etc.) est souvent difficile à cause de la diversité des formalismes syntaxiques qui sont utilisés pour représenter les algorithmes. Nous proposons de les comparer en montrant que toutes ces méthodes dérivent du même principe d'induction (-Y-) (ou ses variantes (-Y), (-i-), (i) ou leurs transformés par ν) et ne diffèrent que par la façon de décomposer l'invariant global utilisé dans (-Y-) en invariants locaux associés à des points du programme.

4.3.2.4.1 Utilisation d'un seul invariant global

Dans les méthodes de preuve d'invariance de Ashcroft [75] et Keller [76], un seul invariant est utilisé dans la preuve. Ashcroft justifie sa méthode de preuve par un théorème qui peut s'exprimer en utilisant notre formalisme comme suit :

$$[\forall \Delta, \Delta' \in S.$$
$$\varepsilon(\Delta) \Rightarrow \psi(\Delta)$$
$$\wedge\ [\exists \underline{\Delta} \in S.\ \varepsilon(\underline{\Delta}) \wedge t^*(\underline{\Delta}, \Delta) \wedge \psi(\Delta) \wedge (\exists a \in A.\ t_a(\Delta, \Delta'))] \Rightarrow \psi(\Delta')]$$
$$\Leftrightarrow$$
$$[\forall \Delta \in S.\ (\exists \underline{\Delta} \in S.\ \varepsilon(\underline{\Delta}) \wedge t^*(\underline{\Delta}, \Delta)) \Rightarrow \psi(\Delta)]$$

Puis Ashcroft remarque ensuite qu'on ne connaît pas exactement l'ensemble des états Δ satisfaisant $[\exists \underline{\Delta} \in S.\ \varepsilon(\underline{\Delta}) \wedge t^*(\underline{\Delta}, \Delta)]$ de sorte que "we could have left this term out of the verification condition entirely".

Il ajoute "however, if the impossibility of reaching certain states is crucial for certain properties of a program to hold" then we can "explicitely incorporate the impossibility into the assertions we wish prove valid and check the above conditions for all states". Autrement dit, on peut être amené à remplacer ψ par I tel que $\forall s \in S.\ I(s) \Rightarrow \psi(s)$. Il s'agit donc bien du principe d'induction (-i) dont la correction a été démontreé ultérieurement par Keller [76].

Dans les deux cas, on choisit l'ensemble des actions comme étant l'ensemble des affectations et tests de chaque processus du programme, de sorte que la méthode consiste à utiliser un seul invariant global et autant de conditions de vérification qu'il y a d'actions. Dans ce cas la méthode de preuve consiste à appliquer directement le principe d'induction. Cependant l'utilisation d'un seul invariant pour décrire le comportement d'un grand programme peut être inadéquate.

Exemple 4.3.2.4.1-1

Si on veut démontrer la correction partielle du programme $[\![11: x:=x+1\ ;\ 12: \ \|\ 21: x:=x+2\ ;\ 22:]\!]$ relativement à une spécification ϕ, ψ, il faudra trouver un invariant I tel que :

$$\phi(x) \Rightarrow I(11, 21, x)$$

$$[I(11, c_2, x') \wedge x = x' + 1] \Rightarrow I(12, c_2, x)$$

$$[I(c_1, 21, x') \wedge x = x' + 2] \Rightarrow I(c_1, 22, x)$$

$$I(12, 22, x) \Rightarrow \psi(x)$$

□

4.3.2.4.2 Utilisation d'invariants sur les variables associés à chaque état de contrôle

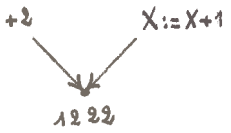
Une des premières tentatives de décomposition de l'invariant global a été faite par Ashcroft-Manna [70]. Leur technique consiste à transformer un programme parallèle en un programme non déterministe équivalent puis à appliquer la méthode de Floyd-Hoare-Naur qui utilise un invariant local, portant sur les variables du programme, associé à chaque point du programme transformé

Exemple 4.3.2.4.2-1

$[\![11 : X := X+1 ; 12 : \| 21 : X := X+2 ; 22 :]\!]$

est transformé en :

1221



auquel on applique la méthode de Floyd [67], ce qui donne :

$\phi(x) \Rightarrow I_{1121}(x)$

$[I_{1121}(x') \wedge x = x'+1] \Rightarrow I_{1221}(x)$

$[I_{1121}(x') \wedge x = x'+2] \Rightarrow I_{1122}(x)$

$[I_{1221}(x') \wedge x = x'+2] \Rightarrow I_{1222}(x)$

$[I_{1122}(x') \wedge x = x'+1] \Rightarrow I_{1222}(x)$

$I_{1222}(x) \Rightarrow \psi(x)$

□

Une autre façon d'expliquer la même idée qui évite d'avoir à transformer le programme consiste à dire que la méthode de Floyd [67] et Ashcroft-Manna [70] s'applique à des programmes dont l'ensemble des états est $S = C \times M$ où C est l'ensemble des états de contrôle et M l'ensemble des états mémoires et consiste à appliquer le principe d'induction (-i) avec la décomposition :

$$A_S = (C \times M \to \{tt, ff\})$$
$$A\tilde{S} = (C \to (M \to \{tt, ff\}))$$

$$\alpha \in (A_S \to A\tilde{S})$$
$$\alpha(I)_c(m) = I(\langle c, m \rangle)$$

$$\gamma \in (A\tilde{S} \to A_S)$$
$$\gamma(\tilde{I})(\langle c, m \rangle) = \tilde{I}_c(m)$$

ce qui conduit évidemment à des méthodes correctes et sémantiquement complètes (cf. 4.3.1.7-1 et 4.3.1.8-2, les hypothèses de 4.3.1.6.2.7 étant satisfaites).

Si cette décomposition est appliquée à un programme parallèle $[\![Pr_0 \| \dots \| Pr_{m-1}]\!]$ où chaque processus Pr_i a m_i points de contrôle, il y a $m_0 \times m_1 \times \dots \times m_{m-1}$ invariants locaux à considérer, ce qui conduit évidemment à des preuves très longues. Ceci peut être évité en utilisant des décompositions moins fines.

4.3.2.4.3 Utilisation d'invariants sur les variables associés à chaque point de contrôle du programme

Considérons un programme dont l'ensemble des états est $S = C_0 \times \ldots \times C_{m-1} \times M$ où $C_i, i \in m$ est l'ensemble des points de contrôle du $i^{ème}$ processus et M l'état mémoire.

On peut remplacer l'invariant global utilisé dans le principe d'induction (-i) par des invariants locaux portant sur l'état mémoire et associés à chaque point de contrôle. Ceci revient à choisir (on suppose $i \neq j \Rightarrow (C_i \cap C_j = \phi)$) :

$$A_S = (C_0 \times \ldots \times C_{m-1} \times M \to \{tt, ff\})$$

$$\tilde{A_S} = \tilde{A_{S_0}} \times \ldots \times \tilde{A_{S_{m-1}}} \quad \text{où} \quad \tilde{A_{S_i}} = (C_i \to (M \to \{tt, ff\}))$$

$$\alpha \in (A_S \to \tilde{A_S})$$

$$\alpha(I)_{i,c}(m) = (\exists c_0 \in C_0, \ldots, c_{i-1} \in C_{i-1}, c_{i+1} \in C_{i+1}, \ldots, c_{m-1} \in C_{m-1} \cdot I(\langle\langle c_0, \ldots, c_{i-1}, c, c_{i+1}, \ldots, c_{m-1}\rangle, m\rangle))$$

$$\gamma \in (\tilde{A_S} \to A_S)$$

$$\gamma(\tilde{I})(\langle\langle c_0, \ldots, c_{m-1}\rangle, m\rangle) = [\forall i \in m \cdot \tilde{I}_{i,c_i}(m)]$$

Exemple 4.3.2.4.3-1 Méthode de Owicki-Gries [76a]

Pour démontrer la correction partielle de :

$[\![11: X := X+1 ; 12: \| 21: X := X+2 ; 22:]\!]$

par la méthode de Owicki-Gries [76a], il faut vérifier les conditions suivantes :

$$\phi \Rightarrow [I_{11}(x) \wedge I_{21}(x)]$$

$$[I_{11}(x') \wedge x = x'+1] \Rightarrow I_{12}(x)$$

$$[I_{21}(x') \wedge x = x'+2] \Rightarrow I_{22}(x)$$

$$[\, I_{11}(x') \wedge I_{21}(x') \wedge x = x'+2 \,] \Rightarrow I_{11}(x)$$

$$[\, I_{12}(x') \wedge I_{31}(x') \wedge x = x'+2 \,] \Rightarrow I_{12}(x)$$

$$[\, I_{21}(x') \wedge I_{11}(x') \wedge x = x'+1 \,] \Rightarrow I_{21}(x)$$

$$[\, I_{22}(x') \wedge I_{11}(x') \wedge x = x'+1 \,] \Rightarrow I_{22}(x)$$

$$[\, I_{12}(x) \wedge I_{22}(x) \,] \Rightarrow \psi(x)$$

(Ces conditions s'obtiennent à partir du principe d'induction (-i) en utilisant 4.3.1.7-1, ce point sera illustré au paragraphe suivant).

□

Keller [76] et Owicki-Gries [76a] ont montré que la méthode de preuve correspondante n'est pas complète en utilisant un argument intuitif basé sur un exemple. Ce raisonnement peut être fait plus rigoureusement en observant que partant du principe d'induction (-i) qui s'écrit :

$$[\, \exists I \in A_S . (f(I) \Rightarrow I) \wedge (I \Rightarrow \psi) \,]$$

où

$$f(I)(s) = [\, \varepsilon(s) \vee (\exists s' \in S .\ I(s') \wedge t(s', s)) \,]$$

la méthode de preuve consiste à démontrer que

$$[\, \exists \tilde{I} \in \tilde{A}_S .\ \alpha \circ f \circ \gamma(\tilde{I}) \Rightarrow \tilde{I} \ \wedge\ \tilde{I} \Rightarrow \alpha(\psi) \,]$$

où

$\tilde{I} \Rightarrow \tilde{J}$ si et seulement si $\forall i \in n, c \in C_i, m \in M .\ \tilde{I}_{i,c}(m) \Rightarrow \tilde{J}_{i,c}(m)$

Posons $\tilde{f} = \alpha \circ f \circ \gamma$, c'est un opérateur monotone (pour $\Rightarrow$) sur le treillis complet $\tilde{A}_S$. D'après le théorème de Tarski (Tarski [55], Cousot-Cousot [79]) le plus fort invariant $\tilde{I}$ satisfaisant $\tilde{f}(\tilde{I}) \Rightarrow \tilde{I}$ est le plus petit point fixe $\mathrm{lfp}(\tilde{f})$ de $\tilde{f}$ (tel que $\tilde{f}(\mathrm{lfp}(\tilde{f})) = \mathrm{lfp}(\tilde{f})$ et si $\tilde{f}(\tilde{I}) \Rightarrow \tilde{I}$ alors $\mathrm{lfp}(\tilde{f}) \Rightarrow \tilde{I}$). Si on peut trouver un programme et un ψ tels que $\mathrm{lfp}(\tilde{f}) \not\Rightarrow \alpha(\psi)$, alors la méthode est incomplète. Il suffit de choisir le programme

0: X := 0; 1: [[11: X := X+1; 12: || 21: X := X+1; 22:]]

pour lequel $\mathit{fff}_{\mathcal{P}}(\tilde{f})$ est :

$I_0(x) = \underline{true}$

$I_1(x) = [x=0]$

$I_{11}(x) = [x \geq 0]$ $\quad I_{21}(x) = [x \geq 0]$

$I_{12}(x) = [x > 0]$ $\quad I_{22}(x) = [x > 0]$

qui ne permet pas de démontrer l'invariance de $\psi(x) = [0 \leq x \leq 2]$.

Remarquons que si $n=1$, la situation est celle de 4.3.2.4.2 et la méthode est sémantiquement complète.
Quand $n>1$, la méthode est incomplète car il n'est pas possible d'exprimer dans $\mathcal{AS}$ que certains états sont inaccessibles (dans le contre-exemple, on ne peut pas exprimer que lorsque l'exécution est en 11 avec $x=1$ dans le processus 1, alors c'est qu'elle est en 22 dans le processus 2). Pour ce faire, il faut pouvoir exprimer des relations entre les états de contrôle des divers processus. On peut utiliser à cet effet des compteurs ordinaux (comme dans Lamport [77]) ou des variables auxiliaires (comme dans Owicki-Gries [76a, 76b], Levin [79]).

4.3.2.4.4 Utilisation d'invariants sur l'état de contrôle et les variables associés à chaque point de contrôle du programme

Nous pouvons extraire de Newton [75] l'idée exprimée plus clairement dans Lamport [77] qui consiste à associer à chaque point de contrôle du programme une assertion portant sur l'état mémoire et l'état de contrôle du programme.
Nous avons donc :

$S = (C_0 \times \ldots \times C_{m-1}) \times M$

$A_S = (S \to \{tt, ff\})$

$\tilde{A_S} = \tilde{A_S}_0 \times \ldots \times \tilde{A_S}_{m-1}$ où $\tilde{A_S}_i = (C_i \to (C_0 \times \ldots \times C_{i-1} \times C_{i+1} \times \ldots \times C_{m-1} \times M \to \{tt, ff\}))$

$\alpha \in (A_S \to \tilde{A_S})$

$\alpha(I)_{i,c}(c_0, \ldots, c_{i-1}, c_{i+1}, \ldots, c_{m-1}, m) = I(\langle\langle c_0, \ldots, c_{i-1}, c, c_{i+1}, \ldots, c_{m-1}\rangle, m\rangle)$

$\gamma \in (\tilde{A_S} \to A_S)$

$\gamma(\tilde{I})(\langle\langle c_0, \ldots, c_{m-1}\rangle, m\rangle) = [\forall i \in m.\ \tilde{I}_{i,c_i}(c_0, \ldots, c_{i-1}, c_{i+1}, \ldots, c_{m-1}, m)]$

Comme (α, γ) est une corrspondance de Galois injective, nous pouvons construire une méthode de preuve correcte et sémantiquement complète comme en 4.3.1.7-1 et 4.3.1.8-2.

Exemple 4.3.2.4.4-1

Considérons l'application du principe d'induction (-i) aux programmes parallèles asynchrones de la forme :

$Ppa \equiv [\![Pra_0 \| \ldots \| Pra_{m-1}]\!]$

(cf. 2.8.2, les prélude et postlude étant vides et les affectations aléatoires et sections critiques étant exclues pour alléger la présentation) en utilisant la décomposition de l'invariant global en invariants locaux définie ci-dessus.

Le principe d'induction (-i) s'écrit :

$[\exists I \in A_S.\ \ \varepsilon \Rightarrow I \wedge F(I) \Rightarrow I \wedge I \Rightarrow \psi]$

où

$\varepsilon(\langle\langle c_0, \ldots, c_{m-1}\rangle, m\rangle) = [\forall i \in m.\ \exists \underline{L}_i \in \mathcal{L}.\ c_i = \underline{L}_i \wedge Pra_i \equiv \underline{L}_i : \beta]$

$F(I)(s) = [\exists s' \in S, i \in m.\ I(s') \wedge t[\![Ppa]\!]_i(s', s)]$

$t[\![Ppa]\!]_i(\langle\langle c'_0, \ldots, c'_{m-1}\rangle, m'\rangle, \langle\langle c_0, \ldots, c_{m-1}\rangle, m\rangle) =$

$[\forall j \in (m \setminus i).\ c'_j = c_j \wedge t[\![Pra_i]\!](\langle c'_i, m'\rangle, \langle c_i, m\rangle)]$

Comme en 4.3.1.7-1, posons $\underset{\sim}{F} = \alpha \circ F \circ \gamma$. Nous avons

$$\underset{\sim}{F}(\tilde{I})_{i,c_i}(c_0,\dots,c_{i-1},c_{i+1},\dots,c_{m-1},m)$$
$$= [\exists s \in S, k \in m.\ \gamma(\tilde{I})(s') \wedge t[\![Ppa]\!]_k(s', \ll c_0,\dots,c_{m-1}\gg, m\gg)]$$
$$[\exists c'_0 \in C_0,\dots,c'_{m-1} \in C_{m-1}, m' \in M, k \in m.\quad \forall j \in m.\ \tilde{I}_{j,c'_j}(c'_0,\dots,c'_{j-1},c'_{j+1},\dots,c'_{m-1},m)$$
$$\wedge\ \forall j \in (m \vee k).\ c'_j = c_j \quad \wedge\ t[\![Pra_k]\!](\langle c'_k, m'\rangle, \langle c_k, m\rangle)]$$
$$[\exists c'_i \in C_i,\ m' \in M.\ \tilde{I}_{i,c'_i}(c_0,\dots,c_{i-1},c_{i+1},\dots,c_{m-1},m')$$
$$\wedge\ \underline{\text{contexte}}\{i\}(\tilde{I})(c_0,\dots,c_{i-1},c'_i,c_{i+1},\dots,c_{m-1},m') \wedge t[\![Pra_i]\!](\langle c'_i,m'\rangle,\langle c_i,m\rangle)]$$
$$[\exists k \in (m \vee i),\ c'_k \in C_k,\ m' \in M.\ \tilde{I}_{i,c_i}(c_0,\dots,c_{i-1},c_{i+1},\dots,c_{k-1},c'_k,c_{k+1},\dots,c_{m-1},m')$$
$$\wedge\ \tilde{I}_{k,c'_k}(c_0,\dots,c_{k-1},c_{k+1},\dots,c_{m-1},m') \wedge \underline{\text{contexte}}\{i,k\}(\tilde{I})(c_0,\dots,c_{k-1},c'_k,c_{k+1},\dots,c_{m-1},m')$$
$$\wedge\ t[\![Pra_k]\!](\langle c'_k,m'\rangle,\langle c_k,m\rangle)]$$

où

$$\underline{\text{contexte}}\ E(\tilde{I})(c_0,\dots,c_{m-1},m) = \forall j \in (m \vee E).\ \tilde{I}_{j,c_j}(c_0,\dots,c_{j-1},c_{j+1},\dots,c_{m-1},m)$$

Le premier terme correspond à la preuve séquentielle (du processus i), le deuxième terme correspond à la preuve d'absence d'interférences (entre i et tout $k \neq i$), le terme de contexte apportant l'information disponible dans les autres processus n'apparaissant pas dans Lamport [77].

