Capacités, invariants et régions abstraites

Romain Bardou

CeProMi — Mars 2009

Système d'invariant voulu

expressivité à la Spec#

- état fermé / ouvert

avec moins d'annotations

- inférence possible au moins partiellement

et des effets cachés

- abstraction du comportement interne

dans un contexte à la Why

Maintenir les invariants

connaître statiquement l'état ouvert / fermé

- moins d'annotations

empêcher les fuites d'objets cachés

- risque de rompre un invariant

Solution avec régions et capacités

régions : informations d'aliasing

- groupes de pointeurs

capacités : informations linéaires sur les régions

- état ouvert / fermé

Ingrédients

régions groupes ou singletons

- opérations de focus / unfocus
- capacités "implicatives" $\sigma \multimap \rho$

régions nommées existentiellement

- création de régions à la volée

Problématiques orthogonales

expressivité du suivi des pointeurs ouverts

- bénéficie des régions singletons
- bénéficie des régions existentielles

abstraction des effets

$\Longrightarrow$ cherchons d'abord sans les régions singletons et existentielles

Ouverture locale

connaître le pointeur ouvert sans régions singletons

idée : portée de l'ouverture limitée statiquement

unpack e_1 **in**
  e_2
end

problème : la région de e_1 contient d'autres pointeurs

solution : sortir e_1 de sa région temporairement

unpack e_1 **as** $x[r]$ **in**
  e_2
end

Allocation

sans régions existentielles

- allocation dans une région existante
- allocation dans une région groupe

objet alloué fermé

- invariant à vérifier
- objet initialisé

syntaxe :

$$\textbf{new}\ \mathcal{C}[\rho](e)$$

Syntaxe : classes

$$
\begin{array}{l}
\textbf{class } \mathcal{C} = \\
\quad \textbf{own } r, \cdots, r \\
\quad \tau \\
\quad \textbf{inv}(x) = P \\
\textbf{end}
\end{array}
$$

avec :

$$\mathcal{C} ::= \langle r, \cdots, r \rangle \ (\tau, \cdots, \tau) \ C$$

Syntaxe : types

$$
\begin{array}{lcll}
\tau & ::= & \mathsf{unit}, \mathsf{int}, \mathsf{bool}, \cdots & \text{Type de base} \\
 & & \tau \times \cdots \times \tau & \text{Tuple} \\
 & & \mathcal{C}[\rho] & \text{Pointeur} \\
 & & \alpha & \text{Variable de type} \\
 & & \alpha[\rho] & \text{Variable de type pointeur}
\end{array}
$$

Syntaxe : définitions et déclarations

val $f(x\colon \tau, \cdots, x\colon \tau)\colon \tau$
  pre $\{\Sigma,\cdots,\Sigma\}$, P
  post $\{\Sigma,\cdots,\Sigma\}$, P
  assigns $\{\rho,\cdots,\rho\}$
  $= e$

Syntaxe : expressions

e	$::=$	v	Valeur
		$(e,\cdots,e)$	Tuples
		$e.i$	Projection
		x	Variable
		$\textbf{let}\ x = e\ \textbf{in}\ e$	Variable locale
		$\textbf{new}\ \mathcal{C}[\rho](e)$	Allocation
		$e := e$	Affectation
		$!e$	Déréférencement
		$\textbf{unpack}\ e\ \textbf{as}\ x[r]\ \textbf{in}\ e$	Ouverture locale
		$e;\ e$	Séquence
		$f(e,\cdots,e)$	Appel de fonction
		$\textbf{if}\ e\ \textbf{then}\ e\ \textbf{else}\ e$	Test
		$\textbf{region}\ r\ \textbf{in}\ e$	Région locale

Syntaxe : régions et capacités

$$
\begin{array}{lcll}
\rho & ::= & r & \text{Nom de région, variable de région} \\
 & & \rho.r & \text{Région locale à une classe}
\end{array}
$$

$$
\begin{array}{lcll}
\Sigma & ::= & \rho^{\times} & \text{Fermée} \\
 & & \rho^{\circ} & \text{Ouverte}
\end{array}
$$

Exemple : calculateur de Morgan

$$
\begin{array}{l}
\textbf{class}\ \mathit{Morgan} = \\
\quad \textbf{own}\ \rho_{sum}, \rho_{count} \\
\quad (\mathit{Long}[\rho_{sum}], \mathit{Long}[\rho_{count}]) \\
\textbf{end}
\end{array}
$$

