Plan

Plan

Plan

Plan

Plan

Plan

Plan

Plan

Plan

Plan

Plan

Plan

Plan

Plan

Plan

Plan

Plan

Plan

Plan

Un petit coup d'oeil au code Caml... (1)

```
type varName = int
type progPoint = int

type op =
  | Add
  | Sub
  | Mult

type expr =
  | Const of int
  | Unknown
  | Var of varName
  | Minus of expr
  | Numop of op * expr * expr

type comp =
  | Eq | Neq
  | Lt | Le
  | Gt | Ge
```

```
type test =
  | Numcomp of comp * expr * expr
  | Not of test
  | And of test * test
  | Or of test * test

type instr =
  | Affect of varName * expr
  | If of test * block * block
  | While of test * block
and block =
  | Empty of progPoint
  | Seq of progPoint * instr * block

type program = {
  instrs : block;
  vars : varName list;
  tabvar : (varName, string) Hashtbl.t
}
```

While : Abstraction des états

Pour tout abstraction des environnements,

$$\left(\mathcal{P}(\mathrm{Env}), \subseteq, \bigcup, \bigcap\right) \underset{\alpha_{\mathrm{Env}}}{\overset{\gamma_{\mathrm{Env}}}{\leftrightarrows}} \left(\mathrm{Env}^\sharp, \sqsubseteq^\sharp_{\mathrm{Env}}, \bigsqcup^\sharp_{\mathrm{Env}}, \bigsqcap^\sharp_{\mathrm{Env}}\right)$$

nous définissons une abstraction correcte de la sémantique collectrice

$$[\![P]\!]^\sharp \in \mathbb{P} \to \mathrm{Env}^\sharp$$

qui doit vérifier

$$\forall k,\ \alpha_{\mathrm{Env}}([\![P]\!]^{\mathrm{col}}_k) \sqsubseteq^\sharp_{\mathrm{Env}} [\![P]\!]^\sharp_k$$

ou de manière équivalente

$$\forall k,\ [\![P]\!]^{\mathrm{col}}_k \subseteq \gamma_{\mathrm{Env}}\left([\![P]\!]^\sharp_k\right)$$

Abstraction des environnements : éléments suffisants

Grâce aux théorèmes précédents, il nous suffit de trouver

- $\mathrm{init}^\sharp_{\mathrm{Env}}$ une approximation correcte de $\mathrm{Env} \in \mathcal{P}(\mathrm{Env})$

$$\mathrm{Env} \subseteq \gamma_{\mathrm{Env}}(\mathrm{init}^\sharp_{\mathrm{Env}})$$

- $[\![x := e]\!]^\sharp \in \mathrm{Env}^\sharp \to \mathrm{Env}^\sharp$ une approximation correcte de $[\![x := e]\!]$

$$\forall \rho^\sharp,\ [\![x := e]\!](\gamma_{\mathrm{Env}}(\rho^\sharp)) \subseteq \gamma_{\mathrm{Env}}([\![x := e]\!]^\sharp(\rho^\sharp))$$

- $[\![\mathbf{assert}\ t]\!]^\sharp \in \mathrm{Env}^\sharp \to \mathrm{Env}^\sharp$ une approximation correcte de $[\![\mathbf{assert}\ t]\!]$.

$$\forall \rho^\sharp,\ [\![\mathbf{assert}\ t]\!](\gamma_{\mathrm{Env}}(\rho^\sharp)) \subseteq \gamma_{\mathrm{Env}}([\![\mathbf{assert}\ t]\!]^\sharp(\rho^\sharp))$$

Preuve

Un petit coup d'oeil au code Caml... (2)

```
module type Lattice =
 sig
  type t

  val eq_dec : t → t → bool

  val order_dec : t → t → bool

  val join : t → t → t

  val meet : t → t → t

  val widen : t → t → t

  val narrow : t → t → t

  val bottom : unit → t
 end
```

```
module type EnvAbstraction =
 sig
  module L : Lattice

  val affect :
         program →
         L.t → varName → expr → L.t

  val init_env : program → L.t

  val back_test :
         program → test → L.t → L.t

  val to_string :
         program → L.t → string
 end
```

Un petit coup d'oeil au code Caml... (3)

```
module Solve =
  functor (AbEnv:EnvAbstraction) →
  struct

    let analyse p = ...

  end
```

Nous pouvons ainsi programmer un interpréteur abstrait paramétré par n'importe quelle abstraction d'environnements.

Les grandes étapes

- Construction d'un système d'équations,

$$
\begin{array}{lcl}
X_0 & = & \mathbf{init}^{\sharp}_{\mathrm{Env}} \\
X_1 & = & [\![X := ?]\!]^{\sharp}(X_0) \\
X_2 & = & [\![\mathbf{assert}\ X < 0]\!]^{\sharp}(X_1) \sqcup^{\sharp} X_4 \\
X_3 & = & [\![\mathbf{assert}\ X < 0]\!]^{\sharp}(X_2) \\
X_4 & = & [\![X := X + 1]\!]^{\sharp}(X_3) \\
X_5 & = & [\![\mathbf{assert}\ X \geq 0]\!]^{\sharp}(X_2) \\
X_6 & = & [\![Y := X]\!]^{\sharp}(X_5) \\
X_7 & = & [\![\mathbf{assert}\ X \geq 0]\!]^{\sharp}(X_1) \\
X_8 & = & [\![Y := 0]\!]^{\sharp}(X_7) \\
X_9 & = & X_6 \sqcup^{\sharp} X_8
\end{array}
$$