Remarquons que le terme correspondant à l'absence d'interférences disparaît quand $m=1$ et que dans ce cas les conditions de vérification correspondent exactement dans tous les cas à celle de la méthode de Floyd [67].

Nous illustrons la condition de vérification :

$\forall i \in m,\ c_i \in C_i$.

$$\underset{\sim}{F}(\tilde{I})_{i,c_i}(c_0,\dots,c_{i-1},c_{i+1},\dots,c_{m-1},m) \Rightarrow \tilde{I}_{i,c_i}(c_0,\dots,c_{i-1},c_{i+1},\dots,c_{m-1},m)$$

sur l'exemple suivant :

$[\![$

11: $\{\tilde{I}_{11}(c_2, x)\}$

$X := X+1;$

12: $\{\tilde{I}_{12}(c_2, x)\}$

$\|$

21: $\{\tilde{I}_{21}(c_1, \ )\}$

$X := X+2;$

22: $\{\tilde{I}_{22}(c_1, x)\}$

$]\!]$

ce qui donne :

- Preuve séquentielle :

$$[\tilde{I}_{11}(c_2, x') \wedge (c_2 = 21 \Rightarrow \tilde{I}_{21}(11, x')) \wedge (c_2 = 22 \Rightarrow \tilde{I}_{22}(11, x')) \wedge x = x'+1] \Rightarrow \tilde{I}_{12}(c_2, x)$$

$$[\tilde{I}_{21}(c_1, x') \wedge (c_1 = 11 \Rightarrow \tilde{I}_{11}(21, x')) \wedge (c_1 = 12 \Rightarrow \tilde{I}_{12}(21, x')) \wedge x = x'+2] \Rightarrow \tilde{I}_{22}(c_1, x)$$

- Absence d'interférences :

$$[\tilde{I}_{11}(21, x') \wedge \tilde{I}_{21}(11, x') \wedge x = x'+2] \Rightarrow \tilde{I}_{11}(22, x)$$

$$[\tilde{I}_{12}(21, x') \wedge \tilde{I}_{21}(12, x') \wedge x = x'+2] \Rightarrow \tilde{I}_{12}(22, x)$$

$$[\tilde{I}_{21}(11, x') \wedge \tilde{I}_{11}(21, x') \wedge x = x'+1] \Rightarrow \tilde{I}_{21}(12, x)$$

$$[\tilde{I}_{22}(11, x') \wedge \tilde{I}_{11}(22, x') \wedge x = x'+1] \Rightarrow \tilde{I}_{22}(12, x)$$

Nous définissons maintenant $\tilde{F} \in (A\tilde{s} \to A\tilde{s})$ par :

$$\tilde{F}(\tilde{I})_{i,c_i}(c_0, \dots, c_{i-1}, c_{i+1}, \dots, c_{m-1}, m)$$
$$= [\exists c'_i \in C_i, m' \in M.\ \tilde{I}_{i,c_i}(c_0, \dots, c_{i-1}, c_{i+1}, \dots, c_{m-1}, m') \wedge$$
$$t[\![Pra_i]\!](\langle c'_i, m'\rangle, \langle c_i, m\rangle)$$
$$\vee\ [\exists k \in (m \vee i),\ c'_k \in C_k,\ m' \in M.\ \tilde{I}_{k,c'_k}(c_0, \dots, c_{k-1}, c_{k+1}, \dots, c_{m-1}, m') \wedge$$
$$t[\![Pra_k]\!](\langle c'_k, m'\rangle, \langle c_k, m\rangle)\ ]$$

Nous observons alors que $\alpha \circ F = \tilde{F} \circ \alpha$. Les conditions de vérification correspondant à $\tilde{F}(\tilde{I}) \Rightarrow \tilde{I}$ sont similaires à celles introduites par Newton [75] (bien que la similitude soit difficile à saisir car Newton utilise une définition des programmes parallèles un peu singulière).

Sur notre exemple, nous obtenons :

- Preuve séquentielle :

$$[\tilde{I}_{11}(c_2, x') \wedge x = x'+1] \Rightarrow \tilde{I}_{12}(c_2, x)$$

$$[\tilde{I}_{21}(c_1, x') \wedge x = x'+2] \Rightarrow \tilde{I}_{22}(c_1, x)$$

Absence d'interférences :

$$[\tilde{I}_{21}(11, x') \wedge x = x'+2] \Rightarrow \tilde{I}_{11}(22, x)$$

$$[\tilde{I}_{21}(12, x') \wedge x = x'+2] \Rightarrow \tilde{I}_{12}(22, x)$$

$$[\tilde{I}_{11}(21, x') \wedge x = x'+1] \Rightarrow \tilde{I}_{21}(12, x)$$

$$[\tilde{I}_{11}(22, x') \wedge x = x'+1] \Rightarrow \tilde{I}_{22}(12, x)$$

Comme $\underset{\sim}{F} \neq \tilde{F}$, nous obtenons un treillis complet de conditions de vérification $\bar{F}(\check{I}) \Rightarrow \check{I}$ où $\underset{\sim}{F} \Rightarrow \bar{F} \Rightarrow \tilde{F}$ correspondant chacune à une méthode de preuve particulière. Par exemple, la méthode de Lamport [77] (aussi bien que celle de Owicki-Gries [76a] si on avait utilisé la définition de α et γ donnée en 4.3.2.4.3) correspond à :

$$\begin{aligned}
&\bar{F}(\check{I})_{i,c_i}(c_0, \dots, c_{i-1}, c_{i+1}, \dots, c_{n-1}, m) \\
&= [\exists c'_i \in C_i,\ m' \in M.\ \tilde{I}_{i,c'_i}(c_0, \dots, c_{i-1}, c_{i+1}, \dots, c_{n-1}, m') \wedge t[\![Pre_i]\!](\langle c'_i, m'\rangle, \langle c_i, m\rangle)] \\
&\quad \vee [\exists k \in (n \vee i),\ c'_k \in C_k,\ m' \in M.\ \tilde{I}_{i,c_i}(c_0, \dots, c_{i-1}, c_{i+1}, \dots, c_{k-1}, c'_k, c_{k+1}, \dots, c_{n-1}, m') \wedge \\
&\qquad \tilde{I}_{k,c'_k}(c_0, \dots, c_{k-1}, c_{k+1}, \dots, c_{n-1}, m') \wedge t[\![Pre_k]\!](\langle c'_k, m'\rangle, \langle c_k, m\rangle)]
\end{aligned}$$

D'après 4.3.1.7-1 la méthode de Lamport [77] est donc correcte et d'après 4.3.1.8-2, elle est sémantiquement complète.

Pour notre exemple, nous obtenons :

- Preuve séquentielle :

$$[\tilde{I}_{11}(c_2, x') \wedge x = x'+1] \Rightarrow \tilde{I}_{12}(c_2, x)$$

$$[\tilde{I}_{21}(c_1, x') \wedge x = x'+2] \Rightarrow \tilde{I}_{22}(c_1, x)$$

- Absence d'interférences

$$[\tilde{I}_{11}(21,x') \wedge \tilde{I}_{21}(11,x') \wedge x=x'+2] \Rightarrow \tilde{I}_{11}(22,x)$$
$$[\tilde{I}_{12}(21,x') \wedge \tilde{I}_{21}(12,x') \wedge x=x'+2] \Rightarrow \tilde{I}_{12}(22,x)$$
$$[\tilde{I}_{21}(11,x') \wedge \tilde{I}_{11}(21,x') \wedge x=x'+1] \Rightarrow \tilde{I}_{21}(12,x)$$
$$[\tilde{I}_{22}(11,x') \wedge \tilde{I}_{11}(22,x') \wedge x=x'+1] \Rightarrow \tilde{I}_{22}(12,x)$$

Observons que bien que toutes les méthodes dérivées d'un F tel que $\underset{\sim}{F} \Rightarrow F \Rightarrow \tilde{F}$ soient sémantiquement complètes, il se peut que pour certains programmes, on puisse montrer que des assertions sont invariantes en utilisant $\underset{\sim}{F}$ et qu'on ne puisse pas le faire en utilisant $\tilde{F}$. C'est le cas dans notre exemple pour :

$\tilde{I}_{11}(c_2,x) = [\text{pair}(x)]$ $\qquad$ $\tilde{I}_{21}(c_1,x) = [x=0 \vee x=1]$

$\tilde{I}_{12}(c_2,x) = [x=1 \vee x=3]$ $\qquad$ $\tilde{I}_{22}(c_1,x) = [x=2 \vee x=3]$

Toutefois, on peut toujours renforcer l'invariant et remplacer

$$\tilde{I}_{i,c_i}(c_0,\dots,c_{i-1},c_{i+1},\dots,c_{n-1},m)$$

par

$$\tilde{I}_{i,c_i}(c_0,\dots,c_{i-1},c_{i+1},\dots,c_{n-1},m) \wedge \underline{\text{contexte}}\{i\}(\tilde{I})(c_0,\dots,c_{n-1},m)$$

donc toutes les conditions de vérification $F(\tilde{I}) \Rightarrow \tilde{I}$ sont fortement équivalentes.

□

Exemple 4.3.2.4.4-2

Nous démontrons la correction partielle du programme 2.8.2.3, mais au lieu d'utiliser le principe d'induction (-i), nous utilisons (-4-), de manière à pouvoir relier la valeur courante n de la variable N à sa valeur initiale $\underline{n}$.

Puisque les deux processus sont symétriques, nous n'avons besoin de raisonner que sur le processus 1. Nous allons montrer que la relation :

$$Inv(\underline{n}, c_2, m, p_1, p_2) = [(c_2 = \ell 1 \wedge p_1 = 2^{\underline{n}-m}) \vee (c_2 \neq \ell 1 \wedge p_1 \times p_2 = 2^{\underline{n}-m})]$$

est invariante dans le processus 1 après initialisation de la variable P1. Puisque la corection partielle découle de l'invariant quand $c_2 = \ell 5$ (le processus 2 a terminé) et $0 \leq m \leq 1$, nous allons aussi

valeur de N après l'exécution de la commande

et 0 ou 1. Puisque la valeur initiale $\underline{n}$ de N est

, la seule difficulté est quand $\underline{n} > 1$. Dans ce cas N est decrementé jusqu'à atteindre la valeur 2. D'une part, chacun des deux processus peut tester que $N > 1$ avant qu'il ne soit decrémenté par l'autre, le decrémenter et terminer. Dans ce cas N vaudra 0 après l'exécution de la commande parallèle. D'autre part, un processus peut tester si $N > 1$ et le decrémenter avant que l'autre ne teste si $N > 1$. Alors les deux processus terminent et $N = 1$ après exécution de la commande parallèle. Pour une preuve d'invariance, ces arguments opeiationnels peuvent s'exprimer "indépendemment du temps", d'où les invariants locaux suivants :

$$\tilde{I}_{11}(\underline{n}, c_2, m, p_1, p_2) = \tilde{I}_{12}(\underline{n}, c_2, m, p_1, p_2)$$

$$\tilde{I}_{12}(\underline{n}, c_2, m, p_1, p_2) = [Inv(\underline{n}, c_2, m, p_1, p_2) \wedge [(c_2 \in \{\ell 1, \ell 2\} \wedge m = \underline{n} \wedge m > 0) \vee (c_2 = \ell 3 \wedge m > 1) \vee (c_2 = \ell 4 \wedge m \geq 1) \vee (c_2 = \ell 5 \wedge 0 \leq m \leq 1)]]$$

$$\tilde{I}_{13}(\underline{n}, c_2, m, p_1, p_2) = [Inv(\underline{n}, c_2, m, p_1, p_2) \wedge [(c_2 \in \{\ell 1, \ell 2, \ell 3\} \wedge m > 1) \vee (c_2 = \ell 4 \wedge m > 1) \vee (c_2 = \ell 5 \wedge m = 1)]]$$

$$\tilde{I}_{14}(\underline{n}, c_2, m, p_1, p_2) = [Inv(\underline{n}, c_2, m, p_1, p_2) \wedge [(c_2 \in \{\ell 1, \ell 2, \ell 3\} \wedge m > 0) \vee (c_2 = \ell 4 \wedge m \geq 0) \vee (c_2 = \ell 5 \wedge 0 \leq m \leq 1)]]$$

$$\tilde{I}_{15}(\underline{n}, c_2, m, p_1, p_2) = [Inv(\underline{n}, c_2, m, p_1, p_2) \wedge [(c_2 = \ell 3 \wedge m = 1) \vee (c_2 \neq \ell 3 \wedge 0 \leq m \leq 1)]]$$

$$\tilde{I}_1(\underline{n}, m, p_1, p_2) = [p_1 \times p_2 = 2^{\underline{n}-m} \wedge 0 \leq m \leq 1]$$

$$\tilde{I}_2(\underline{n}, p) = [p = 2^{\underline{n}}]$$

C'est simple de montrer que ces invariants locaux satisfont les conditions de vérification suivantes :

- Initialisation :

$$[\underline{m} \geq 0] \Rightarrow [\tilde{I}_{11}(\underline{m}, 21, \underline{m}, p_1, p_2) \wedge \tilde{I}_{21}(\underline{m}, 11, \underline{m}, p_2, p_1)]$$

- Preuve séquentielle :

$$\tilde{I}_{11}(\underline{m}, c_2, m, p_1', p_2) \Rightarrow \tilde{I}_{12}(\underline{m}, c_2, m, 1, p_2)$$

$$[\tilde{I}_{12}(\underline{m}, c_2, m, p_1, p_2) \wedge m > 1] \Rightarrow \tilde{I}_{13}(\underline{m}, c_2, m, p_1, p_2)$$

$$[\tilde{I}_{12}(\underline{m}, c_2, m, p_1, p_2) \wedge m \leq 1] \Rightarrow \tilde{I}_{15}(\underline{m}, c_2, m, p_1, p_2)$$

$$\tilde{I}_{13}(\underline{m}, c_2, m', p_1', p_2) \Rightarrow \tilde{I}_{14}(\underline{m}, c_2, m'-1, 2p_1', p_2)$$

$$[\tilde{I}_{14}(\underline{m}, c_2, m, p_1, p_2) \wedge m > 1] \Rightarrow \tilde{I}_{13}(\underline{m}, c_2, m, p_1, p_2)$$

$$[\tilde{I}_{14}(\underline{m}, c_2, m, p_1, p_2) \wedge m \leq 1] \Rightarrow \tilde{I}_{15}(\underline{m}, c_2, m, p_1, p_2)$$

- Preuve d'absence d'interférences :

