Extension de la logique de séparation avec des pointfixes et une substitution retardée

Élodie-Jane Sims
Elodie-Jane.Sims@polytechnique.fr

Polytechnique/KSU

Séminaire 68 N Q R T (IRISA)
24 novembre 2004

Motivations

- But: analyses de pointeurs: vérifier les problèmes de déréférencements, d'alias,...
- Les logiques de fragmentation permettent de décrire des propriétés de la mémoire, e.g.
  - x pointe vers une liste de [1;2;3]
    $\exists x_2, x_3. (x \hookrightarrow 1, x_2) * (x_2 \hookrightarrow 2, x_3) * (x_3 \hookrightarrow 3, \texttt{nil})$
  - x et y sont des pointeurs aliasés
    $x = y \wedge \exists x_1, x_2. (x \hookrightarrow x_1, x_2)$
  - Partitionnement: x et y appartiennent à deux parties disjointes du tas qui n'ont pas de pointeurs de l'une à l'autre...

➤ Nous voulons utiliser cette logique, pour des analyses modulaires,
comme langage d'interface entre analyses,
ou comme langage intermediaire entre les programmes et d'autres analyses.

Résultats

- Jusqu'à présent, les formules récursives ne pouvaient pas être exprimées dans cette logique, pas plus que les pre-conditions (wlp) et post-conditions (sp) pour les boucles `while`

- Donc nous avons ajouté des pointfixes et une substitution retardée, et avons exprimé les wlp et sp pour toute commande et toute formule et prouvé leur correction.

Plan

- Programmes
- La logique étendue: $BI^{\mu\nu}$
- Analyse arrière: wlp,
  Analyse avant: sp
- Conclusions
- Triplets

Le domaine *Mémoire*

$$\begin{array}{rcll} Val & = & Int \cup Bool \cup Atoms \cup Loc & Valeurs \\ S & = & Var \rightharpoonup Val & Piles \\ H & = & Loc \rightharpoonup Val \times Val & Tas \\ Mémoire & = & S \times H & \end{array}$$

Nous utiliserons s, h ou m comme nom de mémoire.

Rq: les piles peuvent être des fonctions partielles

Commandes

$$
\begin{array}{rcl}
C & ::= & x := E \\
& | & x := E.i \\
& | & E.i := E' \\
& | & x := \texttt{cons}(E_1, E_2) \\
& | & \texttt{dispose}(E) \\
& | & C_1 ; C_2 \\
& | & \texttt{if}\ E\ \texttt{then}\ C_1\ \texttt{else}\ C_2 \\
& | & \texttt{while}\ E\ \texttt{do}\ C_1 \\
& | & \texttt{skip}
\end{array}
$$

$$
E \quad ::= \quad x \mid n \mid \texttt{nil} \mid True \mid False \mid E_1\ op\ E_2
$$

$$
n \in \mathbb{Z},\ i \in \{1, 2\}
$$

Sémantiques opérationnelles des commandes

$$C, m \rightsquigarrow C', m'$$

$$C, m \rightsquigarrow m'$$

Exemple pour cons

$$\frac{l \in Loc, \quad l \notin dom(h) \cup range(h) \cup range(s), \quad v_1 = [\![E_1]\!]^s, v_2 = [\![E_2]\!]^s}{x := \texttt{cons}(E_1, E_2), s, h \rightsquigarrow [s|x \rightarrow l], [h|l \rightarrow \langle v_1, v_2 \rangle]}$$

La logique étendue: $BI^{\mu\nu}$

Connecteurs	*classiques*				
$E = E'$		$\vert$	$\mathtt{false}$		
$\vert$ $P \Rightarrow Q$		$\vert$	$\exists x.P$		
Connecteurs	*spatiaux*				
$\vert$ $\mathtt{emp}$	Tas vide	$\vert$	$E \mapsto E_1, E_2$	Pointe vers	
$\vert$ $P * Q$	Conj. spatiale	$\vert$	$P \mathrel{-\!\!*} Q$	Imp. spatiale	
Connecteurs	*de pointfixes*				
$\vert$ X_v	Variable de formules	$\vert$	$P[E/x]$	Substitution retardée	
$\vert$ $\boldsymbol{\nu} X_v.P$	Plus grand pointfixe	$\vert$	$\boldsymbol{\mu} X_v.P$	Plus petit pointfixe	