Exemple : moyenne

$$
\begin{array}{l}
\textbf{val}\ \mathit{average}(\mathit{morgan}\text{: } \mathit{Morgan}[\rho])\text{: } \mathit{Long}[\rho'] \\
\quad \textbf{pre}\ \{\rho^{\times}\},\ !(!\mathit{morgan}).2 > 0 \\
\quad \textbf{post}\ \{\rho^{\times}\},\ !\textbf{result} = !(!\mathit{morgan}).1\ /\ !(!\mathit{morgan}).2 \\
\quad = \textbf{new}\ \mathit{Long}[\rho']\ (!(!\mathit{morgan}).1\ /\ !(!\mathit{morgan}).2)
\end{array}
$$

Exemple : insertion

val *insert*(*morgan*: *Morgan*[ρ], *n*: *Long*[ρ']): unit
  pre $\{\rho^\times\}$
  post $\{\rho^\times\}$,
      $!(!morgan).1 = \textbf{old}(!(!morgan).1) + !n$
    $\wedge\ !(!morgan).2 = \textbf{old}(!(!morgan).2) + 1$
  assigns $\{\rho.\rho_{sum}, \rho.\rho_{count}\}$ =
    unpack *morgan* **as** *morgan*[ρ''] **in**
      unpack (!*morgan*).1 **as** *sum*[ρ_s] **in** $sum := !sum + !n$;
      unpack (!*morgan*).2 **as** *count*[ρ_c] **in** $count := !count + 1$
    end

Typage

séparation entre typage et typage des capacités

- typage similaire à celui de ML
- capacités vues comme une analyse supplémentaire modulaire

Typage : déréférencement

$$\frac{\Gamma \vdash e : \mathcal{C}[\rho] \qquad \mathcal{C} : \tau_1}{\Gamma \vdash !e : \tau_1}\ \text{Deref}$$

Typage : allocation

$$\frac{\mathcal{C} : \tau \qquad \Gamma \vdash e : \tau}{\Gamma \vdash \textbf{new}\ \mathcal{C}[\rho](e) : \mathcal{C}[\rho]}\ \text{Alloc}$$

Typage : ouverture locale

$$\frac{\Gamma \vdash e_1 \ : \ \mathcal{C}[\rho] \qquad \Gamma,\ x \ : \ \mathcal{C}[r] \vdash e_2 \ : \ \tau_2}{\Gamma \vdash \textbf{unpack } e_1 \textbf{ as } x[r] \textbf{ in } e_2 \ : \ \tau_2} \text{ Unpack}$$

Typage des capacités

$$\frac{}{\{\bar{\Sigma}\}\ x\ \{\bar{\Sigma}\}}\ \text{CVar} \qquad \frac{\{\bar{\Sigma}_1\}\ e\ \{\bar{\Sigma}_2,\ \Sigma\}}{\{\bar{\Sigma}_1\}\ e\ \{\bar{\Sigma}_2\}}\ \text{CWeaken}$$

$$\frac{\{\bar{\Sigma}_1\}\ e_1\ \{\bar{\Sigma}_2\} \qquad \{\bar{\Sigma}_2\}\ e_2\ \{\bar{\Sigma}_3\}}{\{\bar{\Sigma}_1\}\ e_1;\ e_2\ \{\bar{\Sigma}_3\}}\ \text{CSeq}$$

$$\frac{\{\bar{\Sigma}_1\}\ e_1\ \{\bar{\Sigma}_2\} \qquad \{\bar{\Sigma}_2\}\ e_2\ \{\bar{\Sigma}_3\} \qquad \{\bar{\Sigma}_2\}\ e_3\ \{\bar{\Sigma}_3\}}{\{\bar{\Sigma}_1\}\ \textbf{if}\ e_1\ \textbf{then}\ e_2\ \textbf{else}\ e_3\ \{\bar{\Sigma}_3\}}\ \text{CIf}$$

Capacités : affectation, déréférencement

$$\frac{e_1 \;:\; \rho \qquad \{\bar{\Sigma}_1\}\ e_1\ \{\bar{\Sigma}_2\} \qquad \{\bar{\Sigma}_2\}\ e_2\ \{\bar{\Sigma}_3,\ \rho^\circ\}}{\{\bar{\Sigma}_1\}\ e_1 := e_2\ \{\bar{\Sigma}_3,\ \rho^\circ\}}\ \text{CAffect}$$

$$\frac{\{\bar{\Sigma}_1\}\ e\ \{\bar{\Sigma}_2\}}{\{\bar{\Sigma}_1\}\ !e\ \{\bar{\Sigma}_2\}}\ \text{CDeref}$$