Pour $k = 1, \ldots, 5$

$$[\tilde{I}_{1k}(\underline{m}, 21, m, p_1, p_2') \wedge \tilde{I}_{21}(\underline{m}, 1k, m, p_1, p_2')] \Rightarrow \tilde{I}_{1k}(\underline{m}, 22, m, p_1, 1)$$

$$[\tilde{I}_{1k}(\underline{m}, 22, m, p_1, p_2) \wedge \tilde{I}_{22}(\underline{m}, 1k, m, p_1, p_2) \wedge m > 1] \Rightarrow \tilde{I}_{1k}(\underline{m}, 23, m, p_1, p_2)$$

$$[\tilde{I}_{1k}(\underline{m}, 22, m, p_1, p_2) \wedge \tilde{I}_{22}(\underline{m}, 1k, m, p_1, p_2') \wedge m \leq 1] \Rightarrow \tilde{I}_{1k}(\underline{m}, 25, m, p_1, p_2)$$

$$[\tilde{I}_{1k}(\underline{m}, 23, m', p_1, p_2') \wedge \tilde{I}_{23}(\underline{m}, 1k, m', p_1, p_2')] \Rightarrow \tilde{I}_{1k}(\underline{m}, 24, m'-1, p_1, 2 \times p_2')$$

$$[\tilde{I}_{1k}(\underline{m}, 24, m, p_1, p_2) \wedge \tilde{I}_{24}(\underline{m}, 1k, m, p_1, p_2) \wedge m > 1] \Rightarrow \tilde{I}_{1k}(\underline{m}, 23, m, p_1, p_2)$$

$$[\tilde{I}_{1k}(\underline{m}, 24, m, p_1, p_2) \wedge \tilde{I}_{24}(\underline{m}, 1k, m, p_1, p_2) \wedge m \leq 1] \Rightarrow \tilde{I}_{1k}(\underline{m}, 25, m, p_1, p_2)$$

- Finalisation :

$$[\tilde{I}_{15}(\underline{m}, 25, m, p_1, p_2) \wedge \tilde{I}_{25}(\underline{m}, 15, m, p_2, p_1)] \Rightarrow \tilde{I}_1(\underline{m}, m, p_1, p_2)$$

$$[\tilde{I}_1(\underline{m}, m, p_1, p_2) \wedge m = 0] \Rightarrow \tilde{I}_2(\underline{m}, p_1 \times p_2)$$

$$[\tilde{I}_1(\underline{m}, m, p_1, p_2) \wedge m \neq 0] \Rightarrow \tilde{I}_2(\underline{m}, 2 \times p_1 \times p_2)$$

$$\tilde{I}_2(\underline{m}, p) \Rightarrow [p = 2^{\underline{m}}]$$

Noter que pour tout le programme, nous avons obtenu 77 conditions de vérification, ce qui est évidemment énorme mais peut être réduit considérablement en choisissant une décomposition moins fine (cf. 4.3.2.4.6 - 3).

□

4.3.2.4.5 Utilisation d'invariants sur les variables et des variables auxiliaires associés à chaque point de contrôle du programme

Après s'être rendus compte que l'utilisation d'invariants sur les variables associés à chaque point de contrôle du programme (comme en 4.3.2.4.3 qui leur semblait la généralisation la plus naturelle de la méthode de Floyd [67] au cas des programmes parallèles) était incomplète, Owicki-Gries [76a] ont introduit la technique des variables auxiliaires. Nous montrons que ces variables auxiliaires ne sont qu'un moyen de raisonner implicitement sur les états de contrôle du programme.

Pour présenter la technique, nous considérons un programme asynchrone $Ppa \equiv [\![Pra_0 \| \ldots \| Pra_{m-1}]\!]$ dont, pour simplifier, le prélude et le postlude sont supposés vides. L'ensemble des états du programme est donc $S = (C_0 \times \ldots \times C_{m-1}) \times M$ où C_i est l'ensemble des points de contrôle du processus Pra_i et M l'ensemble des états mémoire. Notons L_i le point d'entrée du processus Pra_i (de sorte que $Pra_i \equiv L_i : \alpha$) et $C = \bigcup_{i \in m} C_i$ l'ensemble des points de contrôle du programme.

En chaque point $L \in C$ du programme Ppa est associé un commentaire qui est une relation $\psi_L \in (M^2 \to \{tt, ff\})$ ne portant que sur les états mémoire et qui s'interprète comme signifiant que :

$\Psi \in (S \times S \to \{tt, ff\})$

$\Psi(\langle\langle L_0, \ldots, L_{m-1}\rangle, m\rangle, \langle\langle L'_0, \ldots, L'_{m-1}\rangle, m'\rangle) = [\bigwedge_{i=0}^{m-1} (L_i = L_i \wedge \psi_{L'_i}(m, m'))]$

est invariante. Remarquons que Ψ permet de décrire les états de contrôle du programme mais sous une forme extrêmement limitée (puisque par exemple, il est possible d'exprimer qu'un point L du programme est inaccessible en choisissant $\psi_L(m, m) = false$ mais il n'est pas possible d'exprimer de relations entre les états de contrôle des

différents processus du programme, comme par exemple que les points L_i et L_j sont mutuellement exclusifs).

Pour démontrer l'invariance de Ψ pour Ppa, la technique de Owicki-Gries [76a] consiste à considérer un programme transformé Ppa' de la forme $[\![Pra'_0 \| \ldots \| Pra'_{m-1}]\!]$ où chaque processus Pra'_i a essentiellement le même comportement que Pra_i mais contient des commandes d'affectation à des variables auxiliaires n'apparaissant pas dans Ppa et des invariants ψ'_L (portant sur les variables de Ppa mais également sur les variables auxiliaires) qui impliquent les ψ_L.

Plus précisement, l'ensemble VA des variables auxiliaires du programme Ppa' est le plus grand ensemble de variables qui n'apparaissent qu'en membres gauches d'affectations ou en membres droits d'affectations à des variables auxiliaires.

Ppa s'obtient à partir de Ppa' en supprimant les affectations aux variables auxiliaires c'est-à-dire en répétant un nombre fini de fois la transformation $\alpha L : V := E ; \beta \to \alpha\beta$ où $V \in VA$ (puis en effectuant s'il y a lieu certaines simplifications évidentes comme par exemple, $1 : \not\downarrow 2 : V := E ; 3 : \not\downarrow ; 4 :$ est équivalent à $1 : V := E ; 4 :$ ou $1 : \underline{skip} ; 2 : \underline{skip} ; 3 :$ est équivalent à $1 : \underline{skip} ; 3 :$).

L'ensemble des états mémoire de Ppa' peut donc être compris (à un isomorphisme près) comme étant $M \times M'$ où $M' = (VA \to \mathcal{D})$. Comme précédemment, notons C'_i l'ensemble des points de contrôle du processus Pra'_i, $i \in m$, L'_i le point d'entrée du processus Pra'_i et $C' = \bigcup_{i \in m} C'_i$.

En chaque point $L' \in C'$ du programme Ppa' est associé un commentaire qui est une relation $\psi'_L \in ((M \times M')^2 \to \{tt, ff\})$ ne portant que sur les variables de Ppa et les variables auxiliaires et qui s'interprète comme précédemment par l'invariance de Ψ' telle que :

$$\Psi'(\langle L_0, \ldots, L_{m-1} \rangle, m), \langle L'_0, \ldots, L'_{m-1} \rangle, m') = [\bigwedge_{i=0}^{m-1} (L_i = L'_i \wedge \psi'_{L'_i}(m, m'))]$$

Les Ψ'_L doivent être choisis de sorte que pour tous $\underline{m}, \underline{m}', m, m' \in \mathcal{D}$, on ait :

$$\Psi'_L(\langle \underline{m}, \underline{m}' \rangle, \langle m, m' \rangle) \Rightarrow \Psi_L(\underline{m}, m)$$

4.3.2.4.5.1 Correction de la méthode

Observons que la sémantique du programme Ppa s'obtient à partir de la sémantique du programme auxiliaire Ppa' par élimination des états inobservables (appartenant à $((C'_0 \times \ldots \times C'_{m-1}) \vee (C_0 \times \ldots \times C_{n-1})) \times M \times M'$) puis par correspondance à une fonction $f_s \in ((C_0 \times \ldots \times C_{m-1}) \times M \times M' \to (C_0 \times \ldots \times C_{m-1}) \times M)$ entre états près (définie par $f_s(\langle\langle c_0, \ldots, c_{m-1}\rangle, m, m'\rangle) = \langle\langle c_0, \ldots, c_{m-1}\rangle, m\rangle$).
La preuve formelle, simple mais fastidieuse, repose sur le lemme 2.5.4v1.2. De plus, on a $\Psi'(\underline{s}, s) \Rightarrow \Psi(f_s(\underline{s}), f_s(s))$ quand $s \in ((C_0 \times \ldots \times C_{m-1}) \times M \times M')$.

Ceci permet de conclure que l'utilisation de variables auxiliaires est correcte dans la mesure où une propriété invariante pour le programme auxiliaire l'est également pour le programme original obtenu par élimination des variables auxiliaires. C'est la conséquence du théorème 4.1.1v5 et du théorème 4.1.2v1 (qui s'applique au cas où les états initiaux des programmes Ppa et Ppa' sont identiques et se généralise aisément).

Remarquons que, d'après le théorème 4.1.1v5, la méthode de Owicki-Gries se généralise de manière évidente aux preuves d'invariance conditionnelle (dans la mesure où en pratique ϕ ne porterait pas sur les variables auxiliaires).

De la même façon, on constate que la limitation de l'usage des variables auxiliaires aux affectations dans méthode de Owicki-Gries est inutile. Par exemple, les variables auxiliaris peuvent être utilisées dans les tests dans la mesure où ces modifications laissent le flôt de contrôle du programme globalement inchangé.

4.3.2.4.5.2 Complétude sémantique de la méthode

Pour démontrer la correction partielle de l'exemple :

0: X:=0; 1: [[11: X:=X+1; 12: || 21: X:=X+1; 22:]]; 2:

(utilisé pour démontrer que la méthode 4.3.2.4.3 est incomplète), on peut transformer le programme comme suit :

```
0:  ⟨ 01: X:=0;   02: v1:=11 ;   03: v2:=21;   04: ⟩ ;
1:  [[  11:
            ⟨ 111: X:=X+1;   112: v1:=12;   113: ⟩ ;
        12:
    ||
        21:
            ⟨ 211: X:=X+1;   212: v2:=22;   213: ⟩ ;
        22:
    ]];
2:
```

et utiliser les invariants :

$\psi'_1(x, v1, v2) = [v1=11 \wedge v2=21 \wedge x=0]$

$\psi'_{11}(x, v1, v2) = [v1=11 \wedge ((v2=21 \wedge x=0) \vee (v2=22 \wedge x=1))]$

$\psi_{12}(x, v1, v2) = [v1=12 \wedge ((v2=21 \wedge x=1) \vee (v2=22 \wedge x=2))]$

$\psi'_{21}(x, v1, v2) = [v2=21 \wedge ((v1=11 \wedge x=0) \vee (v1=12 \wedge x=1))]$

$\psi_{22}(x, v1, v2) = [v2=22 \wedge ((v1=11 \wedge x=1) \vee (v1=12 \wedge x=2))]$

$\psi'_2(x, v1, v2) = [x=2]$

qui sont ceux qu'on aurait utilisé avec la méthode (à la différence près que les variables v1 et v2 sont utilisées pour simuler les compteurs ordinaux des deux processus).

De manière plus générale, considérons un programme parallèle asynchrone Ppa de la forme $[\![Pra_0 \| \ldots \| Pra_{m-1}]\!]$ et des ψ_L, $L \in C$ tels que Ψ, défini comme précédemment soit invariant. Il faut montrer qu'on peut trouver Ψ' dont on peut démontrer l'invariance pour un programme transformé Ppa' sans se référer aux compteurs ordinaux du programme Ppa'.

Construisons Ppa' en simulant le compteur ordinal de chaque processus Pra_i au moyen d'une variable auxiliaire V'_i (n'apparaissant pas dans le programme Ppa). Pour cela nous supposerons qu'il existe un ensemble de valeurs de cette variable qui, au moyen d'un codage c bijectif, est équipotent à l'ensemble des étiquettes apparaissant dans le processus Pra_i et nous supposerons également qu'aux étiquettes L de Ppa correspondent des étiquettes $L', L'_1, L'_2, \ldots$ n'apparaissant pas dans Ppa et ces étiquettes étant toutes deux à deux distinctes. Ppa' s'obtient à partir de Ppa par la transformation τ suivante :

. $\tau(\underline{L}: [\![Pra_0 \| \ldots \| Pra_{m-1}]\!]; \bar{L}) =$

$\underline{L}: V_0 := \underline{L}_0;\ \underline{L}'_1: V_1 := \underline{L}_1; \ldots;\ \underline{L}'_{m-1}: V_{m-1} := \underline{L}_{m-1};\ \underline{L}'_m:$

$[\![\tau_0(Pra_0) \| \ldots \| \tau_{m-1}(Pra_{m-1})]\!]; \bar{L}:$

. $\tau_i(L:) = L:$

. $\tau_i(L1: \mathcal{C}a;\ L2:\alpha) =$

$L1: \not< L1'1: \mathcal{C}a;\ L1'2: V_i := c(L2);\ L1'3: >;\ \tau_i(L2:\alpha)$

quand $\mathcal{C}a$ est skip, $V_i := E$, $V_i := ?$ ou $\not< Ps >$

. $\tau_i(L1:$ if B then $L2:\alpha;\ L3:$ else $L4:\beta;\ L5:$ fi; $L6:\gamma)$

$L1: \not< L1'1:$ if B then

$L1'2: T_i :=$ true; $L1'3: V_i := c(L2);\ L1'3:$

else

$L1'5: T_i :=$ false; $L1'6: V_i := c(L3);\ L1'4:$

fi; $L1'8: >;$

$L1'':$ if T_i then $\tau_i(L2:\alpha; L3:)\ V_i := c(L6);\ L3''':$

else $\tau_i(L4:\beta; 15:)\ V_i := c(L6);\ L5''':$

fi;

$\tau(L6:\gamma)$

τ_i(L1: **while** B **do** L2: α; L3: **od**; L4: β) =

L1: $\not$ L1'1: **if** B **then**

L1'2: T_i := **true**; L1'3: V_i := c(L2); L1'4:

else

L1'5: T_i := **false**; L1'6: V_i := c(L4); L1'7:

fi; L1'8: $\not$;

L1": **while** T_i **do**

τ_i (L2: α ; L3:)

$\not$ L3'1: **if** B **then**

L3'2: T_i := **true**; L3'3: V_i := c(L2); L3'4:

else

L3'5: T_i := **false**; L3'6: V_i := c(L4); L3'7:

fi; L3'8: $\not$;

L3":

od;

τ_i(L4: β)

Définissons maintenant ψ' comme caractérisant exactement les descendants des états d'entrée de Ppa (au codage des états de contrôle de Ppa par des valeurs des variables auxiliaires de Ppa' près).

Pour tous états $\underline{s}$, s de Ppa, définissons :

$$\Psi''(\underline{s}, s) = [\, \varepsilon(\underline{s}) \wedge (\bigvee_{i \in m} t[\![Pra_i]\!])^*(\underline{s}, s)\,]$$

avec

$$\varepsilon(\langle\langle L_0, \ldots, L_{m-1}\rangle, M\rangle) = (\bigwedge_{i \in m} L_i = \underline{L}_i)$$

de sorte que Ψ'' est invariante pour Ppa. Posons alors :

- Pour toutes les étiquettes L qui figurent dans Ppa et Ppa', remarquons que la transformation de Ppa en Ppa' est telle que lorsque le contrôle est au point L du ième processus de Ppa', on a $v_i = c(L)$. On choisit donc :

$$\Psi'_L(\langle \underline{M}, \underline{M}' \rangle, \langle M, M' \rangle) =$$
$$\Psi''(\langle\langle c^{-1}(\underline{M}'(V_0)), \ldots, c^{-1}(\underline{M}'(V_{m-1}))\rangle, \underline{M}\rangle,$$
$$\langle\langle c^{-1}(M'(V_0)), \ldots, c^{-1}(M'(V_{m-1}))\rangle, M\rangle)$$

de sorte que l'anertion en L dans Ppa' est le plus fort invariant de Ppa en L obtenu en transcrivant le contrôle en terme de variables auxiliaires.

- Pour toutes les étiquettes notées Li'j qui figurent dans une section critique de Ppa' mais pas dans Ppa, on utilise des assertions intermédiaires liant les valeurs courantes des variables (y compris auxiliaires) aux valeurs (symboliques) de ces mêmes variables au début de la section critique, ce qui permet de traiter les sections critiques comme une action atomique (cf. 4.3.2.2.2.4).