$Var_v = \{X_v, Y_v, ...\}$ un ensemble infini de variables de formules

Sémantiques opérationnelles des formules

$$\rho : Var_v \rightharpoonup \mathcal{P}(\textit{Mémoire}) \text{ : environnement}$$

$$[\![P]\!]_\rho \in \mathcal{P}(\textit{Mémoire}) \text{ : sémantiques}$$

$$m \models P \text{ ssi } m \in [\![P]\!]_\emptyset$$

$$P \equiv Q \text{ ssi } \forall \rho.([\![P]\!]_\rho = [\![Q]\!]_\rho) \vee ([\![P]\!]_\rho \text{ et } [\![Q]\!]_\rho \text{ n'existent pas})$$

Sémantiques des connecteurs classiques

$$\begin{aligned}
[\![E = E']\!]_\rho &= \{s, h \mid [\![E]\!]^s = [\![E']\!]^s\} \\
[\![\texttt{false}]\!]_\rho &= \emptyset \\
[\![P \Rightarrow Q]\!]_\rho &= (\mathit{Mémoire} \setminus [\![P]\!]_\rho) \cup [\![Q]\!]_\rho \\
[\![\exists x.\ P]\!]_\rho &= \{s, h \mid \exists v \in \mathit{Val}.\ [s \mid x \to v], h \in [\![P]\!]_\rho\}
\end{aligned}$$

Sémantiques des connecteurs spatiaux

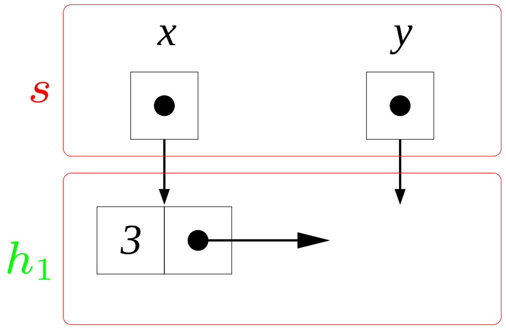
$$\begin{array}{rcl}
[\![\texttt{emp}]\!]_\rho & = & \{s,h \mid dom(h)=\emptyset\} \\
[\![E \mapsto E_1, E_2]\!]_\rho & = & \{s,h \mid dom(h)\boldsymbol{=}\ \boldsymbol{\{}[\![E]\!]^s\boldsymbol{\}} \text{ et } h([\![E]\!]^s)=\langle [\![E_1]\!]^s, [\![E_2]\!]^s\rangle\} \\
[\![P * Q]\!]_\rho & = & \{s, h_0 \cdot h_1 \mid h_0 \sharp h_1,\ \ s,h_0 \in [\![P]\!]_\rho \text{ et } s,h_1 \in [\![Q]\!]_\rho\} \\
[\![P \mathrel{-\!\!*} Q]\!]_\rho & = & \{s,h \mid \forall \boldsymbol{h'}, \text{ if } h\sharp h' \text{ et } s,h' \in [\![P]\!]_\rho \text{ alors } s, h{\cdot}h' \in [\![Q]\!]_\rho\}
\end{array}$$

$h\sharp h'$: $dom(h)$ et $dom(h')$ sont disjoints
$h \cdot h'$: union des tas h et h' disjoints

Exemples de formules

Ex. 1

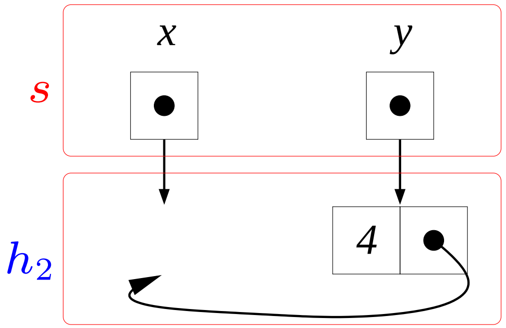
$$s = [x \longrightarrow l_1, y \longrightarrow l_2]$$
$$h_1 = [l_1 \longrightarrow \langle 3, l_2 \rangle]$$



$$\begin{array}{ll} \models & (x \mapsto 3, y) \\ \models & (y \mapsto 4, x) \mathrel{-\!\!*} \\ & ((x \mapsto 3, y) * (y \mapsto 4, x)) \end{array}$$