- Résolution du plus petit point fixe par itération (cf cours théorie des treillis)

Plan

Abstraction non-relationnelle des environnements

Pour tout abstraction numérique (plus quelques opérateurs à préciser...),

$$\left(\mathcal{P}(\mathrm{Num}), \subseteq, \bigcup, \bigcap\right) \underset{\alpha_{\mathrm{Num}}}{\overset{\gamma_{\mathrm{Num}}}{\leftrightarrows}} \left(\mathrm{Num}^\sharp, \sqsubseteq^\sharp_{\mathrm{Num}}, \bigsqcup^\sharp_{\mathrm{Num}}, \bigsqcap^\sharp_{\mathrm{Num}}\right)$$

nous définissons une abstraction sur les environnements

$$\left(\mathcal{P}(\mathbb{V} \to \mathrm{Num}), \subseteq, \bigcup, \bigcap\right) \underset{\alpha_{\mathrm{Env}}}{\overset{\gamma_{\mathrm{Env}}}{\leftrightarrows}} \left(\mathrm{Env}^\sharp, \sqsubseteq^\sharp_{\mathrm{Env}}, \bigsqcup^\sharp_{\mathrm{Env}}, \bigsqcap^\sharp_{\mathrm{Env}}\right)$$

en prenant

$$\begin{aligned}
\mathrm{Env}^\sharp &\stackrel{\mathrm{def}}{=} \mathbb{V} \longrightarrow \mathrm{Num}^\sharp \\
\forall \rho_1^\sharp, \rho_2^\sharp \in \mathrm{Env}^\sharp,\ \rho_1^\sharp \sqsubseteq^\sharp_{\mathrm{Env}} \rho_2^\sharp &\stackrel{\mathrm{def}}{=} \forall x \in \mathbb{V},\ \rho_1^\sharp(p) \sqsubseteq^\sharp_{\mathrm{Num}} \rho_2^\sharp(p) \\
\forall S \subseteq \mathrm{Env}^\sharp,\ \bigsqcup^\sharp_{\mathrm{Env}} S &\stackrel{\mathrm{def}}{=} \bigsqcup^\sharp_{\mathrm{Num}} \{\, \rho^\sharp(p) \mid \rho^\sharp \in S \,\} \\
\forall S \subseteq \mathrm{Env},\ \alpha_{\mathrm{Env}}(S) &\stackrel{\mathrm{def}}{=} \lambda x.\ \alpha_{\mathrm{Num}}(\{\, \rho(x) \mid \rho \in S \,\}) \\
\forall \rho^\sharp \in \mathrm{Env}^\sharp,\ \gamma_{\mathrm{Env}}(\rho^\sharp) &\stackrel{\mathrm{def}}{=} \{\, \rho \mid \forall x \in \mathbb{V},\ \rho(x) \in \gamma_{\mathrm{Num}}(\rho^\sharp(x)) \,\}
\end{aligned}$$

Construction de $[\![x := e]\!]^\sharp$

$$[\![x := e]\!]^\sharp(\rho^\sharp) = \rho^\sharp\left[x \mapsto \mathcal{A}[\![e]\!]^\sharp(\rho^\sharp)\right], \forall \rho^\sharp \in \mathrm{Env}^\sharp$$

avec

$$\forall e \in \mathrm{Expr},\ \mathcal{A}[\![e]\!]^\sharp \in \mathrm{Env}^\sharp \rightarrow \mathrm{Num}^\sharp$$

une évaluation (en avant) abstraite des expressions

$$\begin{array}{rcl}
\mathcal{A}[\![n]\!]^\sharp(\rho^\sharp) & = & \mathbf{const}^\sharp(n) \\
\mathcal{A}[\![?]\!]^\sharp(\rho^\sharp) & = & \top_{\mathrm{Num}} \\
\mathcal{A}[\![x]\!]^\sharp(\rho^\sharp) & = & \rho^\sharp(x) \\
\mathcal{A}[\![e_1\ o\ e_2]\!]^\sharp(\rho^\sharp) & = & o^\sharp\left(\mathcal{A}[\![e_1]\!]^\sharp(\rho^\sharp), \mathcal{A}[\![e_2]\!]^\sharp(\rho^\sharp)\right)
\end{array}$$

Opérateurs requis sur l'abstraction numérique

- $\mathrm{const}^\sharp \in \mathrm{Num} \to \mathrm{Num}^\sharp$ calcule une approximation des constantes

$$\forall n \in \mathbb{Z},\ \{n\} \subseteq \gamma_{\mathrm{Num}}(\mathrm{const}^\sharp(n))$$

- $\top_{\mathrm{Num}} \in \mathrm{Num}^\sharp$ approche n'importe quelle valeur numérique

$$\mathbb{Z} \subseteq \gamma_{\mathrm{Num}}(\top_{\mathrm{Num}})$$

- $o^\sharp \in \mathrm{Num}^\sharp \times \mathrm{Num}^\sharp \to \mathrm{Num}^\sharp$ est une approximation de l'opérateur arithmétique $o \in \{+, -, \times\}$

$$\forall n_1^\sharp, n_2^\sharp \in \mathrm{Num}^\sharp,$$
$$\{\, n_1 \ \bar{o}\ n_2 \mid n_1 \in \gamma_{\mathrm{Num}}(n_1^\sharp),\ n_2 \in \gamma_{\mathrm{Num}}(n_2^\sharp) \,\} \subseteq \gamma_{\mathrm{Num}}(o^\sharp(n_1^\sharp, n_2^\sharp))$$

