Index

Analyse statique de pointeurs et logique BI

Stage de D.E.A.

LIX, École Polytechnique, France
Avril-Août 2002
Directeur : Radhia Cousot

Élodie-Jane Sims.

10/09/02

Plan

- La logique BI
- Analyse arrière
- Analyse avant
- Application à une analyse : Partition du tas

I - BI

BI est une logique permettant de vérifier des propriétés de l'effet sur le tas d'un programme avec pointeurs.

Utilité

- verifier des problèmes de déreferencement
- problèmes d'alias

Commandes des programmes

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

$$
i ::= 1 | 2
$$

$$
\begin{array}{rcc}
E & ::= & x \quad Variable \\
& | & 42 \quad Entier \\
& | & nil \quad nil \\
& | & a \quad Atome
\end{array}
$$

Sémantique opérationnelle :

$$C, s, h \rightsquigarrow s', h'$$

Domaine : pile+tas

$$\begin{array}{rcl} Val & = & Int \cup Atoms \cup Loc \\ s \in S & = & Var \rightharpoonup_{fin} Val \\ h \in H & = & Loc \rightharpoonup_{fin} Val \times Val \end{array}$$

Exemple pour cons

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

Formules de BI

P, Q, R	$::=$	α	Formule atomique
	$\mid$	*false*	
	$\mid$	$P \Rightarrow Q$	Implication classique
	$\mid$	$\exists x.P$	
	$\mid$	*emp*	Tas vide
	$\mid$	$P * Q$	Conjonction spatiale
	$\mid$	$P \mathrel{-\!\!\!*} Q$	Implication spatiale

α	$::=$	$E = E'$	
	$\mid$	$E \mapsto E_1, E_2$	

E	$::=$	x	$Variable$
	$\mid$	42	$Entier$
	$\mid$	nil	nil
	$\mid$	a	$Atome$

Sémantique des formules : $s, h \models P$

$h \sharp h'$: $dom(h)$ et $dom(h')$ disjoints
$h \cdot h'$: union de h et h' de domaines disjoints
$free(P) \subseteq dom(s)$

$s, h \models E \mapsto (E_1, E_2)$ ssi
$dom(h) = \{[\![E]\!]s\}$ et $h([\![E]\!]s) = \langle [\![E_1]\!]s, [\![E_2]\!]s \rangle$

$$s, h \models P * Q \quad ssi \quad \exists h_0, h_1 \; h_0 \sharp h_1, \; h = h_0 \cdot h_1 \; s, h_0 \models P \text{ et } s, h_1 \models Q$$

$$s, h \models P \mathrel{-\!\!*} Q \quad ssi \quad \forall h', \; Si \; h \sharp h' \; et \; s, h' \models P \; Alors \; s, h \cdot h' \models Q$$

Exemples

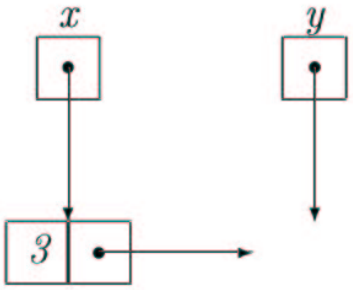


FIG. 1:

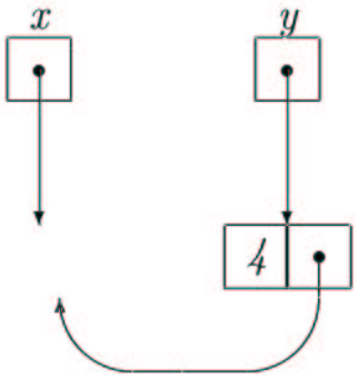


FIG. 2:

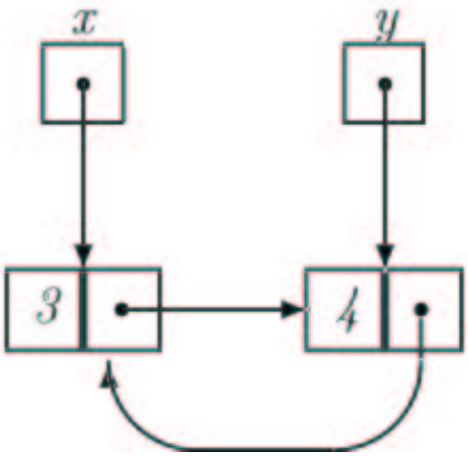


FIG. 3:

$$Fig1 \models (x \mapsto 3, y) \quad Fig2 \models (y \mapsto 4, x)$$
$$Fig3 \models (x \mapsto 3, y) * (y \mapsto 4, x)$$
$$Fig3 \not\models (x \mapsto 3, y) \wedge (y \mapsto 4, x)$$
$$Fig1 \models (y \mapsto 4, x) \twoheadrightarrow ((x \mapsto 3, y) * (y \mapsto 4, x))$$
$$Fig2 \models (x \mapsto 3, y) \twoheadrightarrow ((x \mapsto 3, y) * (y \mapsto 4, x))$$

Vocabulaire sur la sémantique des commandes

- **Configuration** : C, s, h ou *configuration terminale* s, h
- C, s, h **bloquée** : $\nexists K\ C, s, h \rightsquigarrow K$
- C, s, h **sûre** : Si $C, s, h \rightsquigarrow^* K$ alors K n'est pas bloquée

Interprétation des triplets

$\{P\}C\{Q\}$ vrai :

Si $\forall s, h$ tq $free(P) \cup free(Q) \subseteq dom(s)$
Si $s, h \models P$ alors

– C, s, h est sûre

– si $C, s, h \rightsquigarrow^* s', h'$ alors $s', h' \models Q$

Remarque

$\{true\}C\{true\}$ peut être faux,

Exemple
$\{true\}x := nil; x.1 := 3\{true\}$ est faux

Axiome d'extension du tas

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

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

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

Ü utile pour la modularité

II- Analyse arrière

On va chercher P t.q. $\{P\}C\{true\}$ et alors l'exécution de C sera sûre à partir de n'importe quel état satisfaisant P.

Ou, on va exprimer une propriété Q que l'on veut vérifier à la fin de l'exécution de C, et chercher P tq $\{P\}C\{Q\}$

wlp : weakest liberal precondition

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

Exemple de wlp

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

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

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

' $\{\exists x_1, x_2 \ \ (E \mapsto x_1, x_2) * ((E \mapsto E', x_2) \mathrel{-\!\!*} P)\}$
$E.1 := E'$
$\{P\}$
$x_1, x_2 \notin free(E, E', P)$

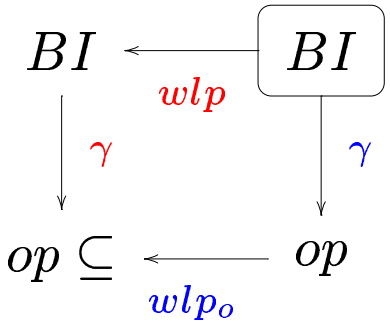
' $\{\forall x' \ \ (x' \mapsto E_1, E_2) \mathrel{-\!\!*} P[x'/x]\}$
$x := cons(E_1, E_2)$
$\{P\}$
$x' \notin free(E_1, E_2, P)$

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

Preuve des wlp

wlp dans le domaine opérationnel:
$wlp_o(\Delta, C) = \{s, h | C, s, h$ est sûre
$\wedge$(si $C, s, h \rightsquigarrow^* s', h'$ alors $s', h' \in \Delta\}$

Concrétisation :
$\gamma(P) = \{s, h | s, h \models P\}$



Nous avons prouvé par induction sur la syntaxe de C que :

$$\gamma(wlp(P, C)) \subseteq wlp_o(\gamma(P), C)$$

III- Analyse avant

Nous voudrions sp tq:

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

Mais :

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

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

Ü Analyse en deux étapes

‹ Trouver la condition d'absence d'erreur $wlp(\mathit{true}, C)$

› Calculer sp pour des formules sûres.