Ex. 2

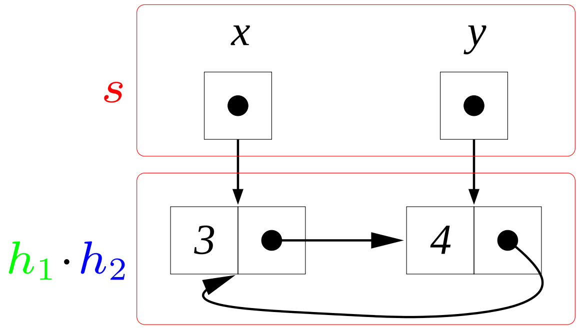
$$s = [x \longrightarrow l_1, y \longrightarrow l_2]$$
$$h_2 = [l_2 \longrightarrow \langle 4, l_1 \rangle]$$



$$\models (y \mapsto 4, x)$$

Ex. 3

$$s = [x \longrightarrow l_1, y \longrightarrow l_2]$$
$$h_1 \cdot h_2 = \begin{bmatrix} l_1 \longrightarrow \langle 3, l_2 \rangle, \\ l_2 \longrightarrow \langle 4, l_1 \rangle \end{bmatrix}$$



$$\models (x \mapsto 3, y) * (y \mapsto 4, x)$$
$$\not\models (x \mapsto 3, y) \wedge (y \mapsto 4, x)$$

Différence entre $*$ et $\wedge$

- $(x \mapsto 1, 2) * (x \mapsto 1, 2) \equiv \texttt{false}$
- $(x \mapsto 1, 2) \wedge (x \mapsto 1, 2) \equiv (x \mapsto 1, 2)$
- $(x \mapsto 1, 2) \wedge \neg(x \mapsto 1, 2) \equiv \texttt{false}$
- $(x \mapsto 1, 2) * \neg(x \mapsto 1, 2) \equiv (x \mapsto 1, 2) * \texttt{true}$

Sémantiques des connecteurs de pointfixes

$$
\begin{array}{rcl}
[\![X_v]\!]_\rho & = & \rho(X_v) \text{ if } X_v \in dom(\rho) \\[1em]
[\![\mu X_v.P]\!]_\rho & = & \mathrm{lfp}_\emptyset^\subseteq \lambda Y.\ [\![P]\!]_{[\rho|X_v \to Y]} \\[1em]
[\![\nu X_v.P]\!]_\rho & = & \mathrm{gfp}_\emptyset^\subseteq \lambda Y.\ [\![P]\!]_{[\rho|X_v \to Y]} \\[1em]
[\![P[E/x]]\!]_\rho & = & \left\{ s, h \mid \begin{array}{l} [\![E]\!]^s \text{ existe et} \\ [s \mid x \to [\![E]\!]^s], h \in [\![P]\!]_\rho \end{array} \right\}
\end{array}
$$

Remarques:

- $[\![\ ]\!]$ peut ne pas être défini: e.g. $[\![X_v]\!]_\emptyset$, $[\![\mu X_v.\neg X_v]\!]_\rho$
- $[\![\texttt{true}[y/x]]\!] = \{s, h \mid [\![y]\!]^s \text{ existe }\}$ connecteur de substitution retardée
  $[\![\texttt{true}\{y/x\}]\!] = [\![\texttt{true}]\!] = \mathit{Mémoire}$ Capture avoiding substitution

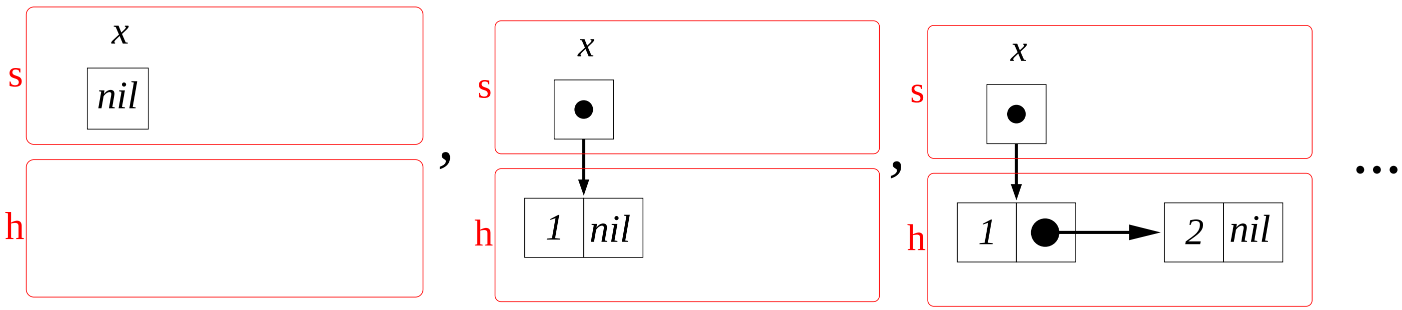
Exemple de formules récursives : les listes