Construction de $[\![\textbf{assert } t]\!]^\sharp$

$$
\begin{array}{rcl}
[\![\textbf{assert } e_1\ c\ e_2]\!]^\sharp(\rho^\sharp) & = & \left([\![e_1]\!]{\downarrow}^\sharp_{\text{expr}}\ (\rho^\sharp, n_1^\sharp) \sqcap^\sharp_{\text{Env}} [\![e_2]\!]{\downarrow}^\sharp_{\text{expr}}\ (\rho^\sharp, n_2^\sharp)\right) \\
 & & \text{avec } (n_1^\sharp, n_2^\sharp) = c^\sharp\ (\mathcal{A}[\![e_1]\!]^\sharp(\rho^\sharp), \mathcal{A}[\![e_2]\!]^\sharp(\rho^\sharp)) \\
[\![\textbf{assert } t_1 \textbf{ and } t_2]\!]^\sharp(\rho^\sharp) & = & \left([\![\textbf{assert } t_1]\!]^\sharp \sqcap^\sharp_{\text{Env}} [\![\textbf{assert } t_2]\!]^\sharp\right)
\end{array}
$$

- $c^\sharp \in \text{Num}^\sharp \times \text{Num}^\sharp \to \text{Num}^\sharp \times \text{Num}^\sharp$ calcule un raffinement de deux valeurs numériques abstraites, sachant qu'elles vérifient une condition c
- $[\![e]\!]{\downarrow}^\sharp_{\text{expr}} \in \text{Env}^\sharp \times \text{Num}^\sharp \to \text{Env}^\sharp$: $[\![e]\!]{\downarrow}^\sharp_{\text{expr}}\ (\rho^\sharp, n^\sharp)$ calcule un raffinement de l'environnement abstrait $\rho^\sharp$, sachant que l'expression e s'évalue par la valeur numérique abstraite $n^\sharp$ dans cet environnement

Opérateurs requis sur l'abstraction numérique

$$\left\{(n_1,n_2) \mid n_1 \in \gamma_{\mathrm{Num}}(n_1^\sharp),\ n_2 \in \gamma_{\mathrm{Num}}(n_2^\sharp),\ n_1 [\![c]\!]\!\uparrow_{\mathrm{comp}} n_2 \right\}$$
$$\subseteq \gamma_{\mathrm{Num}}(m_1^\sharp) \times \gamma_{\mathrm{Num}}(m_2^\sharp)$$
$$\text{avec } (m_1^\sharp, m_2^\sharp) = [\![c]\!]\!\downarrow^\sharp_{\mathrm{comp}} (n_1^\sharp, n_2^\sharp)$$

$$\begin{array}{rcl}
[\![n]\!]\!\downarrow^\sharp_{\mathrm{expr}} (\rho^\sharp, n^\sharp) & = & \begin{cases} \bot_{\mathrm{Env}} & \text{si } \mathrm{const}^\sharp(n) \sqcap^\sharp_{\mathrm{Num}} n^\sharp = \bot_{\mathrm{Num}} \\ \rho^\sharp & \text{sinon} \end{cases} \\
[\![?]\!]\!\downarrow^\sharp_{\mathrm{expr}} (\rho^\sharp, n^\sharp) & = & \rho^\sharp \\
[\![x]\!]\!\downarrow^\sharp_{\mathrm{expr}} (\rho^\sharp, n^\sharp) & = & (\rho^\sharp[x \mapsto \rho^\sharp(x) \sqcap^\sharp_{\mathrm{Num}} n^\sharp]) \\
[\![-e]\!]\!\downarrow^\sharp_{\mathrm{expr}} (\rho^\sharp, n^\sharp) & = & [\![e]\!]\!\downarrow^\sharp_{\mathrm{expr}} (\rho^\sharp, -^\sharp(n^\sharp)) \\
[\![e_1\, o\, e_2]\!]\!\downarrow^\sharp_{\mathrm{expr}} (\rho^\sharp, n^\sharp) & = & \left([\![e_1]\!]\!\downarrow^\sharp_{\mathrm{expr}} (\rho^\sharp, n_1^\sharp) \sqcap^\sharp_{\mathrm{Env}} [\![e_2]\!]\!\downarrow^\sharp_{\mathrm{expr}} (\rho^\sharp, n_2^\sharp)\right)
\end{array}$$
$$\text{avec } (n_1^\sharp, n_2^\sharp) = [\![o]\!]\!\downarrow^\sharp_{\mathrm{op}} (n^\sharp, \mathcal{A}[\![e_1]\!]^\sharp(\rho^\sharp), \mathcal{A}[\![e_2]\!]^\sharp(\rho^\sharp))$$

Opérateurs requis sur l'abstraction numérique

$$[\![o]\!]{\downarrow}^{\sharp}_{\mathrm{op}} \in \mathrm{Num}^{\sharp} \times \mathrm{Num}^{\sharp} \times \mathrm{Num}^{\sharp} \rightarrow \mathrm{Num}^{\sharp} \times \mathrm{Num}^{\sharp}$$

$[\![o]\!]{\downarrow}^{\sharp}_{\mathrm{op}}\ (n^{\sharp}, n_1^{\sharp}, n_2^{\sharp})$ calcule un raffinement de deux valeurs numériques $n_1^{\sharp}$ et $n_2^{\sharp}$ sachant que le résultat de l'opération binaire o vaut $n^{\sharp}$ sur elles