On a ainsi:

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

vrai

Remarque, parfois $P \models wlp(\mathit{true}, C)$, et on a $\{P\}C\{sp(P,C)\}$ vrai.

Exemple de $wlp(\mathbf{true}, C)$

$$wlp(\mathsf{true}, x := E) = is(E)$$

$$wlp(\mathsf{true}, x := E.i) = \exists x_1 \exists x_2 . E \hookrightarrow x_1, x_2$$

$with \;\; x_i \notin FV(E)$

$$wlp(\mathsf{true}, E.i := E') = \exists x_1 \exists x_2 . (E \mapsto x_1, x_2) \wedge is(E')$$

$with \;\; x_i \notin FV(E, E')$

$$wlp(\mathsf{true}, x := \mathtt{cons}(E_1, E_2)) = is(E_1) \wedge is(E_2)$$

$$wlp(\mathsf{true}, \mathtt{dispose}(E)) = \exists x_1 \exists x_2 . E \hookrightarrow x_1, x_2$$

$with \;\; x_i \notin FV(E)$

Exemple de sp

$$\{P\}x := E\{\exists x'.\ P[x'/x] \wedge x = E[x'/x]\}$$

$$\{P\}x := E.i\{\exists x'.\ P[x'/x] \wedge x = (E[x'/x]).i\}$$

$x' \notin FV(E, P)$

$$\{P\}$$
$$E_1.1 := E_2$$
$$\{\exists x_1 \exists x_2.(E_1 \mapsto E_2, x_2) * ((E_1 \mapsto x_1, x_2) \twoheadrightarrow P)\}$$

$x_i \notin FV(E_1, E_2, P)$

$$\{P\}$$
$$x := \mathit{cons}(E_1, E_2)$$
$$\{\exists x'.(P[x'/x] * (x \mapsto E_1[x'/x], E_2[x'/x]))\}$$

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

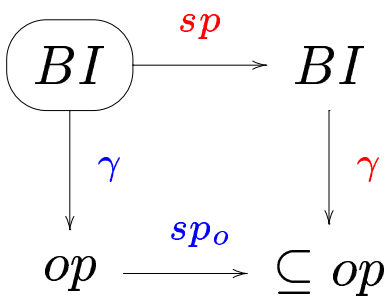
Preuve des sp

sp dans le domaine opérationnel:
$sp_o(\Delta, C) = \{s', h' | \exists s, h \in \Delta.\ C, s, h \rightsquigarrow^* s', h'\}$

Nous pouvons réécrire la définition de triplet vrai:

$\{P\}C\{Q\}$ vrai ssi

$P \models wlp(\mathit{true}, C) \wedge sp_o(\gamma(P), C) \subseteq \gamma(Q)$

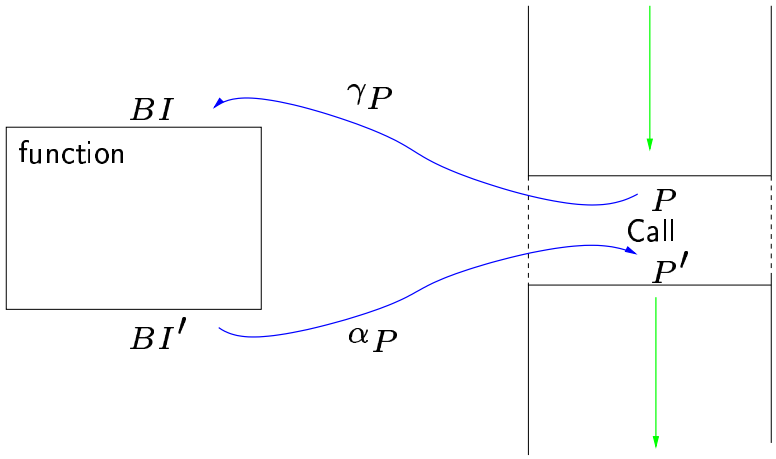


Nous avons prouvé par induction sur la syntaxe de C que :

If $P \models wlp(\mathit{true}, C)$ then $sp_o(\gamma(P), C) \subseteq \gamma(sp(P, C))$

IV- Partitions

But général : utiliser BI comme interface avec une analyse dans un domaine P



Domaine des partitions : ensemble de $[a/b]$.