"x pointe vers une liste finie d'entiers non-cyclique"

$$\texttt{nclist}(x) \triangleq \boldsymbol{\mu X_v}.\ (x = \texttt{nil}) \vee \exists x_1, x_2.(\texttt{isint}(x_1) \wedge (x \mapsto x_1, x_2 * \boldsymbol{X_v[x_2/x]}))$$

$$\texttt{isint}(x) \triangleq \exists n. n = x + 1$$

À noter, la combinaison des pointfixes et de la substitution retardée pour écrire des définitions récursives
"$\texttt{nclist}(x) = (x = \texttt{nil}) \vee \exists x_1, x_2.\texttt{isint}(x_1) \wedge (x \mapsto x_1, x_2 * \texttt{nclist}(x_2))$"



Formules des arbres

"x pointes vers un arbre d'entiers"

$$\mathtt{tree}(x) \triangleq \mu X_v. \begin{array}{l} (x = \mathtt{nil}) \vee \exists x_v, x_l, x_r, x'.\mathtt{isint}(x_v) \wedge \\ (x \mapsto x_v, x' \ * \ x' \mapsto x_l, x_r \ * \ X_v[x_l/x] \ * \ X_v[x_r/x]) \end{array}$$

Quelques propriétés de la logique

- $[\ /\ ]$ n'est pas $\{\ /\ \}$
  avec
  $\{\ /\ \}$: capture avoiding substitution (subsitution habitutelle)
  $[\ /\ ]$: connecteur de substitution retardée

- Les théorèmes de déroulement sont valides
  $\mu X_v.\ P \equiv P\{\mu X_v.\ P/X_v\}$
  $\nu X_v.\ P \equiv P\{\nu X_v.\ P/X_v\}$

- $\{\ /\ \}$: le théroème habituel de renommage des variables n'est pas valide (voir transparents suivants)

- quelques simplifications sur $[\ /\ ]$ (voir transparents suivants)

$\{\ /\ \}$: pas de théorème de renommage de variables

$$\exists y.P \not\equiv \exists z.P\{z/y\}$$

avec $z \notin Var(P)$ *(quand* $y \neq z$*)*

Contre exemples:

- $$\begin{array}{ccc} [\![\nu X_v.y = 3 \wedge \exists \boldsymbol{y}.(X_v \wedge \boldsymbol{y} = 5)]\!]_\emptyset & \not\equiv & [\![\nu X_v.y = 3 \wedge \exists \boldsymbol{z}.(X_v \wedge \boldsymbol{z} = 5)]\!]_\emptyset \\ = & & = \\ \emptyset & & [\![y = 3]\!]_\emptyset \end{array}$$

- $$\begin{array}{ccc} [\![\exists \boldsymbol{y}.\nu X_v.\boldsymbol{y} = 3 \wedge \exists y.(X_v \wedge y = 5)]\!]_\emptyset & \not\equiv & [\![\exists \boldsymbol{z}.\nu X_v.\boldsymbol{z} = 3 \wedge \exists y.(X_v \wedge y = 5)]\!]_\emptyset \\ = & & = \\ \emptyset & & \textit{Mémoire} \end{array}$$

Détails de la sémantique de $\nu X_v.y = 3 \wedge \exists y.(X_v \wedge y = 5)$

$$
\begin{array}{ll}
& [\![\nu X_v.y = 3 \wedge \exists y.(X_v \wedge y = 5)]\!]_\emptyset \\
= & \mathrm{gfp}_\emptyset^\subseteq \lambda Y.\ [\![y = 3 \wedge \exists y.(X_v \wedge y = 5)]\!]_{[X_v \to Y]} \\
= & \mathrm{gfp}_\emptyset^\subseteq \lambda Y.\ [\![y = 3]\!]_{[X_v \to Y]} \cap [\![\exists y.(X_v \wedge y = 5)]\!]_{[X_v \to Y]} \\
= & \mathrm{gfp}_\emptyset^\subseteq \lambda Y.\ \{s, h \mid s(y) = 3\} \cap \{s, h \mid \exists v.[s \mid y \to v], h \in [\![X_v \wedge y = 5]\!]_{[X_v \to Y]}\} \\
= & \mathrm{gfp}_\emptyset^\subseteq \lambda Y.\ \{s, h \mid s(y) = 3\} \cap \{s, h \mid \exists v.[s \mid y \to v], h \in Y \wedge [s \mid y \to v](y) = 5\} \\
= & \mathrm{gfp}_\emptyset^\subseteq \lambda Y.\ \{s, h \mid s(y) = 3\} \cap \{s, h \mid [s \mid y \to 5], h \in Y\} \\
= & \emptyset
\end{array}
$$