- Les autres étiquettes figurant dans Ppa' mais pas dans Ppa ont été introduites à cause des tests (dans une commande alternative ou itérative). Pour les étiquettes notées L'' qu'on trouve sous la forme

... L: ♯ L'1 : if B then

L'2 : T_i := true ; L'3 : V_i := c(L2) ; L'4 :

else

L'5 : T_i := false ; L'6 : V_i := c(L4) ; L'7 :

fi ; L'8 : ♯ ;

L'' : ...

dans Ppa', on choisit :

$$\Psi'_{L''}(\langle \underline{M}, \underline{M}' \rangle, \langle M, M' \rangle) = [M \in \underbrace{dom}(\mathbb{B}[\![B]\!]) \wedge$$
$$[\mathbb{B}[\![B]\!](M) \wedge M'(T_i) \wedge M'(V_i) = c(L2) \wedge \Psi'_{L2}(\langle \underline{M}, \underline{M}' \rangle, \langle M, M' \rangle)]$$
$$\vee$$
$$[\neg\mathbb{B}[\![B]\!](M) \wedge \neg M'(T_i) \wedge M'(V_i) = c(L4) \wedge \Psi'_{L4}(\langle \underline{M}, \underline{M}' \rangle, \langle M, M' \rangle)]]$$

- Pour les étiquettes notées L''' qu'on trouve sous la forme

... L1 :

$V_i := c(L2)$;

L''' : ...

dans Ppa et qui précèdent un <u>else</u> ou un <u>fi</u>, on choisira

$$\psi'_{L'''}(\langle \underline{M}, \underline{M}' \rangle, \langle M, M' \rangle) =$$
$$[\psi'_{L1}(\langle \underline{M}, \underline{M}' \rangle, \langle M, M' \rangle) \wedge M'(V_i) = c(L2)]$$

On vérifie alors aisément (bien que les calculs soient très fastidieux) que les conditions de vérification correspondant à l'application de la méthode 4.3.2.4.3 pour ψ' et Ppa' sont satisfaites (car ψ'' satisfait les conditions de vérification de la méthode 4.3.2.4.4 pour Ppa) et que

$$\psi'_L(\langle \underline{M}, \underline{M}' \rangle, \langle M, M' \rangle) \Rightarrow \psi_L(\underline{M}, M)$$

pour toutes les étiquettes figurant dans Ppa et Ppa'.

4.3.2.4.6 Utilisation d'invariants sur l'état de contrôle et les variables associés à chaque processus du programme

Si on utilise les décompositions 4.3.2.4.3 ou 4.3.2.4.4 pour un programme $[\![Pra_0 \| \dots \| Pra_{m-1}]\!]$ où chaque processus Pra_i, $i \in m$ a m_i points de contrôle, il y a $\sum_{i \in m} (m_i - 1)$ conditions de vérifications correspondant à la preuve séquentielle et $\sum_{i \in m} \sum_{j \in m_i} \sum_{k \in (m \setminus i)} (m_k - 1) = \sum_{i \neq j} m_i (m_j - 1)$ conditions de vérification correspondant à la preuve d'absence d'interférences.

Pour éviter la prolifération des conditions de vérification correspondant à la preuve d'absence d'interférences, on peut choisir une décomposition moins fine, comme celle proposée par Lamport [80] qui consiste à associer un invariant global à chaque processus du programme.

Exemple 4.3.2.4.6-1

Pour démontrer la correction partielle du programme

$[\![11 : X := X+1 ; 12 : \| \; 21 : X := X+2 ; 22 :]\!]$

il faut vérifier les conditions suivantes :

$$\Phi(x) \Rightarrow [I_1(11, 21, x) \wedge I_2(11, 21, x)]$$

$$[I_1(11, c_2, x') \wedge x = x'+1] \Rightarrow I_1(12, c_2, x)$$

$$[I_2(c_1, 21, x') \wedge x = x'+2] \Rightarrow I_2(c_1, 22, x)$$

$$[I_1(c_1, 21, x') \wedge x = x'+2] \Rightarrow I_1(c_1, 22, x)$$

$$[I_2(11, c_2, x') \wedge x = x'+1] \Rightarrow I_2(12, c_2, x)$$

$$[I_1(12, 22, x) \wedge I_2(12, 22, x)] \Rightarrow \Psi(x)$$

□

La décomposition choisie est la suivante (Cousot- Cousot [84]) :

$S = (C_0 \times \ldots \times C_{m-1}) \times M$

$A\Delta = (S \to \{tt, ff\})$

$A\tilde{\Delta} = (m \to (C_0 \times \ldots \times C_{m-1} \times M \to \{tt, ff\}))$

$\alpha \in (A\Delta \to A\tilde{\Delta})$

$\alpha(I)_i\ (c_0, \ldots, c_{m-1}, m) = I(\langle\langle c_0, \ldots, c_{m-1}\rangle, m\rangle)$

$\gamma \in (A\tilde{\Delta} \to A\Delta)$

$\gamma(\tilde{I})(\langle\langle c_0, \ldots, c_{m-1}\rangle, m\rangle) = \bigwedge_{i \in m} \tilde{I}_i\ (c_0, \ldots, c_{m-1}, m)$

Exemple 4.3.2.4.6-2

La détermination des conditions de vérification pour démontrer la correction partielle de programmes parallèles asynchrones de la forme :

$[\![Pra_0 \| \ldots \| Pra_{m-1}]\!]$

est tout à fait similaire à 4.3.2.4.3-1 . Nous obtenons (Cousot- Cousot [84]):

- Initialisation

$\phi(m) \Rightarrow \forall i \in m.\ \tilde{I}_i\ (\underline{L}_0, \ldots, \underline{L}_{m-1}, m)$ (où $Pra_i \equiv \underline{L}_i : \alpha$, $i \in m$)

- Preuve séquentielle

$[\tilde{I}_i\ (c_0, \ldots, c_{i-1}, c'_i, c_{i+1}, \ldots, c_{m-1}, m') \wedge t[\![Pra_i]\!](\langle c'_i, m'\rangle, \langle c_i, m\rangle)]$

$\Rightarrow \tilde{I}_i\ (c_0, \ldots, c_{i-1}, c_i, c_{i+1}, \ldots, c_{m-1}, m)$

- Preuve d'absence d'interférences , $(i \neq k)$:

$[\tilde{I}_i\ (c_0, \ldots, c_{k-1}, c'_k, c_{k+1}, \ldots, c_{m-1}, m') \wedge t[\![Pra_k]\!](\langle c'_k, m'\rangle, \langle c_k, m\rangle)]$

$\Rightarrow \tilde{I}_i\ (c_0, \ldots, c_{k-1}, c_k, c_{k+1}, \ldots, c_{m-1}, m')$

- Finalisation

$\forall i \in m.\ \tilde{I}_i\ (\bar{L}_0, \ldots, \bar{L}_{m-1}, m) \Rightarrow \psi(m)$ (où $Pra_i \equiv \alpha \bar{L}_i :$, $i \in m$)

□

Exemple 4.3.2.4.6-3

Nous illustrons la méthode de preuve 4.3.2.4.6-2 sur l'exemple 2.8.2.3. De manière évidente il ne peut s'agir que d'une reformulation de 4.3.2.4.4-2 en utilisant un invariant global par processus plutôt que des invariants locaux associés à chaque point de contrôle du programme. Comme les processus du programme sont symétriques, nous utilisons le même invariant $\tilde{I}$ pour $\tilde{I}_1$ et $\tilde{I}_2$, ce qui permet de n'avoir à raisonner que sur un seul processus.

Pour pouvoir désigner certains états de contrôle du programme, nous introduisons (cf. Cousot-Cousot [84]) :

Not-started = $(c_1 = 11 \wedge c_2 = 21)$

Pra_2-started = $(c_1 = 11 \wedge c_2 \neq 21)$

Pra_1-started = $(c_1 \neq 11 \wedge c_2 = 21)$

Started = $(c_1 \neq 11 \wedge c_2 \neq 21)$

L'idée centrale du programme 2.8.2.3 est de maintenir invariante la relation suivante :

$$Inv(\underline{m}, c_1, c_2, m, p_1, p_2) = [(\text{Not-started} \wedge m = \underline{m}) \vee (Pra_2\text{-started} \wedge p_2 = 2^{\underline{m}-m}) \vee (Pra_1\text{-started} \wedge p_1 = 2^{\underline{m}-m}) \vee (\text{started} \wedge p_1 \times p_2 = 2^{\underline{m}-m})]$$

L'autre observation essentielle pour la preuve de correction partielle est que le programme peut se terminer seulement quand $0 \leq m \leq 1$. Pour montrer ceci, nous introduisons :

Before-test = $[(c_1 \in \{11, 12\} \wedge c_2 \in \{21, 22\}) \vee (c_1 = 14 \wedge c_2 = 24)]$

After-test = $[(c_1 = 13 \wedge c_2 \in \{21, 22, 23\}) \vee (c_1 \in \{11, 12, 13\} \wedge c_2 = 23)]$

After-test-and-decrement = $[(c_1 = 14 \wedge c_2 \in \{21, 22, 23\}) \vee (c_1 \in \{11, 12, 13\} \wedge c_2 = 24)]$

One-decrement-left = $[(c_1 = 15 \wedge c_2 = 23) \vee (c_1 = 13 \wedge c_2 = 25)]$

No-decrement-left = $[(c_1 = 15 \wedge c_2 \neq 23) \vee (c_1 \neq 13 \wedge c_2 = 25)]$

Pour chaque processus, nous pouvons choisir l'invariant global suivant :

$$\tilde{I}(\underline{m}, c_1, c_2, m, p_1, p_2) = [Inv(\underline{m}, c_1, c_2, m, p_1, p_2) \wedge [(Before\text{-}test \wedge m \geq 0) \vee (After\text{-}test \wedge m > 1) \vee (After\text{-}test\text{-}and\text{-}decrement \wedge m \geq 1) \vee (One\text{-}decrement\text{-}left \wedge m = 1) \vee (No\text{-}decrement\text{-}left \wedge 0 \leq m \leq 1)]]$$

qui satisfait les conditions de vérification suivantes :

- Initialisation

$$[\underline{m} \geq 0] \Rightarrow \tilde{I}(\underline{m}, 11, 21, \underline{m}, p_1, p_2)$$

- Preuve séquentielle du processus 1

$$\tilde{I}(\underline{m}, 11, c_2, m, p_1', p_2) \Rightarrow \tilde{I}(\underline{m}, 12, c_2, m, 1, p_2)$$

$$[\tilde{I}(\underline{m}, 12, c_2, m, p_1, p_2) \wedge m > 1] \Rightarrow \tilde{I}(\underline{m}, 13, c_2, m, p_1, p_2)$$

$$[\tilde{I}(\underline{m}, 12, c_2, m, p_1, p_2) \wedge m \leq 1] \Rightarrow \tilde{I}(\underline{m}, 15, c_2, m, p_1, p_2)$$

$$\tilde{I}(\underline{m}, 13, c_2, m', p_1', p_2) \Rightarrow \tilde{I}(\underline{m}, 14, c_2, m'-1, 2 \times p_1', p_2)$$

$$[\tilde{I}(\underline{m}, 14, c_2, m, p_1, p_2) \wedge m > 1] \Rightarrow \tilde{I}(\underline{m}, 13, c_2, m, p_1, p_2)$$

$$[\tilde{I}(\underline{m}, 14, c_2, m, p_1, p_2) \wedge m \leq 1] \Rightarrow \tilde{I}(\underline{m}, 15, c_2, m, p_1, p_2)$$

- La preuve d'absence d'interférences du processus 2 avec l'invariant global du processus 1 est exactement la preuve séquentielle du processus 2

- Finalisation

$$\tilde{I}(\underline{m}, 15, 25, m, p_1, p_2) \Rightarrow \tilde{I}_1(\underline{m}, m, p_1, p_2)$$

En comparaison avec 4.3.2.4.4-2, l'utilisation d'une décomposition moins fine conduit, pour cet exemple, à une factorisation naturelle de conditions de vérification similaires et donc seulement à 8 conditions de vérification (au lieu de 77 dans 4.3.2.4.4-2). Cet exemple montre que le choix de la bonne décomposition de l'invariant global en invariants locaux dépend du problème à résoudre.

□

4.3.2.4.7 Utilisation d'un invariant global et d'invariants locaux

Un certain nombre de méthodes de preuve de propriétés d'invariance des programmes utilisent un invariant global sur l'état mémoire (qui s'appelle le "resource invariant" chez Hoare[72], Owicki-Gries[76b], le "monitor invariant" chez Howard[76], le "global invariant" chez Apt-Francez-deRoever[80], etc.) en même temps que des invariants locaux associés à divers points de contrôle du programme et portant sur l'état mémoire et l'état de contrôle (ou bien sur l'état mémoire uniquement en utilisant des variables auxiliaires pour simuler l'état de contrôle).

Si $S = (C_0 \times \ldots \times C_{m-1}) \times M$ et $A_S = (S \to \{tt, ff\})$, cette décomposition se définit comme suit :

$$\tilde{A_S} = \tilde{A_S}_g \times \tilde{A_S}_l$$

où $\quad \tilde{A_S}_g = (M \to \{tt, ff\})$

$$\tilde{A_S}_l = (\tilde{A_S}_0 \times \ldots \times \tilde{A_S}_{m-1})$$

$$\tilde{A_S}_i = (C_i \to (C_0 \times \ldots \times C_{i-1} \times C_{i+1} \times \ldots \times C_{m-1} \times M \to \{tt, ff\})), \quad i \in m$$

$$\alpha \in (A_S \to \tilde{A_S})$$

$$\alpha(I) = \langle \tilde{I}_g, \tilde{I}_l \rangle$$

où $\quad \tilde{I}_g(m) = (\exists c_0 \in C_0, \ldots, c_{m-1} \in C_{m-1} \cdot I(\langle\langle c_0, \ldots, c_{m-1}\rangle, m\rangle))$

$$\tilde{I}_{l_{i,c_i}}(c_0, \ldots, c_{i-1}, c_{i+1}, \ldots, c_{m-1}, m) = I(\langle\langle c_0, \ldots, c_i, \ldots, c_{m-1}\rangle, m\rangle)$$

$$\gamma \in (\tilde{A_S} \to A_S)$$

$$\gamma(\langle \tilde{I}_g, \tilde{I}_l \rangle)(\langle\langle c_0, \ldots, c_{m-1}\rangle, m\rangle) = [\tilde{I}_g(m) \wedge \forall i \in m, c \in C_i \cdot \tilde{I}_{l_{i,c}}(c_0, \ldots, c_{i-1}, c_{i+1}, \ldots, c_{m-1}, m)]$$

Cette décomposition est évidemment sémantiquement complète puisque l'invariant global $\tilde{G}$ est redondant et les invariants locaux $\tilde{I}$ peuvent être choisis comme en 4.3.2.4.4.