$[a/b]$: assertion disant qu'on ne peut pas atteindre le même objet à partir de a et b (par déréférencements successifs)

But :

- trouver une traduction $\gamma_P : Partitions \to BI$,
- vérifier l'analyse avant des partitions
- Écrire l'analyse arrière. La vérifier (non fait).

Exemple des sp du domaine des partitions

$$\{P\}x := nil\{P \cup \{[x/z]|\forall z\}\}$$

$$\{P\}x := y\{P\backslash\{[x/z]|\forall z\}\cup\{[x/z]|[y/z] \in P\}\}$$

$$\{P\}x.i := y\{P \setminus \{[v/w]|([y/w] \notin P \wedge [x/v] \notin P)\}$$

$\gamma_P : Partitions \rightarrow BI$

$$\texttt{isinheap}(x) \equiv \exists x_1, x_2.\ (x \hookrightarrow x_1, x_2)$$

$$\texttt{isdangling}(x) \equiv \texttt{isloc}(x) \wedge \neg\texttt{isinheap}(x)$$

$$\texttt{Nodangling2} \equiv \forall v, v'.\ (\texttt{isinheap}(v) \wedge (v' = v.1 \vee v' = v.2))$$

$$\Rightarrow \neg\ \texttt{isdangling}(v')$$

Par exemple :

$$\left[\begin{array}{c} x \mapsto l_x \\ y \mapsto l_y \\ z \mapsto l_z \end{array}\right], \left[\begin{array}{c} l_x \mapsto 3, 4 \\ l_y \mapsto 5, l_w \end{array}\right]$$

$\models \texttt{isdangling}(z)$
mais $\not\models$ Nodangling2 à cause de l_w.

$\gamma_P : Partitions \rightarrow BI$

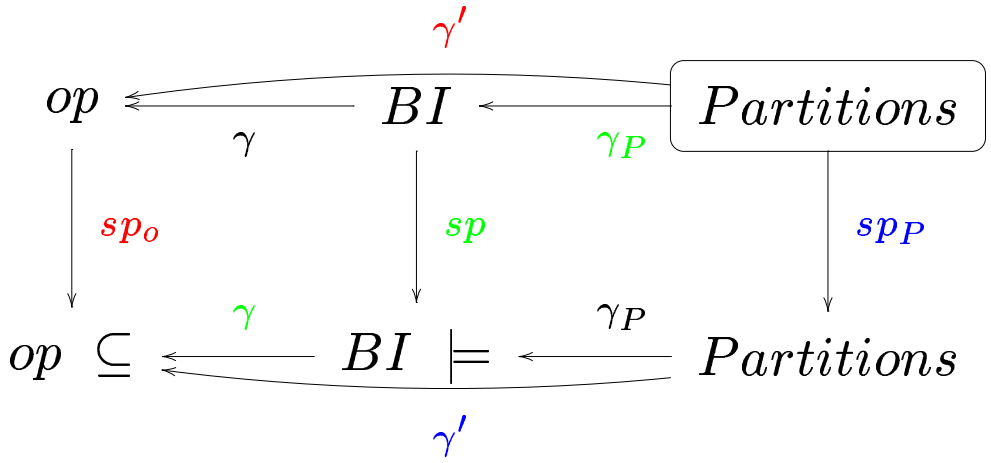
$$
\begin{aligned}
[x/y] \quad \equiv \quad & \neg \texttt{isloc}(x) \vee \neg \texttt{isloc}(y) \\
& \vee \exists x_1, x_2, y_1, y_2. \\
& \quad (x \hookrightarrow x_1, x_2 \wedge \texttt{Nodangling2}) \\
& \quad * (y \hookrightarrow y_1, y_2 \wedge \texttt{Nodangling2}) \\
& \quad * \mathit{true}
\end{aligned}
$$

$$
\begin{aligned}
\gamma_p(P) \quad = \quad & \bigwedge_{[a/b] \in P} [a/b] \\
& \wedge \texttt{Nodangling2} \\
& \wedge \bigwedge_{\forall z \text{“}\in\text{”} P} \neg\, \texttt{isdangling}(z)
\end{aligned}
$$