Détails de la sémantique de $\nu X_v.y = 3 \wedge \exists z.(X_v \wedge z = 5)$

$$
\begin{array}{ll}
 & [\![\nu X_v.y = 3 \wedge \exists z.(X_v \wedge z = 5)]\!]_\emptyset \\
= & \mathrm{gfp}_\emptyset^\subseteq \lambda Y.\ [\![y = 3 \wedge \exists z.(X_v \wedge z = 5)]\!]_{[X_v \to Y]} \\
= & \mathrm{gfp}_\emptyset^\subseteq \lambda Y.\ [\![y = 3]\!]_{[X_v \to Y]} \cap [\![\exists z.(X_v \wedge z = 5)]\!]_{[X_v \to Y]} \\
= & \mathrm{gfp}_\emptyset^\subseteq \lambda Y.\ \{s, h \mid s(y) = 3\} \cap \{s, h \mid \exists v.[s \mid z \to v], h \in [\![X_v \wedge z = 5]\!]_{[X_v \to Y]}\} \\
= & \mathrm{gfp}_\emptyset^\subseteq \lambda Y.\ \{s, h \mid s(y) = 3\} \cap \{s, h \mid \exists v.[s \mid z \to v], h \in Y \wedge [s \mid z \to v](z) = 5\} \\
= & \mathrm{gfp}_\emptyset^\subseteq \lambda Y.\ \{s, h \mid s(y) = 3\} \cap \{s, h \mid [s \mid z \to 5], h \in Y\} \\
= & \{s, h \mid s(y) = 3\}
\end{array}
$$

Définition de la substitution totale

$\{[\;/\;]\}$: subsitution de variable totale

$P\{[z/y]\}$ est P dans lequel tout y est remplacé par z, où qu'il soit

par exemple:

$$(\exists y.P)\{[z/y]\} \triangleq \exists z.(P\{[z/y]\})$$

$$(P[E/x])\{[z/y]\} \triangleq (P\{[z/y]\})[E\{z/y\}/x\{z/y\}]$$

Théorème de renommage de variable pour $BI^{\mu\nu}$

Si

- P est v-clos (les variables dans Var_v sont toutes closes par μ ou ν)
- $z \notin Var(P)$
- $y \notin FV(P)$

alors

$$P \equiv P\{[z/y]\}$$

en particulier $\exists y.P \equiv \exists z.(P\{[z/y]\})$

Equivalences pour $[\ /\ ]$

– Si P ne contient pas de $\mu, \nu, X_v, [\ /\ ]$ alors

$$P[E/x] \equiv P\{E/x\} \wedge is(E)$$

en particulier $(\exists x.P)[E/x] \equiv (\exists x.P) \wedge is(E)$.

– Si $\left[\begin{array}{l} P \text{ est } v\text{-clos} \\ x_1 \notin Var(E) \\ x_1 \neq x_2 \end{array}\right.$ alors $(\exists x_1.P)[E/x_2] \equiv \exists x_1.(P[E/x_2])$

– $(A \vee C)[E/x] \equiv (A[E/x]) \vee (C[E/x])$

– Si $y \notin Var(P)$ alors $\begin{array}{l}(\mu X_v.P)[y/x] \equiv (\mu X_v.P\{[y/x]\}) \wedge is(y)\\(\nu X_v.P)[y/x] \equiv (\nu X_v.P\{[y/x]\}) \wedge is(y)\end{array}$

Pour comprendre la dernière règle, prenons le point de vue du programme qui consiste à voir

- les pointfixes comme des boucles `while`
- les $[\ /\ ]$ comme des assignements

C	$wlp(\texttt{true}, C)$
$x := w;$ `while` $x = y$ `do` $x := x + 1$	$(\nu X_v.(x \neq y) \vee ((x = y) \wedge X_v[x + 1/x]))[w/x]$
$\equiv$	$\equiv$
`while` $w = y$ `do` $w := w + 1$	$(\nu X_v.(w \neq y) \vee ((w = y) \wedge X_v[w + 1/w])) \wedge is(w)$