Pour concilier les avantages des décompositions 4.3.2.4.4 et 4.3.2.4.6 on peut envisager de faire porter l'invariant global $\tilde{G}$ sur l'état mémoire également sur l'état de contrôle, ce qui donne :

$\tilde{A_{sg}} = C_0 \times \ldots \times C_{m-1} \times M \to \{tt, ff\}$)

$\tilde{G}(c_0, \ldots, c_{m-1}, m) = I(\langle\langle c_0, \ldots, c_{m-1}\rangle, m\rangle)$

Dans le cas de programmes parallèles comportant des variables globales qui peuvent être modifiées par tous les processus et des variables locales qui ne sont visibles que dans le processus où elles sont déclarées, on peut imaginer de faire porter l'invariant global sur les variables globales et l'état de contrôle et de faire porter l'invariant local sur les variables locales visibles, sur les variables globales et sur l'état de contrôle. On a alors :

$S = (C_0 \times \ldots \times C_{m-1}) \times M \times (M'_0 \times \ldots \times M'_{m-1})$

$A_s = (S \to \{tt, ff\})$

$\tilde{A_s} = \tilde{A_{sg}} \times \tilde{A_{sl}}$

où $\tilde{A_{sg}} = (C_0 \times \ldots \times C_{m-1} \times M \to \{tt, ff\})$

$\tilde{A_{sl}} = (\tilde{A_{s_0}} \times \ldots \times \tilde{A_{s_{m-1}}})$

$\tilde{A_{s_i}} = (C_i \to (C_0 \times \ldots \times C_{i-1} \times C_{i+1} \times \ldots \times C_{m-1} \times M \times M'_i \to \{tt, ff\}))$, $i \in m$

$\alpha \in (A_s \to \tilde{A_s})$

$\alpha(I) = \langle \tilde{I}_g, \tilde{I}_l \rangle$

où $\tilde{I}_g(c_0, \ldots, c_{m-1}, m) = [\exists m'_0 \in M'_0, \ldots, m'_{m-1} \in M'_{m-1} . I(\langle\langle c_0, \ldots, c_{m-1}\rangle, m, \langle m'_0, \ldots, m'_{m-1}\rangle\rangle)]$

$\tilde{I}_{l_{i,c_i}}(c_0, \ldots, c_{i-1}, c_{i+1}, \ldots, c_{m-1}, m, m') = [\exists m'_0 \in M'_0, \ldots, m'_{i-1} \in M'_{i-1}, m'_{i+1} \in M'_{i+1}, \ldots, m'_{m-1} \in M'_{m-1} .$
$I(\langle\langle c_0, \ldots, c_{m-1}\rangle, m, \langle m'_0, \ldots, m'_{i-1}, m', m'_{i+1}, \ldots, m'_m\rangle\rangle)]$

$\gamma \in (\tilde{A_s} \to A_s)$

$\gamma(\langle \tilde{I}_g, \tilde{I}_l \rangle)(\langle\langle c_0, \ldots, c_{m-1}\rangle, m, \langle m'_0, \ldots, m'_{m-1}\rangle\rangle) =$

$[\tilde{I}_g(c_0, \ldots, c_{m-1}, m) \wedge \forall i \in m . \tilde{I}_{l_{i,c_i}}(c_0, \ldots, c_{i-1}, c_{i+1}, \ldots, c_{m-1}, m, m'_i)]$

On pourrait croire que l'utilisation de variables locales au lieu de variables globales permet de localiser les invariants en ce sens que l'invariant associé à un point de programme ne porte que sur les variables du programme qui sont visibles en ce point du programme. Cette approche n'est pas complète, comme le montre l'exemple suivant :

```
var M : integer ;
    F : boolean ;
    S1 : sémaphore :=1 ;
    S2 : sémaphore :=0 ;
M:=0 ; F:= false ;
[[  var T1 : integer ;
    T1 := 0 ;
    while T1<10 do
        P(S1) ;
        T1 := T1+1 ;
        V(S2) ;
    od ;
    P(S1) ;
    F := true ;
    V(S2) ;
    ⟨ M := M+T1 ⟩ ;
    var T2 : integer ;
    T2 := 0 ;
    P(S2) ;
    while not F do
        T2 : T2-1 ;
        V(S1) ;
        P(S2) ;
    od ;
    ⟨ M := M+T2 ⟩ ;
]] ;
```

Pour démontrer que la valeur finale de M est nulle, il faut évidemment pouvoir établir une relation entre les valeurs finales de T1 et T2 et donc une relation entre les valeurs courantes de T1 et T2 ce qui est impossible avec la décomposition choisie. (Cet argument peut être formalisé à l'aide de points fixes comme en 4.3.2.4.3).

Exemple 4.3.2.4.7-1

La méthode de Levin [79] pour CSP (Hoare [78]) utilise une décomposition similaire mais qui ne porte pas sur l'état de contrôle, les invariants locaux portant sur les variables des processus de CSP et des variables auxiliaires et l'invariant global ne portant que sur des variables auxiliaires. La complétude sémantique de la méthode vient de l'utilisation des variables auxiliaires qui permettent de simuler les relations entre les états de contrôle des processus (comme en 4.3.2.4.5) mais également d'exprimer indirectement les relations entre variables locales des processus à l'aide de variables auxiliaires globales (la technique consistant à recopier les variables locales dans les variables auxiliaires globales après chaque modification des variables locales).

4.3.2.4.8 Classification des méthodes de preuve d'invariance selon la finesse de la décomposition de l'invariant global en invariants locaux

Les méthodes de preuve de programmes peuvent être classées selon la classe des propriétés qu'elles permettent de démontrer (invariance conditionnelle, invariance, fatalité, ...).

Les diverses méthodes pour démontrer des propriétés de programmes dans une même classe peuvent être classées selon le principe d'induction dont elles dérivent.

Dans le cas de propriétés d'invariance, les méthodes de preuve dérivant d'un même principe d'induction de la forme :

$$[\exists I \in As.\ Co(I)]$$

sont de la forme :

$$[\exists \tilde{I} \in \tilde{As}.\ \tilde{Co}(\tilde{I})]$$

et s'obtiennent au moyen d'une correspondance (α, γ) entre As et $\tilde{As}$ telle que $\tilde{Co} \Rightarrow Co \circ \gamma$ (condition suffisante de correction, et éventuellement $Co \Rightarrow \tilde{Co} \circ \alpha$ (condition suffisante de complétude)).
Nous dirons qu'une méthode M1 de preuve basée sur une décomposition $(\tilde{As}_1, (\alpha_1, \gamma_1))$ de As peut se dériver d'une méthode M2 de preuve basée sur une décomposition $(\tilde{As}_2, (\alpha_2, \gamma_2))$ de As, quand il existe une décomposition $(\tilde{As}, (\alpha, \gamma))$ de $\tilde{As}_2$ telle que $\alpha_1 = \alpha \circ \alpha_2$ et $\gamma_1 = \gamma_2 \circ \gamma$ (auquel cas nous écrirons $M_1 \leftarrow M_2$).

Exemple 4.3.2.4.8-1

Si on a démontré une propriété d'invariance pour un programme $[\![Pra_0 \| \ldots \| Pra_{n-1}]\!]$ en utilisant la méthode 4.3.2.4.C, c'est-à-dire en utilisant des invariants $G_0, \ldots, G_{n-1}$ sur l'état de contrôle et les variables associés

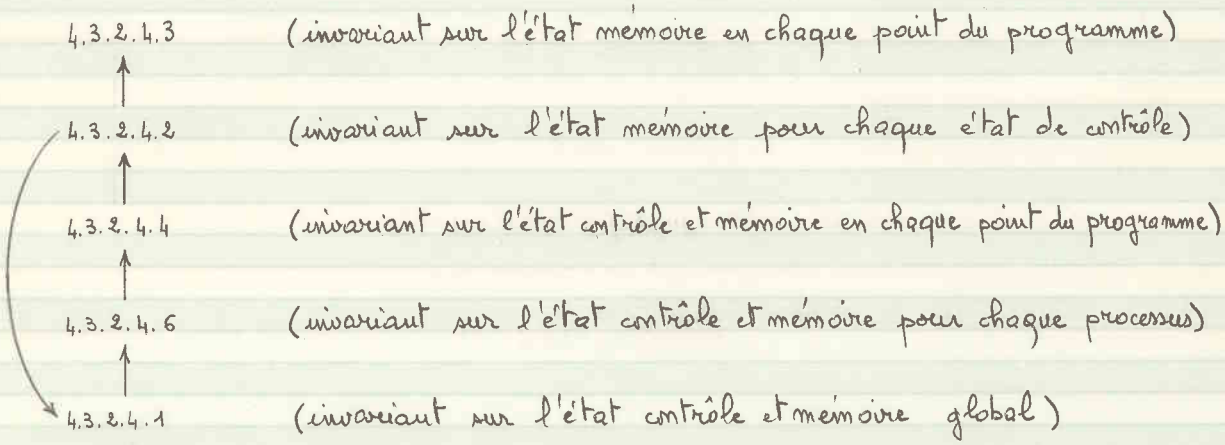
à chaque processus $Pra_0, \ldots, Pra_{m-1}$ du programme, on peut reformuler cette preuve pourqu'elle corresponde à la méthode 4.3.2.4.4 en utilisant des invariants $I_{i,c}$, $i \in m$, $c \in C_i$ définis par $I = \alpha(G)$ tel que

$$I_{i,c}(c_0, \ldots, c_{i-1}, c_{i+1}, \ldots, c_{m-1}, m) = G_i(c_0, \ldots, c_{i-1}, c, c_{i+1}, \ldots, c_{m-1}, m)$$

Réciproquement, une preuve utilisant un invariant global par processus (cf. 4.3.2.4.6) peut se dériver d'une preuve utilisant des invariants locaux associés à chaque point du programme (cf. 4.3.2.4.4) en utilisant $G = \gamma(I)$ tel que

$$G_i(c_0, \ldots, c_{m-1}, m) = \bigvee_{c \in C_i} I_{i,c}(c_0, \ldots, c_{i-1}, c_{i+1}, \ldots, c_{m-1}, m)$$

Plus généralement, les méthodes de preuve considérées au paragraphe 4.3.2.4 peuvent se dériver les unes des autres comme suit :



□

La relation de dérivation est un préordre (transitive et reflexive). Nous dirons que deux méthodes de preuve (basées sur un même principe d'induction) sont "sémantiquement équivalentes" quand chacune se dérive de l'autre. La relation de dérivation induite sur l'ensemble des méthodes de preuve quotienté par la relation d'équivalence sémantique est un ordre partiel. Le supremum correspond au principe d'induction et l'infimum au choix $\tilde{As} = 1$, $\alpha(P) = 0$, $\gamma(0) = tt$, $\tilde{Cv}(\tilde{I}) = tt$ (une méthode correcte mais qui permet de ne rien démontrer). On remarquera que cet ensemble ordonné contient le treillis complet des méthodes d'analyse sémantique des programmes considéré dans Cousot-Cousot [79a].

4.3.2.5 Analyse sémantique des programmes

L'analyse sémantique d'un programme (Cousot P. [77,78,81], Cousot-Cousot [76, 77a, 77b, 77c, 77d, 79a, 80a, 84].) est une analyse statique (i.e. sans exécuter le programme) des propriétés dynamiques (i.e. à l'exécution) de ce programme.

4.3.2.5.1 Analyse d'invariance "en avant"

Soit $\langle S', A', \Sigma' \rangle$ la sémantique d'un programme. L'analyse sémantique d'invariance "en avant" consiste à caractériser la relation $\underset{\sim}{D}'(\phi, \langle S', A', \Sigma' \rangle)$ entre les états $\underline{s}$ satisfaisant une condition ϕ et leurs descendants possibles $\bar{s}$:

$$\underset{\sim}{D}'(\phi, \langle S', A', \Sigma' \rangle) \in (S^2 \to \{tt, ff\})$$

$$\underset{\sim}{D}'(\phi, \langle S', A', \Sigma' \rangle)(\underline{s}, \bar{s}) = [\exists p \in \Sigma', i, j \in |p| . (\phi(\underline{s}) \wedge p_i = \underline{s} \wedge i \leq j \wedge p_j = \bar{s})]$$

Il est toujours possible de se ramener au cas où la condition ϕ ne porte que sur les états initiaux en considérant la sémantique $\langle S, A, \Sigma \rangle$ telle que $S = S'$, $A = A'$ et

$$\Sigma = \{ p \in \Sigma \langle S, A \rangle : \phi(p_0) \wedge \exists q \in \Sigma' . p \twoheadrightarrow q \}$$

et en posant :

$$\underset{\sim}{D}(\phi, \langle S, A, \Sigma \rangle) \in (S^2 \to \{tt, ff\})$$

$$\underset{\sim}{D}(\phi, \langle S, A, \Sigma \rangle)(\underline{s}, \bar{s}) = [\exists p \in \Sigma, j \in |p| . (\phi(p_0) \wedge p_0 = \underline{s} \wedge p_j = \bar{s})]$$

car

$$\underset{\sim}{D}'(\phi, \langle S', A', \Sigma' \rangle) = \underset{\sim}{D}(\phi, \langle S, A, \Sigma \rangle)$$

La relation $\underset{\sim}{D}(\phi, \langle S, A, \Sigma \rangle)$ n'étant pas calculable pour tous les programmes, on ne peut envisager que de calculer des approximations supérieures $\tilde{\underset{\sim}{D}}(\phi, \langle S, A, \Sigma \rangle)$ telles que $\underset{\sim}{D}(\phi, \langle S, A, \Sigma \rangle) \Rightarrow \tilde{\underset{\sim}{D}}(\phi, \langle S, A, \Sigma \rangle)$ (ou inférieures, ce qui se traite par dualité).

Soit $\langle S, A, t, \varepsilon\rangle$ le système de transition engendré par $\langle S, A, \Sigma\rangle$. On a $\underset{\sim}{D}(\phi, \langle S, A, \Sigma\rangle) \Rightarrow \underset{\sim}{D}(\phi, \langle S, A, \Sigma\langle S, A, t, \varepsilon\rangle\rangle)$ (en effet $D(\phi, \langle S, A, \Sigma\langle S, A, t, \varepsilon\rangle\rangle)$ est invariante pour $\langle S, A, \Sigma\langle S, A, t, \varepsilon\rangle\rangle$ et donc pour $\langle S, A, \Sigma\rangle$ d'après 4.1.3 n°3). Une première approximation supérieure consiste donc à raisonner sur le système de transition $\langle S, A, t, \varepsilon\rangle$ de sorte qu'en posant :

$$\underset{\sim}{D}(\langle S, A, t, \varepsilon\rangle) \in (S^2 \to \{tt, ff\})$$

$$\underset{\sim}{D}(\langle S, A, t, \varepsilon\rangle)(\underline{s}, \bar{s}) = [\varepsilon(\underline{s}) \wedge t^*(\underline{s}, \bar{s})]$$

il s'agit de caractériser $\underline{D}(\langle S, A, t, \varepsilon \wedge \phi\rangle)$.

On remarque alors comme en 4.3.1.7-1 que $\underset{\sim}{D}(\langle S, A, t, \varepsilon\rangle)$ est le plus petit point fixe $\underset{\sim}{lfp}(F)$ pour $\Rightarrow$ de l'opérateur F sur $As = (S^2 \to \{tt, ff\})$ défini par :

$$F(I)(\underline{s}, \bar{s}) = [(\varepsilon(\underline{s}) \wedge \bar{s} = \underline{s}) \vee (\exists s' \in S, a \in A.\ I(\underline{s}, s') \wedge t_a(s', \bar{s}))]$$

de sorte qu'en utilisant une décomposition (α, γ) des invariants globaux de $\langle As, \Rightarrow\rangle$ en des invariants locaux de $\langle \tilde{As}, \sqsubseteq\rangle$ et un opérateur correspondant $\tilde{F}$ monotone sur $\tilde{As}$ tel que $\alpha \circ F \circ \gamma \sqsubseteq \tilde{F}$, on a $\underset{\sim}{lfp}(F) \Rightarrow \gamma(\underset{\sim}{lfp}(\tilde{F}))$ quand (α, γ) est une (demi-) correspondance de Galois et $\langle \tilde{As}, \sqsubseteq\rangle$ un treillis complet.

Si le treillis complet $\langle \tilde{As}, \sqsubseteq\rangle$ satisfait la condition de chaîne ascendante (toute chaîne strictement croissante pour $\sqsubset$ est finie) alors $\underset{\sim}{lfp}(\tilde{F})$ est calculable itérativement comme $\bigsqcup_{n \geq 0} \tilde{F}^n(\alpha(ff))$.

Si l'itération $X^0 = \alpha(ff), \ldots, X^{n+1} = \tilde{F}(X^n)$ peut ne pas converger en un nombre fini de pas, on utilisera une technique d'extrapolation de la plus petite solution $\underset{\sim}{lfp}(\tilde{F})$ du système d'équations $X = \tilde{F}(X)$ en calculant la limite $\tilde{I} = \bigsqcup_{n \geq 0} \tilde{F}^{\nabla n}(\alpha(ff))$ où $\tilde{F}^{\nabla}(X) = X \nabla \tilde{F}(X)$, d'une itération croissante avec élargissement ∇ satisfaisant :

$$\forall X, Y \in \tilde{As}.\ [(X \sqcup Y) \sqsubseteq (X \nabla Y)]$$

(ce qui assure que $\underset{\sim}{lfp}(\tilde{F}) \sqsubseteq \tilde{I}$) et la condition qu'il n'existe pas de chaîne infinie strictement croissante de la forme $X^0, \ldots, X^{i+1} = X^i \nabla \tilde{F}(X^i), \ldots$ (ce qui assure la convergence).