Capacités : régions locales

$$\frac{\{\bar{\Sigma}_1,\ r^{\times}\}\ e\ \{\bar{\Sigma}_2\}}{\{\bar{\Sigma}_1\}\ \textbf{region}\ r\ \textbf{in}\ e\ \{\bar{\Sigma}_2 - r\}}\ \text{CRegion}$$

Capacités : allocation

$$\frac{\{\bar{\Sigma}_1,\ \mathbf{own}(\mathcal{C}[\rho])\}\ e\ \{\bar{\Sigma}_2,\ \mathbf{own}(\mathcal{C}[\rho])\}}{\{\bar{\Sigma}_1\}\ \mathbf{new}\ \mathcal{C}[\rho](e)\ \{\bar{\Sigma}_2\}}\ \text{CAlloc}$$

Rappel :

class $\mathcal{C}$ $=$
  own $r,\cdots,r$
  τ
  inv$(x) = P$
end

Capacités : ouverture locale

$$\frac{e_1 : \mathcal{C}[\rho] \qquad \{\bar{\Sigma}_1\}\ e_1\ \{\bar{\Sigma}_2,\ \rho^{\times}\} \qquad \{\bar{\Sigma}_2,\ r^{\circ},\ \mathbf{own}(\mathcal{C}[r])\}\ e_2\ \{\bar{\Sigma}_3,\ r^{\circ},\ \mathbf{own}(\mathcal{C}[r])\}}{\{\bar{\Sigma}_1\}\ \mathbf{unpack}\ e_1\ \mathbf{as}\ x[r]\ \mathbf{in}\ e_2\ \{\bar{\Sigma}_3,\ \rho^{\times}\}}\ \text{CUnpack}$$

Rappel :

$$\frac{\Gamma \vdash e_1 : \mathcal{C}[\rho] \qquad \Gamma,\ x : \mathcal{C}[r] \vdash e_2 : \tau_2}{\Gamma \vdash \mathbf{unpack}\ e_1\ \mathbf{as}\ x[r]\ \mathbf{in}\ e_2 : \tau_2}\ \text{Unpack}$$

Appel de fonction : exemple

$$
\begin{array}{l}
\textbf{val } f(x\text{: } Long[\rho_1],\ y\text{: } Long[\rho_2])\text{: unit} \\
\quad \textbf{pre } \{\rho_1{}^{\times},\ \rho_2{}^{\times}\} \\
\quad \textbf{post } \{\rho_1{}^{\times},\ \rho_2{}^{\times}\}, \\
\qquad\quad !x = \textbf{old}(!x) + 1 \\
\qquad \wedge\ !y = \textbf{old}(!y) + 1 \\
\quad \textbf{assigns } \{\rho_1,\ \rho_2\} = \\
\qquad \textbf{unpack } x \textbf{ as } x[r] \textbf{ in } x := !x + 1; \\
\qquad \textbf{unpack } y \textbf{ as } y[r] \textbf{ in } y := !y + 1
\end{array}
$$

si ρ_1 et ρ_2 unifiés

- besoin de deux fois la même capacité
- appel de f impossible

Appel de fonction : typage

$$\frac{\begin{array}{c} f(\tau_1, \ \cdots, \ \tau_n) : \ \tau \\ \Gamma \vdash e_1 : \ \tau_1 \ \sigma \qquad \cdots \qquad \Gamma \vdash e_n : \ \tau_n \ \sigma \end{array}}{\Gamma \vdash f(e_1, \ \cdots, \ e_n) : \ \tau \ \sigma} \text{Call}$$

où σ est la substitution permettant d'instancier f

- substitution des variables de type polymorphes
- substitution des variables de région