Analyse arrière: wlp

wlp : plus faible precondition libérale, telle que

$$[\![wlp(P,C)]\!]_\emptyset = \left\{ m \mid \begin{array}{l} - C, m \text{ ne peut pas conduire à une erreur} \\ - \{m' \mid C, m \leadsto^* m'\} \subseteq [\![P]\!]_\emptyset \end{array} \right\}$$

wlp est exprimée pour tout P et tout C

- $wlp(P, x := E) = P[E/x]$
- $wlp(P, \texttt{while } E \texttt{ do } C) = \boldsymbol{\nu X_v}.\ ((E = \texttt{true} \wedge wlp(\boldsymbol{X_v}, C)) \vee (E = \texttt{false} \wedge P))$

Remarque: $[\![wlp(\texttt{true}, C)]\!]_\emptyset = \{m \mid C, m$ ne peut pas conduire à une erreur$\}$

Analyse avant: sp

sp : plus forte postcondition, telle que

$$[\![sp(P, C)]\!]_\emptyset = \{m' \mid \exists m \in [\![P]\!]_\emptyset.\ C, m \rightsquigarrow^* m'\}$$

sp est exprimé pour tout P et tout C

- $sp(P, x := E) = \exists x'.\ P[x'/x] \wedge x = E\{x'/x\}$ avec $x' \notin FV(E, P)$
- $sp(P, \texttt{while}\ E\ \texttt{do}\ C) = (E = \texttt{false}) \wedge (\boldsymbol{\mu X_v}.sp(\boldsymbol{X_v} \wedge E = \texttt{true}, C) \vee P)$

Conclusions

- $\surd$ Nous avons étendu la logique de fragmentation pour permettre d'exprimer les formules recursives, et de pouvoir les utiliser pour instancier les règles de triplets existantes et nouvelles

- $\surd$ Nous avons une analyse arrière (wlp) et avant (sp) avec la preuve de leurs corrections pour toute formule et toute commande, en particulier pour les boucles `while`

- ➠ L'utilisation de la logique de fragmentation comme langage d'interface ou intermédiaire est du travail en cours... (nous avons déjà fait un petit essai avec une analyse de partitionnement)

Autres options pour la logique de fragmentation

- les piles sont des fonctions totales
- arithmétique de pointeurs
- logique intuitioniste (une formule valabe pour une mémoire, est valable pour toute mémoire avec un tas étendu)
- pour les points fixes, on aurait pu mettre des fonctions pour paramétrer les variables de formules

Triplets

$\{P\}C\{P'\}$ ssi

$\forall m$ si $m \models P$ alors
- C peut être exécuté à partir de m sans erreur
- si $C, m \leadsto^* m'$ alors $m' \models P'$

Cette définition diffère de celle habituelle des triplets de Hoare.
(Si $m \models P$ et $C, m \leadsto^* m'$ alors $m' \models P'$)

En particulier avec cette définition, si $\{P\}C\{\texttt{true}\}$, alors C peut être exécuté sans erreur depuis toute mémoire satisfaisant P.

wlp et sp comme triplets

Nous avons,

$$\forall C.\forall P.\ \{wlp(P, C)\}C\{P\} \text{ vrai}$$

Mais nous n'avons pas

$$\forall C.\forall P.\ \{P\}C\{sp(P, C)\} \text{ vrai}$$

parce que parfois

$$\nexists Q.\ \{P\}C\{Q\} \text{ true}$$

ex : $\{\mathtt{true}\}x := \mathtt{nil}; x.1 := 3\{?\}$

Nous avons par contre $\forall C.\forall P.\ \{P \wedge wlp(\mathtt{true}, C)\}C\{sp(P, C)\}$ vrai

car $wlp(\mathtt{true}, C)$ est la condition de sûreté.

Axiomes de Reynolds

Séquence

$$\frac{\{P\}C\{Q\} \quad \{Q\}C'\{R\}}{\{P\}C;C'\{R\}}$$

Conséquence

$$\frac{P \models P' \quad \{P'\}C\{R'\} \quad R' \models R}{\{P\}C\{R\}}$$

$P \models Q$ ssi $\forall m \;\; si \;\; m \models P \;\; alors \;\; m \models Q$

Règles de déduction pour $\models$ *(pour axiome Conséquence)*

- Les règles de la logique classique.

- $\dfrac{P' \models P \quad Q' \models Q}{P' * Q' \models P * Q}$

- $\dfrac{R * P \models Q}{R \models P \mathrel{-\!\!*} Q}$

- $\dfrac{R \models P \mathrel{-\!\!*} Q \quad R' \models P}{R * R' \models Q}$

- Pas affaiblissement $(P * Q \not\models P)$, Pas contraction $(P \not\models P * P)$ car on a une notion de taille du tas avec $*$

Axiome d'extension du tas

$$\boxed{\frac{\{P\}C\{Q\}}{\{P * R\}C\{Q * R\}}}$$

$$ModifiesOnly(C) \cap free(R) = \emptyset$$

$ModifiesOnly(C)$: ens. var. apparaissant seules à gauche de $:=$ dans C

Pourquoi la restriction est nécessaire ?