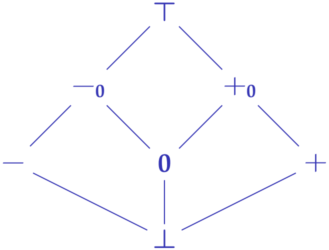
$$\begin{array}{l}
\forall n^{\sharp}, n_1^{\sharp}, n_2^{\sharp} \in \mathrm{Num}^{\sharp}, \\
\quad \left\{ (n_1, n_2) \mid n_1 \in \gamma_{\mathrm{Num}}(n_1^{\sharp}),\ n_2 \in \gamma_{\mathrm{Num}}(n_2^{\sharp}),\ (n_1 o n_2) \in \gamma_{\mathrm{Num}}(n^{\sharp}) \right\} \\
\qquad \subseteq \gamma_{\mathrm{Num}}(m_1^{\sharp}) \times \gamma_{\mathrm{Num}}(m_2^{\sharp}) \\
\qquad\qquad\qquad\qquad \text{avec } (m_1^{\sharp}, m_2^{\sharp}) = [\![o]\!]{\downarrow}^{\sharp}_{\mathrm{op}}\ (n^{\sharp}, n_1^{\sharp}, n_2^{\sharp})
\end{array}$$

Un petit coup d'oeil au code Caml... (4)

```
module type NumAbstraction =
 sig
  module L : Lattice

  val backTest : comp → L.t → L.t → L.t * L.t

  val minus : L.t → L.t

  val semOp : op → L.t → L.t → L.t

  val back_semOp : op → L.t → L.t → L.t → L.t * L.t

  val const : int → L.t

  val top : L.t

  val to_string : string → L.t → string
 end

module EnvNotRelational = functor (AN:NumAbstraction) →
 (struct ... end : EnvAbstraction)
```

Plan

Abstraction des signes



$$
\begin{array}{rcl}
\gamma_{\text{Num}}(\bot) & = & \emptyset \\
\gamma_{\text{Num}}(-) & = & \{\, z \mid z < \mathbf{0} \,\} \\
\gamma_{\text{Num}}(\mathbf{0}) & = & \{\mathbf{0}\} \\
\gamma_{\text{Num}}(+) & = & \{\, z \mid z > \mathbf{0} \,\} \\
\gamma_{\text{Num}}(-_{\mathbf{0}}) & = & \{\, z \mid z \leq \mathbf{0} \,\} \\
\gamma_{\text{Num}}(+_{\mathbf{0}}) & = & \{\, z \mid z \geq \mathbf{0} \,\} \\
\gamma_{\text{Num}}(\top) & = & \mathbb{Z}
\end{array}
$$

Exercice : donner les opérateurs abstraits de l'abstraction numérique par signe

Un petit coup d'oeil au code Caml... (5)

```
module SignAbNum : NumAbstraction = struct ... end

module S = Solve(EnvNotRelational(SignAbNum))

let analyse file =
  let p = parse_prog file in S.analyse p
```

Défaut d'injectivité de γ_{Env}

γ_{Env} n'est pas injective lorsque $\gamma_{\text{Num}}(\bot_{\text{Num}}) = \emptyset$.
Exemple :

$$\gamma_{\text{Env}}([x \mapsto \bot,\ y \mapsto \top]) = \gamma_{\text{Env}}([x \mapsto \top,\ y \mapsto \bot]) = \emptyset$$

Cela peut occasionner des imprécisions si on n'y prend pas garde.

- solution : opérateur de réduction

Utilisation d'un opérateur de réduction

L'opérateur $\mathrm{reduc}^\sharp \in \mathrm{Env}^\sharp \to \mathrm{Env}^\sharp$ permet de détecter ce type d'environnements abstraits et de les remplacer par $\bot_{\mathrm{Env}}$

$$\mathrm{reduc}^\sharp(\rho^\sharp) = \begin{cases} \bot_{\mathrm{Env}} & \text{si } \exists x \in \mathbb{V},\ \rho^\sharp(x) = \bot_{\mathrm{Num}} \\ \rho^\sharp & \text{sinon} \end{cases}$$

Utilisation

- chaque calcul sur $\mathrm{Env}^\sharp$ est suivi d'une utilisation de $\mathrm{reduc}^\sharp$ afin de rester dans l'ensemble $\mathrm{reduc}^\sharp(\mathrm{Env}^\sharp)$ sur lequel γ_{Env} est injectif
- on peut parfois spécialiser l'opérateur $\mathrm{reduc}^\sharp \circ f^\sharp$ pour chaque opérateur abstrait $f^\sharp$

Opérateur de réduction optimal

Pour être correct un opérateur de réduction $\rho \in L^\sharp \to L^\sharp$ doit être conservatif

$$\forall x,\ \gamma(x) \sqsubseteq \gamma(\rho(x))$$

Pour être utile, en terme de précision, ρ doit être réductif

$$\forall x,\ \rho(x) \sqsubseteq x$$

et la réduction doit être optimale (ρ idempotent)

$$\rho \circ \rho = \rho$$

Définition (fermeture inférieure)

Un opérateur $\rho \in L \to L$ est une fermeture inférieure sur un treillis $(L, \sqsubseteq, \sqcup, \sqcap)$ si il est monotone, réductif et idempotent.

Un autre joli théorème

Théorème

Si $\alpha : L \rightarrow L^\sharp, \gamma : L^\sharp \rightarrow L$ *est une connexion de Galois, l'opérateur* $\rho = \alpha \circ \gamma$ *est une fermeture inférieure qui vérifie*

$$\gamma \circ \rho = \gamma$$

$\alpha \circ \gamma$ fourni donc un opérateur de réduction optimal.