Appel de fonction : capacités

$$\frac{\{\bar{\Sigma}_1\}\ e_1\ \{\bar{\Sigma}_2\} \qquad \cdots \qquad \{\bar{\Sigma}_n\}\ e_n\ \{\bar{\Sigma}\} \\ \bar{\Sigma} \ = \ \bar{\Sigma}'' \uplus \mathbf{pre}(f)\ \sigma \qquad \bar{\Sigma}' \ = \ \bar{\Sigma}'' \uplus \mathbf{post}(f)\ \sigma}{\{\bar{\Sigma}_1\}\ f(e_1, \cdots, e_n)\ \{\bar{\Sigma}'\}}\ \text{CCall}$$

où $\uplus$ est l'union des multi-ensembles

Sémantique

sémantique à petits pas

tas H : adresses $\rightarrow_\rho$ valeurs

affectation et déréférencement :

$$\begin{array}{rcl} H[p \mapsto v],\ p := v & \longrightarrow & H[p \mapsto v'],\ () \\ H[p \mapsto v],\ !p & \longrightarrow & H[p \mapsto v],\ v \end{array}$$

Sémantique : allocation

si p est une adresse fraiche :

$$H,\ \textbf{new}\ \mathcal{C}[\rho](v) \quad \longrightarrow \quad H[p \mapsto_\rho v],\ p$$

Sémantique : ouverture

$$H,\ \textbf{unpack}\ p\ \textbf{as}\ x[r]\ \textbf{in}\ v \quad\longrightarrow\quad H,\ v$$

règle particulière de passage au contexte :

$$\frac{H[p \mapsto_\rho v][p \mapsto_r v],\ e[p/x] \ \longrightarrow\ H'[p \mapsto_\rho v'][p \mapsto_r v'],\ e'[p/x]}{H[p \mapsto_\rho v],\ \textbf{unpack}\ p\ \textbf{as}\ x[r]\ \textbf{in}\ e \ \longrightarrow\ H'[p \mapsto_\rho v'],\ e'}$$

problème : affectation simultanée dans deux régions

- changer la sémantique de l'affectation ?
- imposer des capacités pour la lecture ?

Sémantique : appel de fonction

Définition (satisfaction du contrat d'une fonction)

$$H,\, H' \models f(v_1,\, \cdots,\, v_n) \longrightarrow v$$

si, et seulement si :

$$\begin{cases} H \models_{\mathbf{pre}} f(v_1,\, \cdots,\, v_n) \\ H,\, H' \models_{\mathbf{post}} f(v_1,\, \cdots,\, v_n) \longrightarrow v \\ H,\, H' \models_{\mathbf{assigns}} f \end{cases}$$

si $H,\, H' \models f(v_1,\, \cdots,\, v_n) \longrightarrow v$:

$$H,\ f(v_1,\ \cdots,\ v_n) \ \longrightarrow\ H',\ v$$

Correction

Théorème (Correction) Si :

$$\begin{cases} \Gamma \vdash e \colon \tau \\ \{\bar{\Sigma}_1\}\ e\ \{\bar{\Sigma}_2\} \\ coh(H, \bar{\Sigma}_1) \\ H, e \longrightarrow H', e' \end{cases}$$

alors il existe $\bar{\Sigma}_1$ et $\bar{\Sigma}_2$ tels que :

$$\begin{cases} \Gamma \vdash e' \colon \tau \\ \{\bar{\Sigma}_1{}'\}\ e'\ \{\bar{\Sigma}_2{}'\} \\ coh(H', \bar{\Sigma}_1{}') \end{cases}$$

Correction des définitions de fonction

Corollaire (Correction d'une fonction) Soit une fonction f définie par :

$$\textbf{val}\ f(x_1 \colon \tau_1, \cdots, x_n \colon \tau_n) \colon \tau \cdots = e$$

et soit un tas H et des valeurs $v_1, \cdots, v_n$ de types respectifs $\tau_1, \cdots, \tau_n$, tels que $H \models_{\mathbf{pre}} f(v_1, \cdots, v_n)$. Pour toute valeur r de type τ telle que $e[v_1/x_1, \cdots, v_n/x_n] \longrightarrow^* r$, on a :

$$H, H' \models f(v_1, \cdots, v_n) \longrightarrow r$$

Abstraction

cacher des effets[1]

modules à la Caml

abstraction du contrat d'une fonction

- types
- pré-condition et ses capacités
- post-condition et ses capacités
- clause assigns