$\{(y \hookrightarrow 1,2)\}x := y\{(y \hookrightarrow 1,2)\}$ vrai

$\{(y \hookrightarrow 1,2) * (x \hookrightarrow 3,4)\}x := y\{(y \hookrightarrow 1,2) * (x \hookrightarrow 3,4)\}$ faux

Pourquoi la restriction suffit ?

- pas besoin de x d'un $x.i$

car alors P est forcement de la forme

$(x \mapsto x_1, x_2) * P'$ pour que C soit exécutable

et donc si x est dans R, R intersecte avec P et $P * R$ jamais satisfait et c'est ok.
- pas besoin de y d'un $x := y$ car
si y n'est pas un pointeur il ne pourra être modifié que par un $y := ...$ et donc apparaitra dans la restriction
si y est un pointeur, il ne peut pas être modifié par un $x := ...$ et pour être modifié par un $x.i := ...$ cela implique que l'on avait le P de la forme $(y \mapsto x_1, x_2) * P'$ et on a la même réponse que précédement
- pourquoi on ne peut pas modifier une variable non mentionnée dans C mais dans P ? Parce que l'on a pas de $E.i.j$ ni de $E.i = E'.j$

Axiomes de Reynolds 2

- $\{P[E/x]\}x := E\{P\}$

- $\{\exists x_1, x_2\ \ (E \hookrightarrow x_1, x_2) \wedge P[x_1/x]\}x := E.1\{P\}$

  $x_1, x_2 \notin free(E)$,$x_1 \notin free(P)$

- $$\begin{array}{c}\{\exists x_1, x_2\ \ (E \mapsto x_1, x_2) * ((E \mapsto E', x_2) \mathrel{-\!\!*} P)\} \\ E.1 := E' \\ \{P\}\end{array}$$

  $x_1, x_2 \notin free(E, E', P)$

- $$\begin{array}{c}\{\forall x'\ \ (x' \mapsto E_1, E_2) \mathrel{-\!\!*} P[x'/x]\} \\ x := \mathit{cons}(E_1, E_2) \\ \{P\}\end{array}$$

  $x' \notin free(E_1, E_2, P)$

$$(E \hookrightarrow x_1, x_2) \triangleq (\mathtt{true} * E \mapsto x_1, x_2)$$

Dispose

Sémantique opérationnelle

$$\frac{l \in Loc \quad l \in dom(h) \quad [\![E]\!]s = l}{\textit{dispose}(E), s, h \rightsquigarrow s, (h - l)}$$

Axiome

$\{P * \exists x_1, x_2\ (E \mapsto x_1, x_2)\}$
dispose(E)
$\{P\}$

Exemple

$\{\mathtt{false}\}$
$\{\mathtt{true}\ *\ \exists x_1, x_2\ (x \mapsto x_1, x_2)\ *\ \exists y_1, y_2\ (x \mapsto y_1, y_2)\}$
dispose(x)

$\{\texttt{true} * \exists x_1, x_2\ (x \mapsto x_1, x_2)\}$
dispose(x)
$\{\texttt{true}\}$

FIN

La logique de fragmentation n'est pas une logique linéaire

En logique lineaire

$$((P \mathrel{-\!\!*} Q) \Rightarrow (P \Rightarrow Q))$$

Mais pas en logique de fragmentation, car

$$m \models x \mapsto 1,1 \Rightarrow m \models (x \mapsto 1,1) \mathrel{-\!\!*} \texttt{false}$$

mais

$$m \models x \mapsto 1,1 \Rightarrow m \not\models (x \mapsto 1,1) \Rightarrow \texttt{false}$$

Exemple : unfolding `nclist42`

$\texttt{nclist42}(x) \triangleq \mu X_v.(x = \texttt{nil}) \vee \exists x_2.((x \mapsto 42, x_2) * X_v[x_2/x])$ (with $x_2 \neq x$).
Alors
$\texttt{nclist42}(x_2) = \mu X_v.(x_2 = \texttt{nil}) \vee \exists x_3.((x_2 \mapsto 42, x_3) * X_v[x_3/x_2])$ (with $x_3 \neq x_2$).