Plan

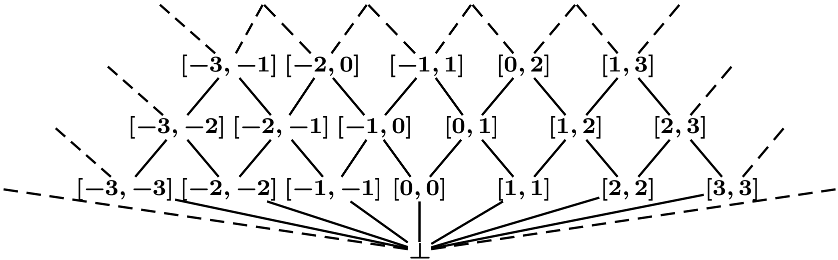
Abstraction par intervalle

$$\mathrm{Int} \stackrel{\text{def}}{=} \{\, [a, b] \mid a, b \in \overline{\mathbb{Z}},\ a \leqslant b \,\} \cup \{\bot\}$$

avec $\overline{\mathbb{Z}} = \mathbb{Z} \cup \{-\infty, +\infty\}$.

Exercice : donner les différents éléments de cette abstraction.

Problème de convergence



Un tel treillis ne vérifie pas la condition de chaîne ascendante.

Solution : approximation dynamique

- on extrapole la limite avec un op. d'élargissement ∇

  Idée : $\underset{n}{[-3,3]} \quad \nabla \quad \underset{n+1}{[-5,3]} \quad = \quad \underset{\text{extrapolation}}{[-\infty,3]}$

Approximation de point fixe

Lemma

Soit $(A, \sqsubseteq, \sqcup, \sqcap)$ *un treillis complet et* f *un opérateur monotone sur* A*. Si* a *est un post-point fixe de* f $(f(a) \sqsubseteq a)$*, alors* $\mathrm{lfp}(f) \sqsubseteq a$*.*

La décision de calculer une sur-approximation de $\mathrm{lfp}(f)$ peut être prise dans plusieurs cas :

- Le treillis ne vérifie pas la condition de chaîne ascendante, l'itération $\bot, f(\bot), \ldots, f^n(\bot), \ldots$ peut ne jamais terminer.
- La condition de chaîne ascendante est vérifiée mais la chaîne d'itération est trop longue pour permettre un calcul efficace.
- Enfin, certaines abstractions ne vérifient pas « l'hypothèse raisonnable » décrit précédemment. Le treillis sous-jacent n'est alors pas complet et la limite des itérations croissantes n'appartient pas au domaine d'abstraction.

Élargissement

Idée : on a une suite croissante

$$x^0 = \bot, x^{n+1} = F(x^n) \qquad = x^n \sqcup F(x^n)$$

On voudrait la remplacer par quelque chose comme

$$y^0 = \bot, y^{n+1} = y^n \nabla F(y^n)$$

de telle sorte que (y^n) soit croissante (i), $x^n \sqsubseteq y^n$ (ii), et (y^n) converge en nombre fini d'étape (iii), avec ∇ indépendant de F.

Élargissement : opérateur $\nabla : L \times L \to L$ tel que

- $\forall x, x' \in L, x \sqcup x' \sqsubseteq x \nabla x'$ (implique (i) & (ii))
- Si $x^0 \sqsubseteq x^1 \sqsubseteq \ldots$ séquence croissante, alors $y^0 = x^0, y^{n+1} = y^n \nabla x^{n+1}$ n'est pas strictement croissante *i.e.* converge en un nombre fini d'étapes (implique (iii)).

En anglais , on parle de *widening*.

Utilisation : on remplace $x^0 = \bot, x^{n+1} = F(x^n)$
par $y^0 = \bot, y^{n+1} = y^n \nabla F(y^n)$

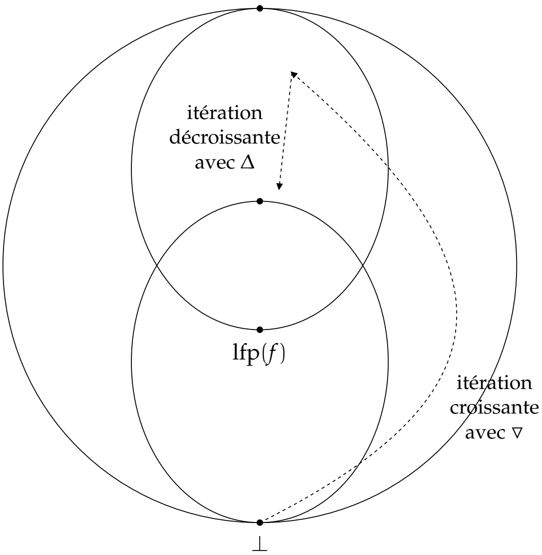
Élargissement : théorème

Théorème

Soit L *est un treillis complet,* $F : L \rightarrow L$ *une fonction monotone, et* $\nabla : L \times L \rightarrow L$ *un opérateur d'élargissement. La suite* $y^0 = \bot, y^{n+1} = y^n \nabla F(y^n)$ *converge en un nombre fini d'étapes vers un post-point-fixe* y *de* F.

Conséquence immédiate : on a $\mathrm{lfp}(F) \sqsubseteq y$.

Schéma


itération
décroissante
avec Δ
$\mathrm{lfp}(f)$
itération
croissante
avec ∇
$\perp$

Exemple : élargissement sur les intervalles

Idée : dès qu'une borne supérieure croit (ou qu'une borne inférieure décroit), on extrapole par $+\infty$ (ou $-\infty$). Après une telle extrapolation, la borne en question ne peut plus bouger.