Si $\tilde{I}$ n'est pas un point fixe de $\tilde{F}$, alors $\mathrm{lfp}_{\cdot}(\tilde{F}) \sqsubseteq \tilde{F}(\tilde{I}) \sqsubset \tilde{I}$ et l'approximation supérieure $\tilde{I}$ de $\mathrm{lfp}_{\cdot}(\tilde{F})$ peut être améliorée puisque $\mathrm{lfp}_{\cdot}(\tilde{F}) \sqsubseteq \tilde{F}^{n}(\tilde{I})$ pour tout $n \geq 0$. De manière plus générale il est ensuite possible d'améliorer l'approximation $\tilde{I}$ de $\mathrm{lfp}_{\cdot}(\tilde{F})$ en calculant la limite $\tilde{J} = \prod_{n \geq 0} \tilde{F}^{\Delta n}(\tilde{I})$ où $\tilde{F}^{\Delta}(X) = X \Delta \tilde{F}(X)$, d'une itération décroissante avec rétrécissement Δ tel que :

$$\forall X, Y \in \tilde{A}^{\sharp} . [(X \sqcap Y) \sqsubseteq (X \Delta Y) \sqsubseteq Y]$$

(de sorte que $\mathrm{lfp}_{\cdot}(\tilde{F}) \sqsubseteq \tilde{J}$) et toute chaine strictement décroissante de la forme $X^0, \ldots, X^{i+1} = X^i \Delta \tilde{F}(X^i), \ldots$ est finie (de sorte que l'itération converge).

En pratique, la décomposition (α, γ) est choisie sous la forme $(\alpha_a \circ \alpha_p, \gamma_p \circ \gamma_a)$ où (α_p, γ_p) est un tion utilisée pour une méthode de preuve (cf. par 4.3.2.4) et (α_a, γ_a) introduit une approximation, qui peut être grossière, determinée en fonction du problème posé. De ce fait l'équation $X = \tilde{F}(X)$ se présente généralement sous la forme d'un système d'équations :

$$\begin{cases} X_i = \tilde{F}_i(X_0, \ldots, X_{m-1}) \\ i \in m \end{cases}$$

Le calcul de $\bigsqcup_{n \geq 0} \tilde{F}^{\sharp n}(\alpha(ff))$ peut alors se faire en utilisant toute stratégie chaotique voire asynchrone équivalente (Cousot-P [77]). En particulier, étant donné le graphe de dépendance du système d'équations (i dépend de j si le résultat de $F_i(X_0, \ldots, X_{m-1})$ dépend de X_j), il peut être avantageux d faisant autes X_i corrs à une exe du graphe de dépendance

et d'autre part en procédant récursivement de manière identique à l'intérieur de chaque

composante fortement connexe. Dans le cas d'itérations utilisant une extrapolation (élargissement ou rétrécissement), il suffit d'utiliser l'extrapolation pour les composantes X_i telles que i est un point de coupure du graphe de dépendance (les points de coupure étant choisis de sorte que tout cycle dans le graphe de dépendance passe par un point de coupure).

Exemple 4.3.2.5.1-1

Pour étendre Cousot-Cousot [76, 77a] au cas des programmes parallèles de la forme :

$$[\![Pr_0 \parallel \ldots \parallel Pr_{m-1}]\!]$$

où

$$S = (C_0 \times \ldots \times C_{m-1}) \times (\mathcal{V} \to \mathcal{D}) \quad \text{et}$$
$$\mathcal{D} = \{ z \in \mathbb{Z} : l_i \leq z \leq h_i \}$$

on peut utiliser la décomposition :

$$A_\Delta = (S \to \{tt, ff\})$$
$$\tilde{A}_\Delta = \prod_{i \in m} \prod_{c \in C_i} (\mathcal{V} \to \mathbb{Z} \times \mathbb{Z})$$

où

$$\alpha(I)_{i,c}[V] = \langle \underset{\sim}{\inf}\{v \in \mathbb{Z} : P(v)\}, \underset{\sim}{\sup}\{v \in \mathbb{Z} : P(v)\} \rangle \quad \text{si } \exists v.\, P(v)$$
$$= \bot \quad \text{si } \forall v. \neg P(v)$$

et

$$P(v) = [\exists c_0 \in C_0, \ldots, c_{i-1} \in C_{i-1}, c_{i+1} \in C_{i+1}, \ldots, c_{m-1} \in C_{m-1}, m \in (\mathcal{V} \to \mathcal{D}).$$
$$I(\langle\langle c_0, \ldots, c_{i-1}, c, c_{i+1}, \ldots, c_{m-1} \rangle, m[V \leftarrow v]\rangle)]$$

Cette décomposition ne permet pas d'exprimer de relations entre les compteurs ordinaux des différents processus. On obtiendra donc des résultats plus précis en utilisant la décomposition suivante :

$$\tilde{A}_\Delta = \prod_{i \in m} \prod_{c \in C_i} (\langle C_0 \times \ldots \times C_{i-1} \times C_{i+1} \times \ldots \times C_{m-1} \rangle \to (\mathcal{V} \to \mathbb{Z} \times \mathbb{Z}))$$

où

$$\alpha(I)_{i,c}[c_0,\dots,c_{i-1},c_{i+1},\dots,c_{m-1},V] = \langle Inf\{v\in\mathbb{Z} : Q(v)\}, Sup\{v\in\mathbb{Z} : Q(v)\}\rangle \quad \text{si } \exists v.\, Q(v)$$
$$= \bot \quad \text{si } \forall v.\neg Q(v)$$

et

$$Q(v) = [\exists m \in (\mathbb{V}\to\mathbb{Z}).\ I(\langle\langle c_0,\dots,c_{i-1},c,c_{i+1},\dots,c_{m-1}\rangle, m[V\leftarrow v]\rangle]$$

Par exemple pour le programme 2.8.2.3 :

```
0: {m≥0}
   ||
     11:
          P1 := 1 ;
     12:
          while N>1 do
     13:
               ⌊ N:=N-1 ;  P1:=2×P1 ⌋ ;
     14:
          od ;
     15:
   ||
     21:
          P2 := 1 ;
     22:
          while N>1 do
     23:
               ⌊ N:=N-1 ;  P2:=2×P2 ⌋ ;
     24:
          od ;
     25:
   ||;
1:
   if N=0 then P:=P1×P2 else P:=2×P1×P2 fi ;
2:
```

L'invariant local approché D_{jl} attaché au point l du processus $j=1,2$ est choisi comme étant un intervalle de valeurs pour chaque variable $v\in\mathbb{V}$ et chaque point de contrôle $k\in C_{\tilde{j}}$ de l'autre processus $\tilde{j}$ (où $\tilde{1}=2$ et $\tilde{2}=1$). $D_{jl}(\tilde{j}k)$ est alors un triplet $\langle m, p_j, p_{\tilde{j}}\rangle$ des valeurs abstraites des variables N, P_j, $P_{\tilde{j}}$ où chaque m, p_j, $p_{\tilde{j}}$ est soit $\bot$ (qui corspond à ff) ou un intervalle de valeurs numériques $[a,b]$ tel que $li\le a\le b\le hi$ où li et hi sont respectivement les plus petit et plus grand entier representable en machine. Par exemple $D_{14}(23) = \langle[1,hi-1],[2,hi],[1,hi]\rangle$ signifie qu'au point 14 du processus 1, il est vrai que $((1\le m\le hi-1)\wedge(2\le p_1\le hi)\wedge(1\le p_2\le hi))$ quand le contrôle est au point 23 du processus 2.

Nous utiliserons les notations suivantes :

- $\langle m, p_j, p_j^{\sim} \rangle [N] = m$, $\langle m, p_j, p_j^{\sim} \rangle [p_j] = p_j$, $\langle m, p_j, p_j^{\sim} \rangle [p_j^{\sim}] = p_j^{\sim}$

- $\langle m, p1, p2 \rangle [p1/p1'] = \langle m, p1', p2 \rangle$ (substitution)

  $\langle m, p1, p2 \rangle [p2/p2'] = \langle m, p1, p2' \rangle$

- $\bot \wedge x = x \wedge \bot = \bot$ si $x \in (\{\bot\} \cup \{[a,b] : a \leq b\})$ (conjonction approchée)

  $[a,b] \wedge [c,d] = [\underset{\sim}{Sup}(a,c), \underset{\sim}{Inf}(b,d)]$

  ssi $\underset{\sim}{Sup}(a,c) \leq \underset{\sim}{Inf}(b,d)$

  $= \bot$ ssi $(b<c)$ ou $(d<a)$

- $\bot \vee x = x \vee \bot = x$ si $x \in (\{\bot\} \cup \{[a,b] : a \leq b\})$ (disjonction approchée)

  $[a,b] \vee [c,d] = [\underset{\sim}{Inf}(a,c), \underset{\sim}{Sup}(b,d)]$

- $\bot - 1 = \bot$

  $[a,b] - 1 = [a-1, b-1] \wedge [li, hi]$ (decrémentation approchée)

- $2 \times \bot = \bot$

  $2 \times [a,b] = [2 \times a, 2 \times b] \wedge [li, hi]$ (décalage gauche approché)

Dans le système d'équations approchées de point fixe suivant nous supposons qu'initialement nous avons $n \geq 0$. Pour chaque équation nous distinguons un terme correspondant à la preuve séquentielle et un terme correspondant au contrôle d'absence d'interférence :

$D_0 = \langle [0, hi], [li, hi], [li, hi] \rangle$

$D_{11}(2\cdot1) = D_0$

$D_{11}(2k) = inter_{11}(2k)$ $k = 2, \ldots, 5$

$D_{12}(2k) = D_{11}(2k)[p1/[1,1]] \vee inter_{12}(2k)$ $k = 1, \ldots, 5$

$D_{13}(2k) = (D_{12}(2k) \wedge \langle [2, hi], [li, hi], [li, hi] \rangle) \vee (D_{14}(2k) \wedge \langle [2, hi], [li, hi], [li, hi] \rangle) \vee inter_{13}(2k)$ $k = 1, \ldots, 5$

$D_{14}(2k) = \langle D_{13}(2k)[N] - 1, 2 \times D_{13}(2k)[p1], D_{13}(2k)[p2] \rangle \vee inter_{14}(2k)$ $k = 1, \ldots, 5$

$D_{15}(2k) = (D_{12}(2k) \wedge \langle [li, 1], [li, hi], [li, hi] \rangle) \vee (D_{14}(2k) \wedge \langle [li, 1], [li, hi], [li, hi] \rangle) \vee inter_{15}(2k)$ $k = 1, \ldots, 5$

où

$$inter_{1R}(21) = \langle \perp, \perp, \perp \rangle$$

$$inter_{1R}(22) = (D_{1R}(21) \wedge D_{21}(1R)[p2 / [1,1]])$$

$$inter_{1R}(23) = (D_{1R}(22) \wedge D_{22}(1R) \wedge \langle [2,hi], [li,hi], [li,hi] \rangle) \vee (D_{1R}(24) \wedge D_{24}(1R) \wedge \langle [2,hi], [li,hi], [li,hi] \rangle)$$

$$inter_{1R}(24) = \langle (D_{1R}(23)[N] \wedge D_{23}(1R)[N]) - 1,\ D_{1R}(23)[P1] \wedge D_{23}(1R)[P1],\ \ldots \times (D_{1R}(23)[P2] \wedge D_{23}(1R)[P2]) \rangle$$

$$inter_{1R}(25) = (D_{1R}(22) \wedge D_{22}(1R) \wedge \langle [li,1], [li,hi], [li,hi] \rangle) \vee (D_{1R}(24) \wedge D_{24}(1R) \wedge \langle [li,1], [li,hi], [li,hi] \rangle)$$

... équations similaires pour le processus 2 ...

$$D_1 = D_{15}(25) \wedge D_{25}(15)$$

Ce système d'équations peut être résolu au moyen d'une stratégie itérative asynchrone (Cousot-P [77]). Initialement on pose $D_{il}(\check{i}k) = \langle \perp, \perp, \perp \rangle$ pour $i = 1, 2$, $l \in C_i$, $k \in C_i^{\sim}$. Puis on itère appliquant n'importe quelle équation du système d'équations jusqu'à stabilisation.

La convergence peut être accelerée utilisant les techniques d'extrapolation décrites dans Cousot-Cousot [76, 77]. Ceci consiste à définir un opérateur d'élargissement ∇ tel que :

$$\perp \nabla x = x$$

$$[a,b] \nabla [c,d] = [\underline{\text{if}}\ c<a\ \underline{\text{then}}\ li\ \underline{\text{else}}\ a\ ,\ \underline{\text{if}}\ d>b\ \underline{\text{then}}\ hi\ \underline{\text{else}}\ b]$$

et remplacer les équations D_{j3}, $j = 1, 2$ par :

$$D_{j3}(\check{j}k) = D_{j3}(\check{j}k) \nabla [(D_{j2}(\check{j}k) \wedge \langle [2,hi], [li,hi], [li,hi] \rangle) \vee (D_{j4}(\check{j}k) \wedge \langle [2,hi], [li,hi], [li,hi] \rangle) \vee inter_{j3}(\check{j}k)]$$

puis résoudre itérativement. Le résultat que nous avons obtenu (pour le processus 1) est :

	k=1			k=2			k=3		
	m	p1	p2	m	p1	p2	m	p1	p2
$D_{11}(2k)$	$\langle[0,hi],[li,hi],[li,hi]\rangle$			$\langle[0,hi],[li,hi],[1,1]$			$\langle[2,hi],[li,hi],[1,hi]\rangle$		
$D_{12}(2k)$	$\langle[0,hi],[1,1],[li,hi]\rangle$			$\langle[0,hi],[1,1],[1,1]\rangle$			$\langle[2,hi],[1,1],[1,hi]\rangle$		
$D_{13}(2k)$	$\langle[2,hi],[1,hi],[li,hi]\rangle$			$\langle[2,hi],[1,hi],[1,1]\rangle$			$\langle[2,hi],[1,hi],[1,hi]\rangle$		
$D_{14}(2k)$	$\langle[1,hi-1],[2,hi],[li,hi]\rangle$			$\langle[1,hi-1],[2,hi],[1,1]\rangle$			$\langle[1,hi-1],[2,hi],[1,hi]\rangle$		
$D_{15}(2k)$	$\langle[0,1],[1,hi],[li,hi]\rangle$			$\langle[0,1],[1,hi],[1,1]\rangle$			$\langle[1,1],[2,hi],[1,hi]\rangle$		

	k=4			k=5		
	m	p1	p2	m	p1	p2
$D_{11}(2k)$	$\langle[1,hi-1],[li,hi],[2,hi]\rangle$			$\langle[0,1],[li,hi],[1,hi]$		
$D_{12}(2k)$	$\langle[1,hi-1],[1,1],[2,hi]\rangle$			$\langle[0,1],[1,1],[1,hi]\rangle$		
$D_{13}(2k)$	$\langle[1,hi-1],[1,hi],[2,hi]\rangle$			$\langle[1,1],[1,hi],[2,hi]\rangle$		
$D_{14}(2k)$	$\langle[0,hi-2],[2,hi],[2,hi]\rangle$			$\langle[0,1],[2,hi],[1,hi]\rangle$		
$D_{15}(2k)$	$\langle[0,1],[1,hi],[2,hi]\rangle$			$\langle[0,1],[1,hi],[1,hi]\rangle$		

Observons que nous obtenons :

$D_1 = \langle[0,1],[1,hi],[1,hi]\rangle$

ce qui montre que $0 \leq N \leq 1$ à la sortie de la commande parallèle du programme, ce qui n'est pas complètement trivial à démontrer à la main.

□

Exemple 4.3.2.5.1 - 2

Pour étendre Cousot-Holbwachs [78] au cas des programmes parallèles, il suffit de considérer une décomposition de la forme $(\alpha_a \circ \alpha_p, \gamma_p \circ \gamma_a)$ où (α_p, γ_p) décompose $(S^\sharp \to \{tt, ff\})$ en $\prod_{i \in m} \prod_{c \in C_i} (\mathbb{R}^n \to \{tt, ff\})$ (ou $\prod_{i \in m} \prod_{c \in C_i} (\mathbb{Z}^n \to \{tt, ff\})$) tandis que la décomposition (α_a, γ_a) consiste pour chaque composante $i \in m$ à approcher les éléments P_i de $(\mathbb{R}^n \to \{tt, ff\})$ par le prédicat caractérisant l'enveloppe convexe de $\{X : P_i(X)\}$. Les opérateurs $\sqcup$ et ∇ correspondants ont été proposés par Holbwachs [78]. En particulier si P et Q caractérisent deux polyèdres

convexes de $\mathbb{R}^m$ tels que $P \Rightarrow Q$ et $P \neq Q$ alors l'élargissement $P \nabla Q$ de P par Q est obtenu en éliminant du système de contraintes linéaires P toutes les inéquations non satisfaites par Q. Lorsque la dimension du polyèdre caractérisé par P est strictement inférieure à m, on réécrit au préalable le système d'inéquations P sous une forme qui maximise le nombre de contraintes satisfaites par Q.