Vérification de l'analyse avant des partitions

Nous avons prouvé pour chaque commande que :

$$sp_o(\gamma'(P), C) \subseteq \gamma'(sp_P(P, C))$$



with $\gamma' = \gamma \circ \gamma_P$.

Conclusion

Nous nous intéressons à BI car c'est une logique qui permet des analyses modulaires.

Nous avons:

$\surd$ complété l'opérateur wlp et prouvé sa correction par rapport à la sémantique opérationnelle

$\surd$ développé un opérateur sp et prouvé sa correction

$\surd$ utilisé BI comme interface pour une analyse de partition du tas et prouvé la correction de cette analyse

Nous voulons utiliser BI comme interface à des analyses de type shape- ou escape-analyzes.

$while$

$$wlp(P, while\ E\ do\ C) = gfp^{\models}_{\mathsf{true}}\ \lambda X.\ ((E = \mathsf{true} \wedge wlp(X, C))\ \vee (E = \mathsf{false} \wedge P))$$

$$wlp(\mathsf{true}, while\ E\ do\ C) = gfp^{\models}_{\mathsf{true}}\ \lambda X.\ ((E = \mathsf{true} \wedge wlp(X, C))\ \vee E = \mathsf{false})$$

$$sp(P, while\ E\ do\ C) = (E = \mathsf{false}) \wedge lfp^{\models}_{P}\ \lambda X.(sp(X \wedge E = \mathsf{true}, C) \vee X)$$

FIN

$\surd$ Différence entre $*$ et $\wedge$

- $(x \mapsto 1, 2) * (x \mapsto 1, 2)$: tjs faux
- $(x \mapsto 1, 2) \wedge (x \mapsto 1, 2)$ equi. $(x \mapsto 1, 2)$
- $(x \mapsto 1, 2) * \neg(x \mapsto 1, 2)$ vrai si x pointe sur $\langle 1, 2 \rangle$
- $(x \mapsto 1, 2) \wedge \neg(x \mapsto 1, 2)$: tjs faux

$\surd$ Tableau des wlp

$$
\begin{array}{lll}
wlp(P, & x := E) = & P[E/x] \\
wlp(P, & x := E.i) = & \exists x_1 \exists x_2.(P[x_i/x] \wedge (E \hookrightarrow x_1, x_2)) \\
 & & \quad with \;\; x_i \notin FV(E, P) \\
wlp(P, & E.1 := E') = & \exists x_1 \exists x_2.(E \mapsto x_1, x_2) * ((E \mapsto E', x_2) \mathrel{-\!\!*} P) \\
 & & \quad with \;\; x_i \notin FV(E, E', P) \\
wlp(P, & E.2 := E') = & \exists x_1 \exists x_2.(E \mapsto x_1, x_2) * ((E \mapsto x_1, E') \mathrel{-\!\!*} P) \\
 & & \quad with \;\; x_i \notin FV(E, E', P) \\
wlp(P, & x := \texttt{cons}(E_1, E_2)) = & \forall x'.(x' \mapsto E_1, E_2) \mathrel{-\!\!*} P[x'/x] \\
 & & \quad with \;\; x' \notin FV(E_1, E_2, P) \\
wlp(P, & \texttt{dispose}(E)) = & P * (\exists a \exists b.(E \mapsto a, b)) \\
 & & \quad with \;\; a, b \notin FV(E) \\
wlp(P, & C_1; C_2) = & wlp(wlp(P, C_2), C_1) \\
wlp(P, & if \;\; E \;\; then \;\; C_1 \;\; else \;\; C_2) = & (E = \mathit{true} \wedge wlp(P, C_1)) \\
 & & \vee (E = \mathit{false} \wedge wlp(P, C_2)) \\
wlp(P, & if \;\; E \;\; then \;\; C_1) = & (E = \mathit{true} \wedge wlp(P, C_1)) \vee (E = \mathit{false} \wedge P) \\
wlp(P, & while \;\; E \;\; do \;\; C_1) = & gfp \overset{\models}{}_{\mathit{true}} \; \lambda X.((E = \mathit{true} \wedge wlp(X, C_1)) \\
 & & \qquad \vee (E = \mathit{false} \wedge P)) \\
 & & \\
wp(P, & while \;\; E \;\; do \;\; C_1) = & lfp \overset{\models}{}_{P} \; \lambda X.((E = \mathit{true} \wedge wp(X, C_1)) \vee X)
\end{array}
$$

