BI as an Assertion Langage for Mutable Data Structures

S. Ishtiaq - P. O'Hearn

29/04/02

Élodie-Jane Sims.

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

Motivation des auteurs

– verifier des problèmes de déreferencement

– problèmes d'alias

Commandes des programmes

$$
\begin{array}{rcc}
C & ::= & x := E \\
& | & x := E.i \\
& | & E.i := E' \\
& | & x := \mathit{cons}(E_1, E_2) \\
& | & \ldots \\
& | & C; C
\end{array}
$$

$$
i \quad ::= \quad 1|2
$$

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

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

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

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

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

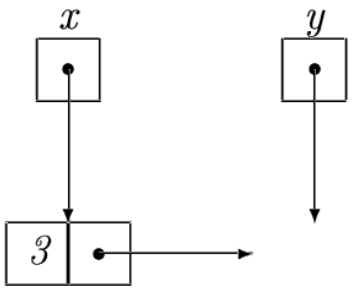

FIG. 1:

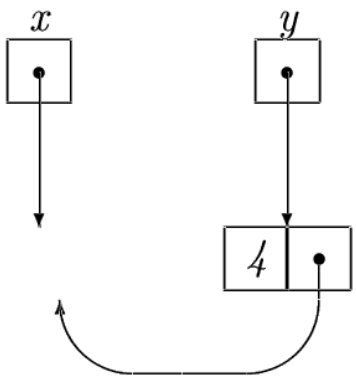

FIG. 2:

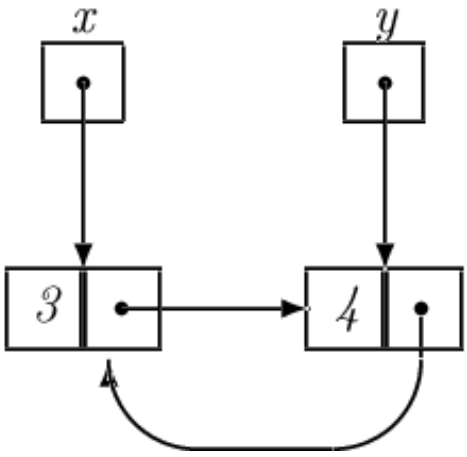

FIG. 3:

$Fig1 \models (x \mapsto 3, y) \quad Fig2 \models (y \mapsto 4, x)$
$Fig3 \models (x \mapsto 3, y) * (y \mapsto 4, x)$
$Fig3 \not\models (x \mapsto 3, y) \wedge (y \mapsto 4, x)$
$Fig1 \models (y \mapsto 4, x) \mathrel{-\!\!*\!\!\!*} ((x \mapsto 3, y) * (y \mapsto 4, x))$
$Fig1 \models (x \mapsto 3, y) \mathrel{-\!\!*\!\!\!*} ((x \mapsto 3, y) * (y \mapsto 4, x))$

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

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 bloqué ou K est terminale

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

$\{P\}C\{Q\}$ différent de $P \Rightarrow Q$ et de $P \models Q$

$\{true\}C\{true\}$ peut être faux,

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

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 $\{P\}C\{Q\}$

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

Axiomes de Reynolds 2

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

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

  $x_1, x_2 \notin free(E, E', P)$
- $\{\forall x' \ \ (x' \mapsto E_1, E_2) \mathrel{-\!\!*} P[x'/x]\}$
  $x := cons(E_1, E_2)$
  $\{P\}$

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

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

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 $E.i = E'.j$

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.

Résumé de l'apport du papier

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

- $*$ et $-\!\!*$
  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

✓ *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.

✓ *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.
???

✓ *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?

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

✓ 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 $\mathrel{-\!\!*}$
- 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

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

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

Commande modifiant juste le tas

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

Commande modifiant la pile et le tas

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

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

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

Pourquoi

$$ModifiesOnly(C) \cap free(R) = \emptyset \textbf{ ?}$$

Si on a $\{\mathit{true} * P\}C\{\mathit{true} * Q\}$,
C peut être un programme modifiant une partie du tas qui n'est pas dans le domaine du tas vérifiant P.
(car true *veut dire "n'importe quel tas hors du domaine de P")*.
Alors que dans $\{\mathit{true} * P * R\}$ *true* veut dire "n'importe quel tas hors du domaine de P et de R"
Si le "domaine de *true*" modifié par C intersecte avec le domaine de R on aura $\{P * R\}C\{Q * R'\}$.

Sémantique intuitionniste

Condition de monotonicité : Si $h' \sqsubseteq h$ et $s, h \models P$ alors $s, h' \models P$

Enlève *emp*.
Rajoute définitions pour $\wedge$, $\vee$, $\forall$.
On redéfinit $\mapsto$ avec le sens ancien de $\hookrightarrow$.
On définit $\Rightarrow$ vrai à l'ancien sens pour toute extention de la pile.

Pas tiers exclu : ne satisfait pas $(x \mapsto 2, 2) \wedge \neg(x \mapsto 2, 2)$
Mais $s, h \models (E \mapsto E_1, E_2) * P$ ssi $s, h@l \models (E \mapsto E_1, E_2)$ et $s, h - l \models P$

Traduction int. -> cla.
$\mapsto$ -> $\hookrightarrow$
$P \Rightarrow Q$ -> *true* $\rightarrow\!\!\!*(trad(P) \Rightarrow trad(Q))$

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.