Par exemple, en choisissant la décomposition (α_p, γ_p) comme en 4.3.2.4.4 ce qui permet d'associer à chaque point du programme une relation linéaire entre l'état mémoire initial et les états de contrôle et mémoire courants, on obtient pour le programme 2.8.3.3.1 les résultats suivants (nous ignorons la variable Any qui ne peut prendre qu'une seule valeur) :

0: {true}

[[11: $\{c_2 \geq 21 \wedge c_3 \geq 31 \wedge c_3 \geq c_2 + 9 \wedge c_3 \geq 2c_2 - 14 \wedge 2c_3 \leq c_2 + 44 \wedge c_3 \leq c_2 + 11\}$

while true do

12: $\{c_2 \geq 21 \wedge c_3 \geq 31 \wedge c_3 \geq c_2 + 9 \wedge c_3 \geq 2c_2 - 14 \wedge c_3 \leq 34 \wedge c_3 \leq c_2 + 11\}$

P! any;

14: $\{c_3 = 24 \wedge 21 \leq c_2 \leq 23\}$

V! any;

13: $\{c_2 \geq 21 \wedge c_3 \geq 32 \wedge c_3 \geq 2c_2 - 14 \wedge c_3 \leq 34 \wedge c_3 \leq c_2 + 12\}$

od;

15: {false}

```
||
21: { c1 ≥ 11 ∧ c3 ≥ 31 ∧ c2 ≥ c1 + 19 ∧ c3 ≥ 2c1 + 6 ∧ 2c3 ≤ c1 + 54 ∧ c3 ≤ c2 + 21 }
    while true do
22:     { c1 ≥ 1 ∧ c3 ≥ 31 ∧ c3 ≥ c1 + 19 ∧ c3 ≥ 2c1 + 6 ∧ c3 ≤ 34 ∧ c3 ≤ c1 + 21 }
        P! any ;
24:     { c3 = 34 ∧ 11 ≤ c1 ≤ 13 }
        V! any ;
23:     { c1 ≥ 11 ∧ c3 ≥ 32 ∧ c3 ≥ 2c1 + 6 ∧ c3 ≤ 34 ∧ c3 ≤ c1 + 22 }
    od ;
25: { false }
||
31: { 11 ≤ c2 ≤ 12 ∧ 21 ≤ c2 ≤ 22 }
    while true do
32:     { 11 ≤ c1 ≤ 13 ∧ 21 ≤ c2 ∧ 23 ∧ c1 + c2 ≤ 35 }
        P? Any ;
34:     { c1 ≤ 14 ∧ c2 ≤ 24 ∧ c1 + c2 ≤ 37 ∧ 2c2 + c1 ≥ 56 ∧ 2c1 + c2 ≥ 46 }
        V? Any ;
33:     { 11 ≤ c1 ≤ 13 ∧ 21 ≤ c2 ≤ 23 ∧ c1 + c2 ≥ 33 }
    od ;
35: { false }
|| ;
1: { false }
```

Ces invariants sont suffisants pour démontrer que les points 14 et 24 sont en exclusion mutuelle (puisque si le contrôle pouvait être simultanément en 14 et 24 on aurait $c_1 = 14 \wedge \{ c_3 = 24 \wedge 21 \leq c_2 \leq 23 \}$ et $c_2 = 24 \wedge \{ c_3 = 34 \wedge 11 \leq c_1 \leq 13 \}$ ce qui est faux !). Malheureusement le choix de la numérotation des points du programme a une influence sur la précision des résultats, comme le montre l'exemple suivant (le renumérotage interdisant l'expression de l'exclusion mutuelle à l'aide d'une conjonction d'inégalités linéaires) :

0: {true}

[[11: $\{21 \le c_2 \le 24 \wedge 31 \le c_3 \le 34 \wedge 2c_2 - c_3 \ge 10 \wedge 2c_3 - c_2 \ge 40\}$

while true do

12: $\{21 \le c_2 \le 24 \wedge 31 \le c_3 \le 34 \wedge 2c_3 - c_2 \ge 40\}$

P! any;

13: $\{c_2 \le 24 \wedge 32 \le c_3 \le 34 \wedge c_3 - c_2 \le 12 \wedge c_3 + c_2 \ge 54\}$

V! any;

14: $\{21 \le c_2 \le 24 \wedge 32 \le c_3 \le 24\}$

od;

15: {false}

||

21: $\{11 \le c_1 \le 14 \wedge 31 \le c_3 \le 34 \wedge c_3 - 2c_1 \le 10 \wedge 2c_3 - c_1 \ge 50\}$

while true do

22: $\{11 \le c_1 \le 14 \wedge 31 \le c_3 \le 34 \wedge 2c_3 - c_1 \ge 50\}$

P! any;

23: $\{c_1 \le 14 \wedge 32 \le c_3 \le 34 \wedge c_3 - c_1 \le 22 \wedge c_3 + c_1 \ge 44\}$

V! any;

24: $\{11 \le c_1 \le 14 \wedge 32 \le c_3 \le 24\}$

od;

25: {false}

31: $\{11 \le c_1 \le 12 \wedge 21 \le c_2 \le 22\}$

while true do

32: $\{11 \le c_1 \le 14 \wedge 21 \le c_2 \le 24\}$

P? Any;

33: $\{11 \le c_1 \le 14 \wedge 21 \le c_2 \le 24 \wedge c_1 + c_2 \le 37 \wedge c_1 + c_2 \ge 33\}$

V? Any;

34: $\{11 \le c_1 \le 14 \wedge 21 \le c_2 \le 24 \wedge c_1 + c_2 \ge 33\}$

od;

35: {false}

]];

1: {false}

Ces résultats sont maintenant trop faibles pour démontrer l'exclusion mutuelle des points 13 et 23 du programme (puisque $c_1=13 \wedge \{ c_2 \leq 24 \wedge 32 \leq c_3 \leq 34 \wedge c_3 - c_2 \leq 12 \wedge c_3 + c_2 \geq 54 \} \wedge c_2 = 23 \wedge \{ c_1 \leq 14 \wedge 32 \leq c_3 \leq 34 \; c_3 - c_1 \leq 22 \wedge c_3 + c_1 \geq 44 \}$ n'est pas identiquement faux).

Il est évidemment possible d'éviter cette perte d'information, par exemple en choisissant (α_p, γ_p) comme en 4.3.2.4.2, ce qui revient à associer une relation linéaire entre l'état mémoire initial et courant à toute valeur de l'état de contrôle. Dans ce cas l'analyse donne comme résultat l'assertion true associée aux états de contrôle (11, 21, 31), (12, 21, 31), (11, 22, 31), (11, 21, 32), (12, 22, 31), (12, 21, 32), (11, 22, 32), (13, 21, 33), (12, 22, 32), (11, 23, 33), (14, 21, 34), (13, 22, 33), (12, 23, 33), (11, 24, 34), (12, 21, 34), (14, 22, 34), (14, 21, 32), (12, 24, 34), (11, 22, 34), (11, 24, 32), (12, 22, 34), (14, 22, 32), (12, 24, 32), (14, 23, 33), (13, 24, 33), (14, 24, 34), (14, 24, 32) et faux associé aux autres états de contrôle. (on remarque que la reunion de ces points n'est pas convexe d'où la perte d'information avec la décomposition moins fine ci-dessus). Comme pour l'exemple 4.3.2.5.1-1, le coût de ce gain en précision est qu'il faut maintenant considérer un nombre d'invariants de l'ordre du produit et non plus de la somme des nombres de points de contrôle des processus du programme.
□

4.3.2.5.2 Analyse d'invariance "en arrière"

On peut aussi s'intéresser à une approximation de la relation entre les états $\bar{s}$ satisfaisant une condition ψ et leurs ascendants possibles $\underline{s}$:

$$\underset{\sim}{A}(\psi, \langle S, A, \Sigma \rangle) \in (S^* \to \{tt, ff\})$$

$$\underset{\sim}{A}(\psi, \langle S, A, \Sigma \rangle)(\bar{s}, \underline{s}) = [\exists p \in \Sigma, i, j \in |p| \cdot \psi(\bar{s}) \wedge p_j = \bar{s} \wedge i \leq j \wedge p_i = \underline{s}]$$

ce qui se traite comme dans le cas précédent en appliquant la transformation -1 (cf. 4.2.1.2.3) sur $\underset{\sim}{Pref}^{<\omega}(<S,A,\Sigma>)$ et consiste donc à trouver une approximation supérieure de la plus petite solution d'un système d'équations $X = \tilde{B}(X)$ tel que $\alpha \circ B \circ \gamma \sqsubseteq \tilde{B}$ et

$$B(I)(\underline{s},\bar{s}) = [\,(\exists s' \in S,\ a \in A.\ t_a(\underline{s},s') \wedge I(s',\bar{s})) \vee (\underline{s} = \bar{s} \wedge \psi(\bar{s}))\,]$$

Exemple 4.3.2.5.2-1

Soit à chercher une condition suffisante $P(\underline{s})$ sur les états d'entrée d'un programme parallèle :

$$Ps \,[\![\, Pr_0 \,\|\, \ldots \,\|\, \alpha L : p \,\|\, \ldots \,\|\, \alpha' L' : p' \,\|\, \ldots \,\|\, Pr_{m-1} \,]\!]\, ; Ps'$$

pour que les points L et L' dans deux processus différents Pr_i et Pr_j soient mutuellement exclusifs. Soit $<S,A,\Sigma>$ la sémantique du programme. Posons $\mu \in (S \to \{tt,ff\})$ défini par $\mu(<L,M>) = tt$ et $\mu(<<L_0,\ldots,L_{m-1}>,M>) = \neg(L_i = L \wedge L_j = L')$. Il s'agit de trouver P tel que:

$$\forall p \in \Sigma.\ (P(p_0) \Rightarrow \forall i \in p.\ \mu(p_i))$$

Soit $<S,A,t,\varepsilon>$ le système de transition engendré par $<S,A,\Sigma>$. Il suffit de trouver P tel que :

$$\forall \underline{s} \in S.\ [\varepsilon(\underline{s}) \wedge P(\underline{s})] \Rightarrow [\forall \bar{s} \in S.\ t^*(\underline{s},\bar{s}) \Rightarrow \mu(\bar{s})]$$

On peut donc choisir $P(\underline{s})$ tel que :

$$[\varepsilon(\underline{s}) \wedge [\exists \bar{s} \in S.\ t^*(\underline{s},\bar{s}) \wedge \neg\mu(\bar{s})]] \Rightarrow \neg P(\underline{s})$$

c'est-à-dire comme la négation d'une condition nécessaire sur les états d'entrée pour qu'ils aient comme descendants un état où le contrôle est simultanément en L et L'. Cette condition s'obtient comme approximation supérieure du plus petit point fixe de

$$b(I)(s) = [(\exists s' \in S, a \in A.\ t_a(s,s') \wedge I(s')) \vee \neg\mu(s)]$$

□

4.3.2.5.3 Analyse d'invariance "avant-arrière"

L'analyse d'invariance "avant-arrière" consiste à trouver une approximation supérieure de l'ensemble des descendants des états initiaux (satisfaisant une condition initiale ϕ portant par exemple sur les états d'entrée), qui satisfont une condition δ (dérivée par exemple des déclarations) et sont ascendants des états finaux (satisfont une condition finale ψ portant par exemple sur les états de sortie):

$$\underset{\sim}{\Delta A}(\phi,\delta,\psi,\langle S,A,\Sigma\rangle) \in (S^2 \to \{tt,ff\})$$

$$\underset{\sim}{\Delta A}(\phi,\delta,\psi,\langle S,A,\Sigma\rangle)(\underline{s},s,\bar{s}) =$$

$$[\exists p \in \Sigma, i,j \in |p|.\ \phi(\underline{s}) \wedge p_0 = \underline{s} \wedge \delta(s) \wedge p_i = s \wedge \psi(\bar{s}) \wedge p_j = \bar{s} \wedge i \leq j]$$

Une première approximation supérieure consiste à raisonner sur le système de transition $\langle S,A,t,\varepsilon\rangle$ engendré par la sémantique $\langle S,A,\Sigma\rangle$. On a:

$$\underset{\sim}{\Delta A}(\phi,\delta,\psi,\langle S,A,\Sigma\rangle)(\underline{s},s,\bar{s}) \Rightarrow (\underset{\sim}{lfp}(F)(\underline{s},s) \wedge \delta(s) \wedge \underset{\sim}{lfp}(B)(s,\bar{s}))$$

où comme précédemment :

$$F(I)(\underline{s},\bar{s}) = [(\underline{s}=\bar{s} \wedge \phi(\underline{s})) \vee (\exists s' \in S, a \in A.\ I(\underline{s},s') \wedge t_a(s',\bar{s}))]$$

$$B(I)(\underline{s},\bar{s}) = [(\exists s' \in S, a \in A.\ t_a(\underline{s},s') \wedge I(s',\bar{s})) \vee (\bar{s}=\underline{s} \wedge \psi(\bar{s}))]$$

Soit $\langle \tilde{A}^{\tilde{\delta}}_{\delta}, \sqsubseteq\rangle$ une décomposition de $\langle A_\delta, \Rightarrow\rangle$ par la correspondance de Galois (α,γ) où $A_\delta = (S^2 \to \{tt,ff\})$. On pose $\tilde{F} = \alpha \circ F \circ \gamma$, $\tilde{B} = \alpha \circ B \circ \gamma$ et $\tilde{\delta} = \alpha(\delta)$. On définit l'opérateur Π sur $\langle \tilde{A}^{\tilde{\delta}}_{\delta}, \sqsubseteq\rangle$ tel que $\forall P,Q \in \tilde{A}^{\tilde{\delta}}_{\delta}.\ [\alpha(\gamma(P) \wedge \gamma(Q)) \sqsubseteq (P \sqcap Q)]$, et Δ est un opérateur de rétrécissement comme en 4.3.2.5.1. A la suite de Cousot-P [78], on peut calculer une approximation supérieure de

$$\alpha(\underset{\sim}{\Delta A}) \quad \text{où} \quad \underset{\sim}{\Delta A}(\underline{s},s,\bar{s}) = [\underset{\sim}{lfp}(F)(\underline{s},s) \wedge \delta(s) \wedge \underset{\sim}{lfp}(B)(s,\bar{s})]$$

comme limite de la suite décroissante finie $X^0,\ldots,X^k,\ldots$ telle que :

$\tilde{\delta} \sqsubseteq X^\circ$

$X^\circ \Delta Z^\circ = X^1$ où $Z^\circ \ni \mathrm{lfp}_{\tilde{\delta}}(f^\circ)$ et $f^\circ(Y) = x^\circ \sqcap F(Y)$

$X^1 \Delta Z^1 = X^2$ où $Z^1 \ni \mathrm{lfp}_{\tilde{\delta}}(f^1)$ et $f^1(Y) = x^1 \sqcap B(Y)$

...

$X^{2k} \Delta Z^{2k} = X^{2k+1}$ où $Z^{2k} \ni \mathrm{lfp}_{\tilde{\delta}}(f^{2k})$ et $f^{2k}(Y) = x^{2k} \sqcap F(Y)$

$X^{2k+1} \Delta Z^{2k+1} = X^{2k+2}$ où $Z^{2k+1} \ni \mathrm{lfp}_{\tilde{\delta}}(f^{2k+1})$ et $f^{2k+1}(Y) = x^{2k+1} \sqcap B(Y)$

...

Pour calculer les Z^k, on utilise une itération chaotique croissante avec élargissement puis si la solution obtenue n'est pas un point fixe on l'améliore par une itération chaotique décroissante avec rétrecissement. Bien entendu si le treillis $\langle \tilde{A}^\sharp, \sqsubseteq \rangle$ satisfait la condition de chaîne ascendante (respectivement descendante) on peut choisir $\nabla = \sqcup$ (respectivement $\Delta = \sqcap$).