$\sqrt{wlp(\mathbf{true}, C)}$

$$
\begin{array}{ll}
wlp(\mathit{true}, x := E) = & is(E) \\
wlp(\mathit{true}, x := E.i) = & \exists x_1 \exists x_2 . E \hookrightarrow x_1, x_2 \\
wlp(\mathit{true}, E.i := E') = & \exists x_1 \exists x_2 . (E \hookrightarrow x_1, x_2) \wedge is(E') \\
wlp(\mathit{true}, x := \mathtt{cons}(E_1, E_2)) = & is(E_1) \wedge is(E_2) \\
wlp(\mathit{true}, \mathtt{dispose}(E)) = & \exists x_1 \exists x_2 . E \hookrightarrow x_1, x_2 \\
wlp(\mathit{true}, C_1; C_2) = & wlp(wlp(\mathit{true}, C_2), C_1) \\
wlp(\mathit{true}, i\ E\ t\ C_1\ e\ C_2) = & (E = \mathit{true} \wedge wlp(\mathit{true}, C_1)) \\
 & \vee (E = \mathit{false} \wedge wlp(\mathit{true}, C_2)) \\
wlp(\mathit{true}, if\ \ E\ \ then\ \ C_1) = & (E = \mathit{true} \wedge wlp(\mathit{true}, C_1)) \\
 & \vee E = \mathit{false} \\
wlp(\mathit{true}, while\ \ E\ \ do\ \ C_1) = & gfp \models_{\mathit{true}} \lambda X.((E = \mathit{true} \wedge wlp(X, C_1)) \\
 & \vee E = \mathit{false})
\end{array}
$$

$\surd$ Preuve des sp 2

Pour prouver les sp, nous avons changé le langage :

Domaine opérationnel : $m = s, h$ ou Ω_o

Sémantique opérationnelle :

$$\begin{array}{rll} C, m \to m' & \text{ssi} & (m = \Omega_o \wedge m' = \Omega_o) \\ & & \vee (m = s, h \wedge m' = \Omega_o \wedge C, s, h \not\rightsquigarrow) \\ & & \vee (m = s, h \wedge m' = s', h' \wedge C, s, h \rightsquigarrow s', h') \\ C, m \to C', m' & \text{ssi} & m = s, h \wedge m' = s', h' \wedge C, s, h \rightsquigarrow C', s', h' \end{array}$$

$$sp_o^2(\Delta, C) = \{m' | \exists m \in \Delta.\ C, m \to^* m'\}$$

Formules : BI + $\texttt{Err}$

$\Omega_o \models \texttt{Err} \quad s, h \not\models \texttt{Err}$

$m \models P \Rightarrow Q$ ssi Si $m \models P$ alors $m \models Q$

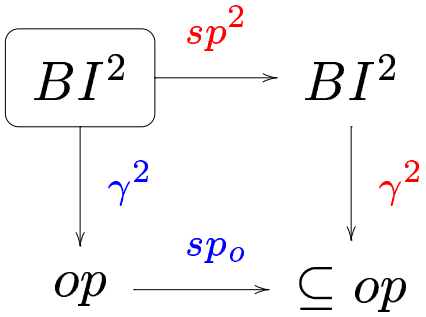
autres formules de BI F: $\Omega_o \not\models F$

$\sqrt{}$

Le sens des triplets est aussi changé.
Si on a $\{P\}C\{Q\}$ alors pour savoir si l'exécution de C est sûre à partir de P, il suffit de vérifier que $Q \not\models \texttt{Err}$.

