

Mémoire de stage de DEA réalisé sous la direction de Patrick Cousot

Laboratoire d'Informatique de l'ENS

---

**Représentation d'ensembles de contraintes de somme  
ou de différence de deux variables et application  
à l'analyse automatique de programmes**

---

ANTOINE MINÉ

18 septembre 2000

# Présentation du sujet

Présentation d'une nouvelle méthode d'analyse statique automatique de programmes qui découvre des invariants numériques nommés **contrainte de somme ou de différence de deux variables** et de la forme :

$$\pm x \pm y \leq c$$

## Exemple

Programme  $p$  :

```
 $x = 12;$   
 $y = 0;$   
while  $z > 0$  do  
   $x = x + 1;$   
   $y = y - 1;$   
   $z = z - 1$   
done
```

à la fin de  $p$ , on a :

$$\begin{array}{rcl} +x + y & \leq & 12 \\ -x - y & \leq & -12 \end{array}$$

# Travaux antérieurs : interprétation abstraite

L'interprétation abstraite présente un cadre formel pour définir des analyses **automatiques**, **approchées** et **sûres** ([CC77]).

Étant donné un ensemble  $\mathcal{V}$  de variables et un ensemble numérique  $\mathbb{I}$ , une analyse numérique est définie par un sous-ensemble  $\mathcal{S}$  de  $\mathcal{P}(\mathcal{V} \mapsto \mathbb{I})$  qui détermine la forme des invariants découverts.

## Exemples

- Intervalles :

$\mathcal{S}$  est l'ensemble des pavés;

les invariants sont de la forme  $x_i \in [c_i, d_i]$ .

- Inéquations linéaires :

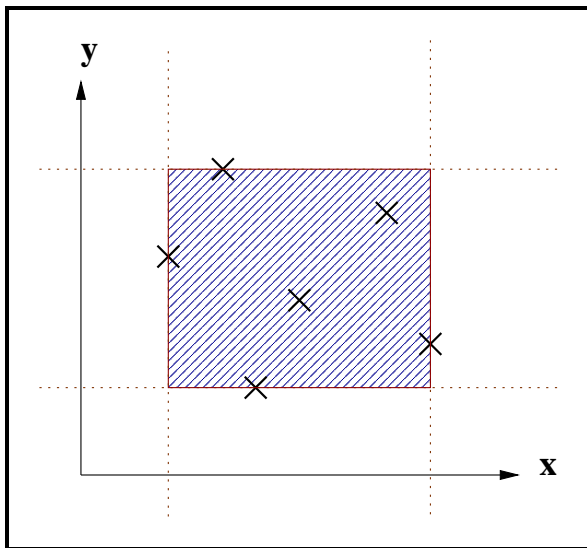
$\mathcal{S}$  est l'ensemble des polyèdres;

les invariants sont de la forme  $\alpha_1 x_1 + \dots + \alpha_n x_n \leq c$ .

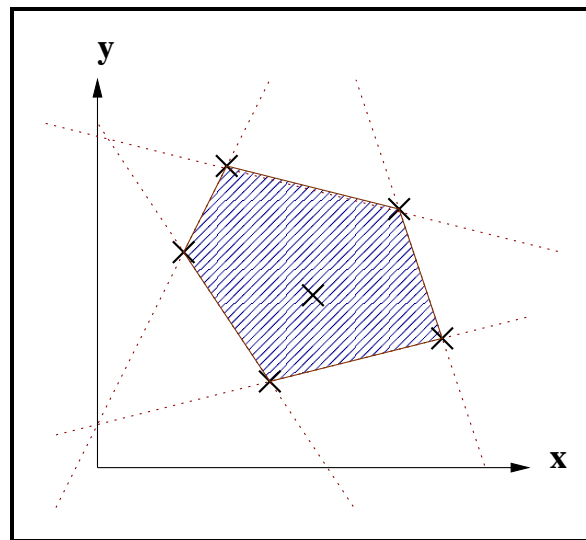
# Exemples d'analyses numériques

Les intervalles sont efficaces (coût linéaire en le nombre de variables) tandis que les inéquations linéaires sont coûteuses (coût exponentiel au pire).

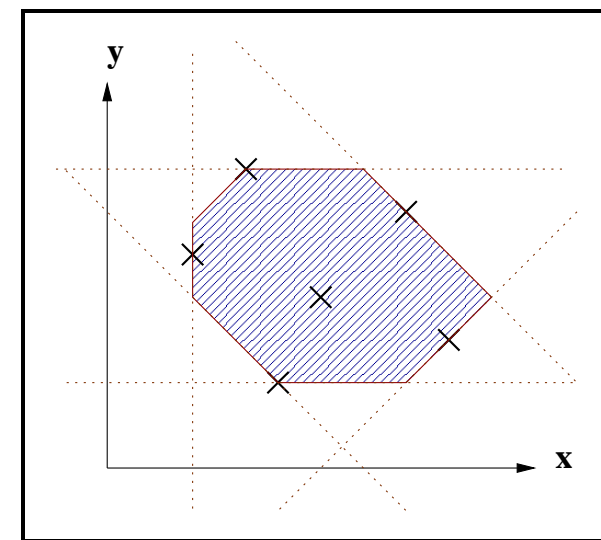
Les inéquations linéaires sont beaucoup plus précises car **relationnelles**.