Définition :

$$
\begin{array}{rcl}
[a,b]\nabla[a',b'] & = & [\quad \textbf{if } a' < a \textbf{ then } -\infty \textbf{ else } a, \\
 & & \quad\;\; \textbf{if } b' > b \textbf{ then } +\infty \textbf{ else } b\,] \\
\perp^\sharp\nabla[a',b'] & = & [a',b']
\end{array}
$$

Exemples :
$[-3,4]\nabla[-3,2] = [-3,4]$
$[-3,4]\nabla[-3,5] = [-3,+\infty]$

Amélioration d'un approximation de point fixe

Théorème

Soit $(A, \sqsubseteq, \sqcup, \sqcap)$ *un treillis complet,* f *un opérateur monotone sur* A *et* a *un post-point fixe de* f*. La chaîne* $(x_n)_n$ *définie par* $\left\{\begin{array}{lcl} x_0 & = & a \\ x_{k+1} & = & f(x_k) \end{array}\right.$ *admet pour limite* ($\bigsqcup\{x_n\}$) *le plus grand point fixe de* f *plus petit que* a *(noté* $\mathrm{gfp}_a(f)$*). En particulier,* $\mathrm{lfp}(f) \sqsubseteq \bigsqcup\{x_n\}$*. Chaque étape intermédiaire de calcul est une approximation correcte :* $\forall k,\ \mathrm{lfp}(f) \sqsubseteq \mathrm{gfp}_a(f) \sqsubseteq x_k \sqsubseteq a$

Théorème

Si Δ *est un opérateur de rétrécissement sur un ensemble partiellement ordonné* $(A, \sqsubseteq)$*, si* f *est un opérateur monotone sur* A *et* a *un post-point fixe de* f *alors la chaîne* $(x_n)_n$ *définie par* $\left\{\begin{array}{lcl} x_0 & = & a \\ x_{k+1} & = & x_k \nabla f(x_k) \end{array}\right.$ *atteint en un nombre fini de pas un post-point fixe de* f *plus petit que* a*.*

Opérateur de rétrécissement sur les intervalles

$$[a,b]\Delta[c,d] = \begin{array}{lll} [& \text{si } a = -\infty \text{ alors } c \text{ sinon } a & ; \\ & \text{si } b = +\infty \text{ alors } d \text{ sinon } b &] \end{array}$$

Intuition : on améliore uniquement les bornes infinis.

En pratique : quelques itération améliorent déjà bien le résultat atteint après élargissement.

- les affectations par des constantes et les tests sur des conditions rendent la séquence descendante efficace : elles *filtrent* les (trop grandes) approximations calculées par l'élargissement

Plan

Produit réduit

Supposons que l'on dispose de deux connexions de Galois

$$\left(\mathcal{A}, \sqsubseteq, \bigsqcup, \bigsqcap\right) \underset{\alpha_1}{\overset{\gamma_1}{\leftrightarrows}} \left(\mathcal{A}^\sharp_1, \sqsubseteq^\sharp_1, \bigsqcup{}^\sharp_1, \bigsqcap{}^\sharp_1\right) \text{ et } \left(\mathcal{A}, \sqsubseteq, \bigsqcup, \bigsqcap\right) \underset{\alpha_2}{\overset{\gamma_2}{\leftrightarrows}} \left(\mathcal{A}^\sharp_2, \sqsubseteq^\sharp_2, \bigsqcup{}^\sharp_2, \bigsqcap{}^\sharp_2\right)$$

pour abstraire un même domaine concret $\mathcal{A}$. Nous pouvons combiner ces deux abstractions avec une nouvelle connexion

$$\left(\mathcal{A}, \sqsubseteq, \bigsqcup, \bigsqcap\right) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \left(\mathcal{A}^\sharp_1 \times \mathcal{A}^\sharp_2, \sqsubseteq^\sharp, \bigsqcup{}^\sharp, \bigsqcap{}^\sharp\right)$$

où

$$(a^\sharp_1, a^\sharp_2) \sqsubseteq^\sharp (b^\sharp_1, b^\sharp_2) \iff a^\sharp_1 \sqsubseteq^\sharp_1 b^\sharp_1 \wedge a^\sharp_2 \sqsubseteq^\sharp_2 b^\sharp_2, \quad \forall (a^\sharp_1, a^\sharp_2), (b^\sharp_1, b^\sharp_2) \in \mathcal{A}^\sharp_1 \times \mathcal{A}^\sharp_2$$
$$\ldots$$

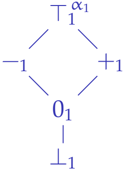
La nouvelle connexion ainsi obtenue est cependant rarement une insertion, même si (α_1, γ_1) et (α_2, γ_2) le sont.

Exemple

Si on combine l'insertion de Galois des signes

$$(\mathcal{P}(\mathbb{Z}), \subseteq, \bigcup, \bigcap) \underset{\alpha_1}{\overset{\gamma_1}{\leftrightarrows}} \left(\{\bot_1, 0_1, -_1, +_1, \top_1\}, \sqsubseteq_1^\sharp, \bigsqcup_1^\sharp, \bigsqcap_1^\sharp\right)$$

$$\begin{array}{rcl} \gamma_1(\bot_1) & = & \emptyset \\ \gamma_1(0_1) & = & \{0\} \\ \gamma_1(-_1) & = & \mathbb{Z}^- \\ \gamma_1(+_1) & = & \mathbb{Z}^+ \\ \gamma_1(\top_1) & = & \mathbb{Z} \end{array}$$