We can prove that $X_v[x_2/x]$ is equivalent to $\texttt{nclist42}(x_2)$.

$$
\begin{array}{lcl}
\texttt{nclist42}(x) & \triangleq & \mu X_v.(x = \texttt{nil}) \vee \exists x_2.((x \mapsto 42, x_2) * X_v[x_2/x]) \\
(unfolding) & = & (x = \texttt{nil}) \vee \exists x_2.((x \mapsto 42, x_2) * ((\mu X_v.(x = \texttt{nil}) \\
 & & \quad \vee \exists x_2.((x \mapsto 42, x_2) * X_v[x_2/x]))[x_2/x])) \\
\text{(variable renaming for } BI^{\mu\nu}) & = & (x = \texttt{nil}) \vee \exists x_2.((x \mapsto 42, x_2) * ((\mu X_v.(x = \texttt{nil}) \\
 & & \quad \vee \exists x_3.((x \mapsto 42, x_3) * X_v[x_3/x]))[x_2/x])) \\
\text{(simplification [/] case } \mu) & = & (x = \texttt{nil}) \vee \exists x_2.((x \mapsto 42, x_2) * (\mu X_v.(x_2 = \texttt{nil}) \\
 & & \quad \vee \exists x_3.((x_2 \mapsto 42, x_3) * X_v[x_3/x_2]))) \\
 & \triangleq & (x = \texttt{nil}) \vee \exists x_2.((x \mapsto 42, x_2) * \texttt{nclist42}(x_2))
\end{array}
$$

Semantics des expressions

$$[\![x]\!]^s \triangleq s(x),\ [\![42]\!]^s \triangleq 42,\ [\![\mathtt{false}]\!]^s \triangleq false,\ [\![E_1 + E_2]\!]^s \triangleq [\![E_1]\!]^s + [\![E_2]\!]^s,\ \ldots$$

Backward Analysis: wlp

wlp : weakest liberal precondition, such that

$$\{wlp(P, C)\}C\{P\}$$

wlp is expressed for tout P et tout C

- $\{P[E/x]\}x := E\{P\}$
- $\{\nu X_v.\ ((E = \texttt{true} \wedge wlp(X_v, C)) \vee (E = \texttt{false} \wedge P))\}$ $\texttt{while}$ E $\texttt{do}$ C $\{P\}$

Forward analysis: sp

We would like to have sp such that:

$$\{P\}C\{sp(P,C)\}$$

But it may happens that :

$$\nexists Q.\ \{P\}C\{Q\}$$

e.g. : $\forall Q.\ \neg(\{\texttt{true}\}x := \texttt{nil}; x.1 := 3\{Q\})$

➜ A two steps analysis

① Express the conditions of no error $wlp(\mathtt{true}, C)$

② Express the sp for tout P et tout C such that

Si $m \models P$ et $C, m \leadsto^* m'$ alors $m' \models sp(P, C)$.
(the usual definition of Hoare triples).

We alors have:

$$\{P \wedge wlp(\mathtt{true}, C)\}C\{sp(P, C)\}$$

$$sp(P,\ \mathtt{while}\ E\ \mathtt{do}\ C) =\ (E = \mathtt{false}) \wedge (\mu X_v.sp(X_v \wedge E = \mathtt{true}, C) \vee P)$$

$[\ /\ ]$ is not $\{\ /\ \}$

$\{\ /\ \}$: capture avoiding substitution
$[\ /\ ]$: postponed substitution connective

$\{\mathtt{true}\}x := y\{\mathtt{true}\}$ is false since the command will be stuck from a state that has no value on its stack for y

but $\{is(y)\}x := y\{\mathtt{true}\}$ is true

so $\{P\{y/x\}\}x := y\{P\}$ is unsound

but $\{P[y/x]\}x := y\{P\}$ is sound

With $is(E) \triangleq (E = E)$, since $[\![E = E]\!]_\rho = \{s, h \mid [\![E]\!]^s$ has a value$\}$

Sémantique intuitionniste

Si une propriété est vrai pour un tas, elle est vrai pour tout tas l'étendant.

$\mapsto$ intuioniste vaut $\hookrightarrow$ classique.

Pas tiers exclu : ne satisfait pas $(x \mapsto 2, 2) \vee \neg(x \mapsto 2, 2)$

Exemple amusant

$$\{\neg\exists x\ x \mapsto 1, 2\} y := \mathit{cons}(1, 2) \{(\neg\exists x\ x \mapsto 1, 2) * (y \mapsto 1, 2)\}$$

Vrai en intuitioniste car pré-cond n'a pas de modèle.
Vrai en classique.