Pour montrer que nos formules sp sont juste, il suffit de prouver par induction sur la syntaxe de C que nous avons:

$$sp_o^2(\gamma^2(P), C) \subseteq \gamma^2(sp^2(P, C))$$



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 $\forall s,h \;\; si \;\; s,h \models P \;\; alors \;\; s,h \models Q$

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 $(P * Q \not\models P)$

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

  même problème de la taille du tas

Dispose

Sémantique opérationnelle

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

Axiome

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

Exemple

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

Axiome d'extension du tas

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

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

$ModifiesOnly(C)$: 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 $E.i = E'.j$

Résumé de l'apport du papier

- modèle classique (pas intuitioniste comme dans le passé)
  permet d'exprimer plus de propriétés, comme *empty*

- $*$ et $\mathrel{-\!\!\!*}$
  calcul des pré-conditions avec preuve de la sûreté et complétude des axiomes

- axiome d'extension
  utile par exemple pour les appels de fonctions

Rq.1 :
- Sûreté : tout triplet dérivable par un axiome est vrai
- Complétude : tout triplet vrai est dérivable par les axiomes

Rq.2 : en fait l'axiome d'assignement donne
$\{\mathit{true}\}x := y\{\mathit{true}\}$
qui est faux

Projet : analyse d'échappement

Pourquoi BI pour l'analyse d'échappement ?

- l'axiome d'extension permet de ne travailler que sur la zone modifiée par le programme.
- manière simple
- règles prouvées

Mais :

- est-ce que BI est adéquat directement ? Approximation ?
- problème d'automatisation pour
  - utiliser l'axiome Conséquence
  - trouver l'invariant automatiquement pour l'axiome *while*
- possibilité d'analyse avant ?

Questions

3 *What about completeness?*

Définissent :

$s, h \in wp(C, Q)$ ssi C, s, h est sûre et $si\ C, s, h \rightsquigarrow^* s', h'\ alors\ s', h' \models Q$

Théorème : $s, h \models BW(C, Q)\ ssi\ s, h \in wp(C, Q)$

$\{P\}C\{Q\}$ dérivable par les axiomes ssi vrai.

3 *What about dangling pointers?*

Nous avons des danglings pointeurs.

Sinon *cons* ne serait pas complète.

ex : $\{true\}x := cons(1, 2)\{\neg(x = y)$,

faux dans notre cas, vrai si pas de dangling pointeurs.

???

3 *Pointwise fashion of* $\Rightarrow$

Questions...

- *Coût de l'analyse* : ?
  Pb de quand on doit utiliser l'axiome conséquence, comment décider ce que l'on fait ? pb comment faire les simplifications dans ex 7 ??
  Ex 7, comment ils trouvent la post-cond au début ?

- *Terminaison* : non, si il y a des boucles

- *Intérêt d'utiliser BI pour l'analyse d'échappement ?*
  - Par rapport au travail de M. Rinard (shape escape analysis), permet de prouver que l'analyse est correcte, que l'on a les bonnes règles. Semble bien moins compliqué...?
  - Par rapport au travail de B. Blanchet (integer escape analysis), je ne sais pas encore ;).
  - FRAME AXIOME : permet d'obtenir directement les variables modifiées par le programme

- *Analyse avant* ?
  Analyse d'échappement est de l'analyse avant, ici on fait de l'analyse arrière.