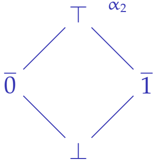


$$\alpha_1(P) = \begin{cases} \bot_1 & \text{si} \quad P = \emptyset \\ 0_1 & \text{si} \quad P = \{0\} \\ -_1 & \text{si} \quad P \not\subseteq \{0\} \text{ et } P \subseteq \mathbb{Z}^- \\ +_1 & \text{si} \quad P \not\subseteq \{0\} \text{ et } P \subseteq \mathbb{Z}^+ \\ \top_1 & \text{si} \quad P \not\subseteq \mathbb{Z}^- \text{ et } P \not\subseteq \mathbb{Z}^+ \end{cases}$$

avec l'insertion des parités

$$(\mathcal{P}(\mathbb{Z}), \subseteq, \bigcup, \bigcap) \underset{\alpha_2}{\overset{\gamma_2}{\leftrightarrows}} \left(\{\bot_2, 0_2, 1_2, \top_2\}, \sqsubseteq_2^\sharp, \bigsqcup_2^\sharp, \bigsqcap_2^\sharp\right)$$

$$\begin{array}{rcl} \gamma_2(\bot_2) & = & \emptyset \\ \gamma_2(0_2) & = & \mathbb{Z}_0 \\ \gamma_2(1_2) & = & \mathbb{Z}_1 \\ \gamma_2(\top_2) & = & \mathbb{Z} \end{array}$$



$$\alpha_2(P) = \begin{cases} \bot_2 & \text{si} \quad P = \emptyset \\ 0_2 & \text{si} \quad P \neq \emptyset \text{ et } P \subseteq \mathbb{Z}_0 \\ 1_2 & \text{si} \quad P \neq \emptyset \text{ et } P \subseteq \mathbb{Z}_1 \\ \top_2 & \text{si} \quad P \not\subseteq \mathbb{Z}_0 \text{ et } P \not\subseteq \mathbb{Z}_1 \end{cases}$$

tous les couples dans
$\{(\bot_1, \bot_2), (\bot_1, 0_2), (\bot_1, 1_2), (\bot_1, \top_2), (0_1, \bot_2)(-_1, \bot_2), (+_1, \bot_2),$
$(\top_1, \bot_2), (0_1, 1_2)\}$ ont la même concrétisation $\emptyset$.

Le problème

Si nous cherchons une approximation correcte de $\mathrm{succ} : \mathcal{P}(\mathbb{Z}) \to \mathcal{P}(\mathbb{Z})$ définie par $\mathrm{succ}(S) = \{ x+1 \mid x \in S \}$, il est tentant de se baser sur les approximations optimales de chaque abstraction.

$$
\begin{array}{rcl}
\mathrm{succ}_1^\sharp(\bot_1) & = & \bot_1 \\
\mathrm{succ}_1^\sharp(0_1) & = & +_1 \\
\mathrm{succ}_1^\sharp(-_1) & = & \top_1 \\
\mathrm{succ}_1^\sharp(+_1) & = & +_1 \\
\mathrm{succ}_1^\sharp(\top_1) & = & \top_1
\end{array}
\qquad
\begin{array}{rcl}
\mathrm{succ}_2^\sharp(\bot_2) & = & \bot_2 \\
\mathrm{succ}_2^\sharp(0_2) & = & 1_2 \\
\mathrm{succ}_2^\sharp(1_2) & = & 0_2 \\
\mathrm{succ}_2^\sharp(\top_2) & = & \top_2
\end{array}
$$

Pour définir $\mathrm{succ}^\sharp : \mathcal{A}_1^\sharp \times \mathcal{A}_2^\sharp \to \mathcal{A}_1^\sharp \times \mathcal{A}_2^\sharp$ par

$$\mathrm{succ}^\sharp(a_1^\sharp, a_2^\sharp) = (\mathrm{succ}_1^\sharp(a_1^\sharp), \mathrm{succ}_2^\sharp(a_2^\sharp))$$

$\mathrm{succ}^\sharp$ est une une approximation correcte mais elle n'est pas du tout optimale, pourquoi ?

Produit réduit

On calcule la réduction optimale $\rho = \alpha \circ \gamma$

ρ	$\perp_1$	0_1	$-_1$	$+_1$	$\top_1$
$\perp_2$	$(\perp_1, \perp_2)$	$(\perp_1, \perp_2)$	$(\perp_1, \perp_2)$	$(\perp_1, \perp_2)$	$(\perp_1, \perp_2)$
0_2	$(\perp_1, \perp_2)$	$(0_1, 0_2)$	$(-_1, 0_2)$	$(+_1, 0_2)$	$(\top_1, 0_2)$
1_2	$(\perp_1, \perp_2)$	$(\perp_1, \perp_2)$	$(-_1, 1_2)$	$(+_1, 1_2)$	$(\top_1, 1_2)$
$\top_2$	$(\perp_1, \perp_2)$	$(0_1, 0_2)$	$(-_1, \top_2)$	$(+_1, \top_2)$	$(\top_1, \top_2)$

et on réalise les calculs sur $\rho(\mathcal{A}_1^\sharp \times \mathcal{A}_2^\sharp)$: produit réduit

Produit réduit

$\rho \circ \mathsf{succ}^\sharp \circ \rho$ est alors une approximation correcte de succ plus précise que $\mathsf{succ}^\sharp$. On a par exemple cette fois

$$\rho \circ \mathsf{succ}^\sharp \circ \rho(0_1, 1_2) = (\bot_1, \bot_2) = \alpha \circ \mathsf{succ} \circ \gamma(0_1, 1_2)$$

L'opérateur de réduction permet ainsi de faire « communiquer » les abstractions par signes et par congruence. La situation n'est toute fois toujours pas optimale puisque
$(-_1, 0_2) = \alpha \circ \mathsf{succ} \circ \gamma(-_1, 1_2) \sqsubset \rho \circ \mathsf{succ}^\sharp \circ \rho(-_1, 1_2) = (\top_1, 0_2)$ Ainsi un opérateur de réduction permet d'améliorer facilement des approximations correctes d'un opérateur f, mais pour obtenir une approximation optimale il faut revenir à la définition de $\alpha \circ f \circ \gamma$.