[1] uniquement les effets personnels, pas les feutres pour tableau blanc

Abstraction : exemple (version concrète)

$$
\begin{array}{l}
\textbf{class } \mathit{Morgan} = \\
\quad \textbf{own } \rho_{sum}, \rho_{count} \\
\quad (\mathit{Long}[\rho_{sum}], \mathit{Long}[\rho_{count}], \mathit{set}) \\
\quad \textbf{inv}(x) = \\
\qquad\quad !x.1 = \mathit{setsum}(x.3) \\
\qquad \wedge\ !x.2 = \mathit{setcount}(x.3) \\
\textbf{end}
\end{array}
$$

$$
\begin{array}{l}
\textbf{val } \mathit{average}(\mathit{morgan}\colon \mathit{Morgan}[\rho])\colon \mathit{Long}[\rho'] \\
\quad \textbf{pre } \{\rho^{\times}\}, !(!\mathit{morgan}).2 > 0 \\
\quad \textbf{post } \{\rho^{\times}\}, !\textbf{result} = !(!\mathit{morgan}).1 \ / \ !(!\mathit{morgan}).2 \\
\quad = \textbf{new } \mathit{Long}[\rho'] \ (!(!\mathit{morgan}).1 \ / \ !(!\mathit{morgan}).2)
\end{array}
$$

Abstraction : exemple (version abstraite)

class *Morgan* =
  set
end

val *average*(*morgan*: *Morgan*[ρ]): *Long*[ρ']
  pre $\{\rho^\times\}$, *setcount*(!*morgan*) > 0
  post $\{\rho^\times\}$, !**result** $=$ *setsum*(!*morgan*) / *setcount*(!*morgan*)

Abstraction : exemple (insertion)

$$
\begin{array}{l}
\textbf{val } \mathit{insert}(\mathit{morgan}\colon \mathit{Morgan}[\rho],\ n\colon \mathit{Long}[\rho'])\colon \mathrm{unit} \\
\quad \textbf{pre } \{\rho^{\times}\} \\
\quad \textbf{post } \{\rho^{\times}\}, \\
\qquad\quad !(!\mathit{morgan}).1 = \textbf{old}(!(!\mathit{morgan}).1) + !n \\
\qquad \wedge\ !(!\mathit{morgan}).2 = \textbf{old}(!(!\mathit{morgan}).2) + 1 \\
\quad \textbf{assigns } \{\rho.\rho_{\mathit{sum}},\ \rho.\rho_{\mathit{count}}\} = \cdots
\end{array}
$$

$$
\begin{array}{l}
\textbf{val } \mathit{insert}(\mathit{morgan}\colon \mathit{Morgan}[\rho],\ n\colon \mathit{Long}[\rho'])\colon \mathrm{unit} \\
\quad \textbf{pre } \{\rho^{\times}\} \\
\quad \textbf{post } \{\rho^{\times}\},\ !\mathit{morgan} = \mathit{setadd}(!\mathit{morgan},\ !n) \\
\quad \textbf{assigns } \emptyset = \cdots
\end{array}
$$

Abstraction : obligation de preuve

$$
\begin{array}{l}
\forall m_a,\ \mathbf{pre}_a(m_a) \Rightarrow \\
\quad \exists m,\ \mathbf{pre}(m) \land \\
\quad\quad \exists m',\ \mathbf{post}(m, m') \Rightarrow \\
\quad\quad\quad \exists {m_a}',\ \mathbf{post}_a(m_a, {m_a}')
\end{array}
$$

obligation de preuve

- modulo les régions
- **pre** et **post** sous-entendent aussi les invariants
- avec parfois $m = m'$ et/ou $m_a = {m_a}'$

Abstraction : correction

instance (?) de la règle de conséquence :

$$\frac{\{P \;\Rightarrow\; P',\; P'\}\; e\; \{Q',\; Q' \;\Rightarrow\; Q\}}{\{P\}\; e\; \{Q\}}$$

Inférence

inférables :

- ouvertures locales (**unpack** $\cdots$ **in**)
- capacités ouvertes

non-inférables :

- clauses **assigns**
- égalités des régions
- capacités fermées

discutable :

- portée des ouverture locales

Conclusion

syntaxe et typage d'un système d'invariants à base de capacités avec sa sémantique et des possibilités d'abstractions

adaptable pour C, Java, ...

lien avec Spec# assez clair

Yapuka

faire les preuves

mieux formaliser l'inférence

étendre avec régions singletons et existentielles

implémenter (dans Jessie ?)

Pub

Essayez Melt pour écrire vos articles et présentations avec :

- la puissance de mise en forme de LaTeX ;
- l'expressivité de Caml !

https://forge.ocamlcore.org/projects/melt/