Arrière: on part d'une proposition sur l'état d'arrivée, on remonte pour trouver les conditions initiales minimales, c'est OK.
Avant: on part d'une proposition sur l'état de départ et on cherche de trouver le maximum d'information sur l'état d'arrivée.
Que faire par exemple si on a
$\{2 \times x = 4\}x := 3\{?\}$,

- $\{?\} = \{x = 3\}$
- $\{?\} = \{2 \times x = 6\}$, vérifier la formule?
- $\{?\} = \{x = 3 \wedge 2 \times x = 6\}$
- ...

Aussi, pb comment trouver ce que l'on peut retirer?

3 Pas clair *while*: dans l'exemple 7 ils disent ils utilisent la règle usuelle de Hoare avec la précondition comme invariant.

3 Appel de fonction : dans l'ex. 8.2 juste l'axiome d'extension

Bricabrac

- La règle du tiers exclu est valide dans notre logique.
- *cons* est complète si on a des “dangling” pointeur.
- nouveauté : $*$ et $-\!\!\!*$
- spécificité : deux types d'implications
- $*$ sans contraction Parce que en intuitioniste on est invariant par extension du tas or des propriétés telle *empty* ne peut pas alors être exprimée.
- Pureté : expressions sans $\mapsto$ ni $.i$ ne dépendent pas du tas
- On n'a pas de $E.i.j$ ni de $E.i := E'.j$

Sémantique opérationnelle :

$$C, s, h \rightsquigarrow s', h'$$

Domaine : pile+tas

$$\begin{array}{rcl} Val & = & Int \cup Atoms \cup Loc \\ S & = & Var \rightharpoonup_{fin} Val \\ H & = & Loc \rightharpoonup_{fin} Val \times Val \end{array}$$

Commandes modifiant juste la pile

$$\frac{[\![E]\!]s = v}{x := E, s, h \rightsquigarrow [s|x \mapsto v], h}$$

$$\frac{[\![E]\!]s = l \in Loc \quad h(l) = r}{x := E.i, s, h \rightsquigarrow [s|x \mapsto \pi_i r], h}$$

Commande modifiant juste le tas

$$\frac{[\![E]\!]s = l \in Loc \quad h(l) = r \quad [\![E']\!]s = v'}{E.i = E', s, h \rightsquigarrow s, [h|l \mapsto (r|i \mapsto v')]}$$

Commande modifiant la pile et le tas

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

Sémantique des formules : $s, h \models P$

$h \sharp h'$: $dom(h)$ et $dom(h')$ disjoints
$h \cdot h'$: union de h et h' de domaines disjoints
$free(P) \subseteq dom(s)$

Formules atomiques

$s, h \models E = E'$ ssi $[\![E]\!]s = [\![E']\!]s$

$s, h \models E \mapsto (E_1, E_2)$ ssi
$dom(h) = \{[\![E]\!]s\}$ et $h([\![E]\!]s) = \langle [\![E_1]\!]s, [\![E_2]\!]s \rangle$

Formules classiques

$$
\begin{array}{lcll}
s, h & \models & \mathit{false} & jamais \\
s, h & \models & P \Rightarrow Q \quad ssi & Si\ s, h \models P\ Alors\ s, h \models Q \\
s, h & \models & \exists x.P \quad ssi & \exists v \in Val.[s|x \mapsto v], h \models P
\end{array}
$$

Formules spatiales

$$
\begin{array}{lcl}
s, h \models \mathit{emp} & ssi & h = [] \text{: tas vide} \\
s, h \models P * Q & ssi & \exists h_0, h_1\ h_0 \sharp h_1,\ h = h_0 \cdot h_1 \\
 & & s, h_0 \models P \text{ et } s, h_1 \models Q \\
s, h \models P \mathrel{-\!\!*} Q & ssi & \forall h',\ Si\ h \sharp h'\ et\ s, h' \models P \\
 & & Alors\ s, h \cdot h' \models Q
\end{array}
$$

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
$(x \mapsto 2, 2) \wedge \neg(x \mapsto 2, 2)$

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