Plan

Polyhedral analysis

A more complex example.

The analysis accepts to replace some constants by parameters.

```
x = 0; y = A;
```

$$\{A \leqslant y \leqslant 2x + A \wedge x \leqslant N\}$$

```
while (x<N) {
  if (?) {
```

$$\{A \leqslant y \leqslant 2x + A \ \wedge \ x \leqslant N - 1\}$$

```
    y = y+2;
```

$$\{A + 2 \leqslant y \leqslant 2x + A + 2 \ \wedge \ x \leqslant N - 1\}$$

```
  };
```

$$\{A \leqslant y \leqslant 2x + A + 2 \ \wedge \ 0 \leqslant x \leqslant N - 1\}$$

```
  x = x+1;
```

$$\{A \leqslant y \leqslant 2x + A \ \wedge \ 1 \leqslant x \leqslant N\}$$

```
}
```

$$\{A \leqslant y \leqslant 2x + A \ \wedge \ N = x\}$$

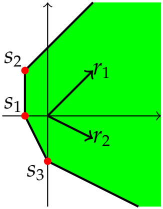
The four polyhedra operations

- $\uplus \in \mathbb{P}_n \times \mathbb{P}_n \rightarrow \mathbb{P}_n$: convex union
  - over-approximates the concrete union in junction points
- $\cap \in \mathbb{P}_n \times \mathbb{P}_n \rightarrow \mathbb{P}_n$: intersection
  - over-approximates the concrete intersection after a conditional intruction
- $[\![\mathtt{x} := e]\!] \in \mathbb{P}_n \rightarrow \mathbb{P}_n$: affine transformation
  - over-approximates the affectation of a variable by a linear expression
- $\nabla \in \mathbb{P}_n \times \mathbb{P}_n \rightarrow \mathbb{P}_n$: widening
  - ensures (and accelerate) convergence of (post-)fixpoint iteration
  - includes heuristics to infer loop invariants

```
x = 0; y = 0;
```
$$P_0 = [\![\mathtt{y} := \mathtt{0}]\!][\![\mathtt{x} := \mathtt{0}]\!](\mathbb{Q}^2) \nabla P_4$$
```
while (x<6) {
  if (?) {
```
$$P_1 = P_0 \cap \{x < 6\}$$
```
    y = y+2;
```
$$P_2 = [\![\mathtt{y} := \mathtt{y} + \mathtt{2}]\!](P_1)$$
```
  };
```
$$P_3 = P_1 \uplus P_2$$
```
  x = x+1;
```
$$P_4 = [\![\mathtt{x} := \mathtt{x} + \mathtt{1}]\!](P_3)$$
```
}
```
$$P_5 = P_0 \cap \{x \geqslant 6\}$$

Library for manipulating polyhedra

- Parma Polyhedra Library[1] (PPL), NewPolka : complex C/C++ libraries
- They rely on the Double Description Method
  - polyhedra are managed using two representations in parallel



- by set of inequalities

$$P = \left\{ (x,y) \in \mathbb{Q}^2 \;\middle|\; \begin{array}{l} x \geqslant -1 \\ x - y \geqslant -3 \\ 2x + y \geqslant -2 \\ x + 2y \geqslant -4 \end{array} \right\}$$

- by set of generators

$$P = \left\{ \lambda_1 s_1 + \lambda_2 s_2 + \lambda_3 s_3 + \mu_1 r_1 + \mu_2 r_2 \in \mathbb{Q}^2 \;\middle|\; \begin{array}{l} \lambda_1, \lambda_2, \lambda_3, \mu_1, \mu_2 \in \mathbb{R}^2 \\ \lambda_1 + \lambda_2 + \lambda_3 = 1 \end{array} \right\}$$

  - operations efficiency strongly depends on the chosen representations, so they keep both

[1] Previous tutorial on polyhedra partially comes from `http://www.cs.unipr.it/ppl/`

Plan

Références (1)

Quelques articles

- introduction formelle

  P. Cousot and R. Cousot. Basic Concepts of Abstract Interpretation.
  `http://www.di.ens.fr/~cousot/COUSOTpapers/WCC04.shtml`

- technique mais très complet (la partie programmation logique est facultative) :

  P. Cousot and R. Cousot. Abstract Interpretation and Application to Logic Programs.
  `http://www.di.ens.fr/~cousot/COUSOTpapers/JLP92.shtml`

- une belle application pour vérifier les commandes de vol des airbus

  P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, D. Monniaux, and X. Rival. The ASTRÉE Analyser.
  `http://www.di.ens.fr/~cousot/COUSOTpapers/ESOP05.shtml`

Références (2)

Sur le web :

- présentation informelle de l'IA avec des jolies dessins

  http://www.astree.ens.fr/IntroAbsInt.html

- un rapide résumé des travaux autour de l'IA

  http://www.di.ens.fr/˜cousot/AI/

- notes de cours très complètes

  http://web.mit.edu/afs/athena.mit.edu/course/16/16.399/www/