Exemple 4.3.2.5.3-1

Si nous poursuivons l'analyse en avant du programme 2.8.2.3 donnée en exemple 4.3.2.5.1-1, par une analyse avant-arrière combinée partant de $\delta(n, p1, p2, p) = (n \in [0, hi] \wedge p1, p2, p \in [li, hi])$ (donnée par les déclarations du programme), nous obtenons les résultats suivants (pour le processus 1, en notant $\div$ la division entière et $\lfloor x = (x \div 2) \times 2$) :

	$k=1$			$k=2$			$k=3$		
	n	$p1$	$p2$	n	$p1$	$p2$	n	$p1$	$p2$
$D_{11}(2k)$	$[0,hi]$	$[li,hi]$	$[li,hi]$	$[0,hi]$	$[li,hi]$	$[1,1]$	$[2,hi]$	$[li,hi]$	$[1,hi \div 2]$
$D_{12}(2k)$	$[0,hi]$	$[1,1]$	$[li,hi]$	$[0,hi]$	$[1,1]$	$[1,1]$	$[2,hi]$	$[1,1]$	$[1,hi \div 2]$
$D_{13}(2k)$	$[2,hi]$	$[1,hi \div 2]$	$[li,hi]$	$[2,hi]$	$[1,hi \div 2]$	$[1,1]$	$[2,hi]$	$[1,hi \div 2]$	$[1,hi \div 2]$
$D_{14}(2k)$	$[1,hi-1]$	$[2,\lfloor hi]$	$[li,hi]$	$[1,hi-1]$	$[2,\lfloor hi]$	$[1,1]$	$[1,hi-1]$	$[2,\lfloor hi]$	$[1,hi \div 2]$
$D_{15}(2k)$	$[0,1]$	$[1,\lfloor hi]$	$[li,hi]$	$[0,1]$	$[1,\lfloor hi]$	$[1,1]$	$[1,1]$	$[2,hi]$	$[1,hi \div 2]$

	k=4			k=5		
	m	p1	p2	m	p1	p2
$D_{11}(2k)$	$\langle [1, hi-1], [li, hi], [2, \lfloor hi \rfloor] \rangle$			$\langle [0,1], [li, hi], [1, \lfloor hi \rfloor] \rangle$		
$D_{12}(2k)$	$\langle [1, hi-1], [1,1], [2, \lfloor hi \rfloor] \rangle$			$\langle [0,1], [1,1], [1, \lfloor hi \rfloor] \rangle$		
$D_{13}(2k)$	$\langle [1, hi-1], [1, hi \div 2], [2, \lfloor hi \rfloor] \rangle$			$\langle [1,1], [1, hi \div 2], [2, \lfloor hi \rfloor] \rangle$		
$D_{14}(2k)$	$\langle [0, hi-2], [2, \lfloor hi \rfloor], [2, \lfloor hi \rfloor] \rangle$			$\langle [0,1], [2, \lfloor hi \rfloor], [1, \lfloor hi \rfloor] \rangle$		
$D_{15}(2k)$	$\langle [0,1], [1, \lfloor hi \rfloor], [2, \lfloor hi \rfloor] \rangle$			$\langle [0,1], [1, \lfloor hi \rfloor], [1, \lfloor hi \rfloor] \rangle$		

On peut associer à chaque point ij du programme l'invariant $\chi\left(\bigcup_{k=1}^{m} D_{ij}(2k)\right)$, ce qui donne :

0: $\{m \in [0, hi]\}$

[[11: $\{m \in [0, hi] \wedge p1 \in [li, hi] \wedge p2 \in [li, hi]\}$

P1 := 1 ;

12: $\{m \in [0, hi] \wedge p1 \in [1,1] \wedge p2 \in [li, hi]\}$

<u>while</u> N>1 <u>do</u>

13: $\{m \in [1, hi] \wedge p1 \in [1, hi \div 2] \wedge p2 \in [li, hi]\}$

(* N := N-1 ; P1 := 2×P1 *) ;

14: $\{m \in [0, hi-1] \wedge p1 \in [2, \lfloor hi \rfloor] \wedge p2 \in [li, hi]\}$

<u>od</u> ;

15: $\{m \in [0,1] \wedge p1 \in [1, \lfloor hi \rfloor] \wedge p2 \in [li, hi]\}$

||

21: $\{m \in [0, hi] \wedge p1 \in [li, hi] \wedge p2 \in [li, hi]\}$

P2 := 1 ;

22: $\{m \in [0, hi] \wedge p1 \in [1,1] \wedge p2 \in [li, hi]\}$

<u>while</u> N>1 <u>do</u>

23: $\{m \in [1, hi] \wedge p1 \in [li, hi] \wedge p2 \in [1, hi \div 2]\}$

(* N := N-1 ; P2 := 2×P2 *) ;

24: $\{m \in [0, hi-1] \wedge p1 \in [li, hi] \wedge p2 \in [2, \lfloor hi \rfloor]\}$

<u>od</u> ;

25: $\{m \in [0,1] \wedge p1 \in [li, hi] \wedge p2 \in [1, \lfloor hi \rfloor]\}$

]] ;

1: $\{n \in [0,1] \wedge p1 \in [1, \lfloor hi \rfloor] \wedge p2 \in [1, \lfloor hi \rfloor]\}$

if N=0 then P:= P1×P2 else P:= 2×P1×P2 fi ;

2: $\{n \in [0,1] \wedge p1 \in [1, \lfloor hi \rfloor] \wedge p2 \in [1, \lfloor hi \rfloor] \wedge p \in [1, hi]\}$

Ces résultats permettent de placer des tests qui doivent être vérifiés pour éviter les erreurs à l'exécution (mais en moins grand nombre que ne le ferait un compilateur n'utilisant que les informations données par les déclarations) ainsi que des tests qu'il est necessaire (mais en général pas suffisant) de vérifier pourque l'exécution du programme se termine sans erreurs à l'exécution et en ne passant que par des états satisfaisant la condition δ (ces derniers ne sont pas placés par un compilateur classique, même pour les tests liés aux déclarations puisque les test à l'exécution sont généralement placés au moment de l'affectation aux variables et pas au moment de leur utilisation). Pour les tests de la premiere sorte, on trouve :

- $p1 \leqslant hi \div 2$ au point 13
- $p2 \leqslant hi \div 2$ au point 23
- $p1 \leqslant hi \div p2$ au point 1 quand N=0
  $p1 \leqslant (hi \div 2) \div p2$ au point 1 quand N≠0

(Un compilateur classique placerait des tests inutiles comme $n-1 \geqslant 0$ aux points 13 et 23 ou des tests plus complexes comme pour tester que $li \leqslant p1 \times p2 \leqslant hi$ ou $li \leqslant 2 \times p1 \times p2 \leqslant hi$ au point 1 puisque le signe de p1 et p2 n'est pas connu). Pour les tests de la seconde sorte, on trouve :

- $n \geqslant 0$ au point 0

(ce test figurerait dans un compilateur classique après la commande de lecture de la variable N). L'interêt d'une analyse en arrière pour introduire des tests nécessaires pour une terminaison normale est mieux illustré par l'exemple suivant (tiré de Cousot-P [78], p.(5)-53) :

```
        const li = ... ;   { plus petit entier, li<0}
              hi = ... ;   { plus grand entier, hi>1000}
        type integer = li..hi ;
        var N, K, I, J : integer
            T: array [0..1000] of integer ;
1:      {m,k,i,j ∈ [li,hi]}
        read (N);
2:      {m∈[0,1000] ∧ k,i,j∈[li,hi]}
        K:=0;
3:      {m∈[0,1000] ∧ k∈[0,0] ∧ i,j∈[li,hi]}
        while K <= N do
4:          {m,k∈[0,1000] ∧ i,j∈[li,hi]}
            read (T[K]);
5:          {m,k∈[0,1000] ∧ i,j∈[li,hi]}
            K:=K+1;
6:          {m∈[0,1000] ∧ k∈[1,1001] ∧ i,j∈[li,hi]}
        od;
7:      {m∈[0,1000] ∧ k∈[0,1001] ∧ i,j∈[li,hi]}
        I:=N;
8:      {i,j∈[0,1000] ∧ k∈[0,1001] ∧ j∈[li,hi]}
        while I<>0 do
9:          {m∈[0,1000] ∧ i∈[1,1000] ∧ k,j∈[li,hi]}
            J:=0;
10:         {m∈[0,1000] ∧ i∈[1,1000] ∧ j∈[0,0] ∧ k∈[li,hi]}
            while J<>I do
11:             {m∈[0,1000] ∧ i∈[1,1000] ∧ j∈[0,999] ∧ k∈[li,hi]}
                if T[J] <= T[J+1] then
12:                 {m∈[0,1000] ∧ i∈[1,1000] ∧ j∈[0,999] ∧ k∈[li,hi]}
                    K:=T[J]; T[J]:=T[J+1]; T[J+1]:=K;
13:                 {m∈[0,1000] ∧ i∈[1,1000] ∧ j∈[0,999] ∧ k∈[li,hi]}
                fi;
```

14: $\{m \in [0,1000] \wedge i \in [1,1000] \wedge i \in [0,999] \wedge k \in [li, ri]\}$

J := J+1;

15: $\{m \in [0,1000] \wedge i,j \in [1,1000] \wedge k \in [li, ri]\}$

od;

16: $\{m \in [0,1000] \wedge i,j \in [1,1000] \wedge k \in [li, ri]\}$

I := I-1;

17: $\{m \in [0,1000] \wedge i,j \in [1,1000] \wedge k \in [li, ri]\}$

od;

18: $\{m \in [0,1000] \wedge i \in [0,0] \wedge j \in [1,1000] \wedge k \in [li, ri]\}$

Cette analyse conduit à introduire le test $0 \leq m \leq 1000$ au point 2 (pour garantir que l'exécution se termine sans erreurs à l'exécution, ce test ne serait évidemment pas introduit par un compilateur classique). Dans ces conditions, il faut également introduire le test $j \leq 999$ au point 11 (ce test étant en fait inutile).

□

4.4 REFERENCES

APT K.R., FRANCEZ N., DE ROEVER W.P. [80], " A proof system for communicating séquential processes", TOPLAS 2, 3(1980), 359-385.

ASHCROFT E.A. [75], "Proving assertions about parallel programs", J. of Comp. and System Science, 10(1975), 110-135.

ASHCROFT E.A., MANNA Z. [70], " Formalization of properties of parallel programs, Machine Intelligence, 6(1970), 17-41.

COUSOT P. [77], "Asynchronous iterative methods for solving a fixed point system of monotone equations in a complete lattice", Rapport de Recherche N°88, IMAG, Université de Grenoble, (Mars 1978).

COUSOT P. [78], "Méthodes itératives de construction et d'approximation de points fixes d'opérateurs monotones sur un treillis, analyse sémantique des programmes, Thèse d'Etat, Université de Grenoble, (Mars 1978).

COUSOT P. [79], "Analysis of the behavior of dynamic discrete systems", Rapport de Recherche n°161, IMAG, Université de Grenoble, (Jan. 1979).

COUSOT P. [81], "Semantic foundations of program analysis", dans "Program flow analysis, theory and applications", S.S. Muchnick & N.J. Jones (eds.), Prentice-Hall, (1981), 303-342.

COUSOT R. [81], " Proving invariance properties of parallel programs by backward induction", Rapport de Recherche CRIN-81-P026, (1981).

COUSOT P., COUSOT R. [76], " Static determination of dynamic properties of programs", Proc. 2nd Int. Symp. on Programming, Paris, Dunod, (Avril 1976), 106-130.

COUSOT P., COUSOT R. [77a], " Abstract interpretation : a unified lattice model for static analysis of programs by construction or approximation of fixpoints ", Conf. Rec. of the 4th ACM Symp. on Principles of Programming Languages, Los Angeles, (Jan. 1977), 238-252.

COUSOT P., COUSOT R. [77b], " Static determination of dynamic properties of generalized type unions", ACM Conf. on Language Design for Reliable Software, Raleigh, SIGPLAN Notices 12, 3(1977), 77-94.

COUSOT P., COUSOT R. [77c], "Static determination of dynamic properties of recursive procedures", Conf. on Formal Description of Programming Concepts, St-Andrews, Canada, North-Holland Pub. Co., (1977), 237-277.

COUSOT P., COUSOT R. [77d], "Automatic synthesis of optimal invariant assertions : mathematical foundations", Proc. ACM Symp. on Artificial Intelligence & Programming Languages, Rochester, SIGPLAN Notices 12, 8(1977), 1-12.

COUSOT P., COUSOT R. [79a], "Systematic design of program analysis frameworks", Conf. Rec. of the 6th ACM Symp. on Principles of Programming Languages, San Antonio, Texas, (1979), 269-282.

COUSOT P., COUSOT R. [79b], " Constructive versions of Tarski's fixed point theorems", Pacific Journal of Math., vol. 82, n°1, (1979), 43-57.

COUSOT P., COUSOT R. [80a], "Semantic analysis of communicating sequential processes", Automata, Languages and Programming, 7th Colloq., Lecture Notes in Computer Sci. 85, Springer-Verlag, (1980), 119-133.

COUSOT P., COUSOT R. [80b], " Constructing program invariance proof methods" Proc. Int. Workshop on Program Construction, INRIA Ed., Tome 1, (1980).

COUSOT P., COUSOT R. [80c], "Reasoning about program invariance proof methods", Rapport de Recherche CRIN-80-P050, (1980).

COUSOT P., COUSOT R. [82a], "Induction principles for proving invariance properties of programs, dans "Tools and Notions for Program Construction", (D. Neel Ed.), Cambridge University Press, (1982), 75-119.

COUSOT P., COUSOT R. [82b], "'A la Floyd' induction principles for proving inevitability properties of programs", Rapport de recherche LRIM-82-04, à paraître dans "Algebraic Methods in Programming", (M. Nivat & J. Reynolds, eds.), Cambridge University Press.

COUSOT P., COUSOT R. [84], "Invariance proof methods and analysis techniques for parallel programs", dans "Automatic program construction techniques", (A. Biermann et al., eds.), MAC MILLAN, (1984), 243-271.

COUSOT P., HALBWACHS N. [78], "Automatic discovery of linear restraints among variables of a program", Conf. Rec. of the 5th ACM Symp. on Principles of Programming Languages, Tucson, Arizona, (1978), 84-97.

DIJKSTRA E.W. [82], "Selected writings on computing: a personal perspective", Springer-Verlag, (1982).

FLOYD R.W. [67], "Assigning meaning to programs", Proc. Symp. in Applied Math., AMS, Providence, RI, (1967), 19-32.

HOARE C.A.R. [69], "An axiomatic basis for computer programming", CACM 12, 10 (1969), 576-580, 583.

HOARE C.A.R. [78], "Toward a theory of parallel programming", dans "Operating Systems Techniques", (Hoare & Perott eds.), Academic Press, (1978).

HOARE C.A.R. [75], "Parallel programming: an axiomatic approach", Computer Languages, 1 (1975), 151-160.

HOARE C.A.R. [78], "Communicating sequential processes", CACM 21, 8 (1978), 666-677.

HOWARD J.H. [76], "Proving monitors", CACM 19, 5 (1976), 273-279.

KELLER R.M. [76], "Formal verification of parallel programs", CACM 19, 7 (1976), 371-384.

LAMPORT L. [77], "Proving the correctnes of multiprocess programs", IEEE Trans. on Soft. Eng., SE3, 2 (1977), 125-143.

LAMPORT L. [80], "The "Hoare Logic" of concurrent programs", Acta Informatica 14, (1980), 21-37.

LEVIN G.M. [79], "A proof technique for communicating sequential processes (with an example)", TR79-401, Comp. Sci. Dept, Cornell U., N.Y., (1979).

MANNA Z. [70], "Mathematical theory of partial correctnes", JCSS 5, 3 (1970), 238-253.

MAZURKIEWICZ A [77], "Concurrent program schemes and their interpretation", Dept. Comp. Sci, Aarhus U., Danemark, DAIMI-PB-78, (1977).

MISRA J. [78], "Some aspects of the verification of loop computations", IEEE Trans. on Soft. Eng., SE-4, 6 (1978), 478-486.

MORRIS J.H., WEGBREIT B. [77], "Subgoal induction", CACM 20, 4 (1977), 209-222.

NAUR P. [66], "Proof of algorithms by general snapshots", BIT 6, (1966), 310-316.

NEWTON G. [75], "Proving properties of interacting processes", Acta Informatica, 4 (1975), 117-126.

OWICKI S., GRIES D. [76a], "An axiomatic proof technique for parallel programs I", Acta Informatica, 6 (1976), 319-340

OWICKI S., GRIES D. [76b], "Verifying properties of parallel programs: an axiomatic approach", CACM 19, 5 (1976), 279-285.

RICART G., AGRAWALA A.K. [81], "An optimal algorithm for mutual exclusion in computer networks", CACM 24, 1 (1981), 9-17.

TARSKI A. [55], "A lattice theoretical fixpoint theorem and its applications", Pacific Journal of Math., 5 (1955), 285-310.

The first part of Radhia Cousot's thesis is available at
https://pcousot.github.io/publications/RadhiaCousotTheseEsSciences.PDF

The second part of Radhia Cousot's thesis is available at
https://pcousot.github.io/publications/RadhiaCousotTheseEsSciences2.PDF

The third part of Radhia Cousot's thesis is available at
https://pcousot.github.io/publications/RadhiaCousotTheseEsSciences3.PDF