Approximation par un pavé.



Approximation par un polyèdre.



Approximation par des contraintes de somme ou différences de deux variables.

# Travaux antérieurs :

## Difference Bound Matrices (DBMs)

Les contraintes de différences de deux variables ( $x - y \leq c$ ) sont utilisés depuis longtemps pour le model-checking d'automates temporisés.

Les **Difference Bound Matrices** (DBMs) ont été introduites pour représenter et manipuler les ensembles de contraintes de différences de deux variables avec un **coût polynomial** ([LLPY]).

### Inconvénients

- Les DBMs ne permettent pas de représenter à la fois les **sommes** et les différences de deux variables.
- Les opérateurs sur les DBMs introduits pour le **model-checking** ne sont pas adaptés à l'analyse statique.

# Plan de l'exposé

Le travail de DEA a consisté en plusieurs parties :

- Définition d'**opérateurs** sur les DBMs adaptés à l'interprétation abstraite (affectation, test, élargissement, ... ).
- Adaptation des DBMs pour représenter les contraintes de **sommes** de deux variables en plus des contraintes de différence de deux variables.
- Définition d'une structure de **treillis complet** sur les DBMs et d'une **correspondance de Galois**.
- Construction d'un **analyseur abstrait** d'un langage simple et **preuve de sûreté** par interprétation abstraite.

Implantation et résultats.

# Définition des Difference Bound Matrices

Soient  $\mathcal{N} = \{x_1, \dots, x_n\}$  un ensemble de variables et  $\bar{\mathbb{I}}$  un ensemble numérique  $\mathbb{I}$  auquel on a ajouté  $+\infty$ .

Un ensemble  $C$  de contraintes de la forme  $x - y \leq c$  est représenté par une matrice  $m$  carrée de taille  $n$  à coefficients dans  $\bar{\mathbb{I}}$  nommée **Difference Bound Matrix** (DBM) :

$$m_{ij} = \begin{cases} c & \text{si } (x_j - x_i \leq c) \in C \\ +\infty & \text{sinon} \end{cases}$$

C'est la matrice d'adjacence d'un graphe pondéré dirigé,  $\mathcal{G} = (\mathcal{N}, \mathcal{A}, w)$ , nommé **graphe de potentiel** et définit par :

$$\mathcal{A} \subseteq \mathcal{N} \times \mathcal{N}, \quad w \in \mathcal{A} \mapsto \mathbb{I},$$

$$\begin{cases} (x_i, x_j) \in \mathcal{A} \text{ et } w(x_i, x_j) = m_{ij} & \text{si } m_{ij} \neq +\infty \\ (x_i, x_j) \notin \mathcal{A} & \text{si } m_{ij} = +\infty \end{cases}$$

# Définition des Difference Bound Matrices (suite)

Le  $\mathcal{N}$ -**domaine** d'une DBM  $m$ , noté  $\mathcal{D}(m)$ , est défini par :

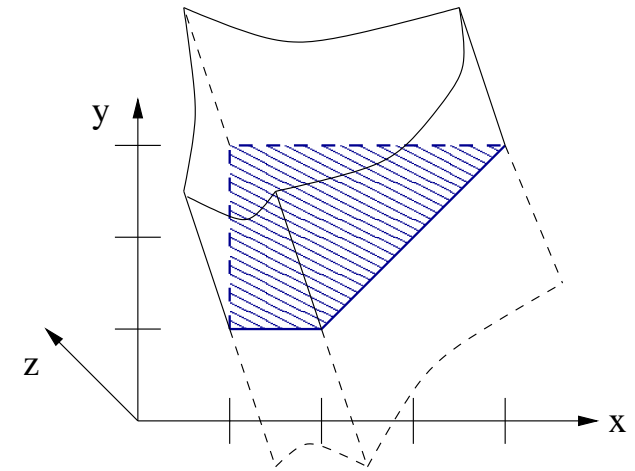
$$\mathcal{D}(m) = \{ (s_1, \dots, s_n) \in \mathbb{I}^n \mid \forall i, j, s_j - s_i \leq m_{ij} \}.$$

## Exemple

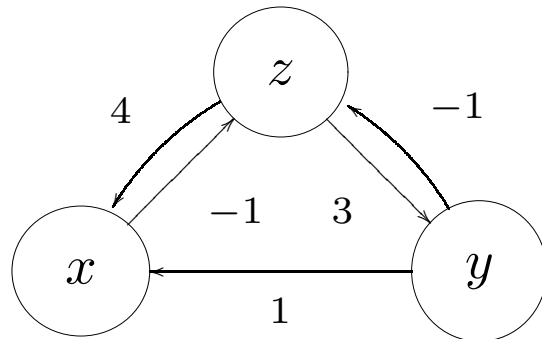
Système de contraintes

$$\left\{ \begin{array}{l} x - z \leq 4 \\ z - x \leq -1 \\ y - z \leq 3 \\ z - y \leq -1 \\ x - y \leq 1 \end{array} \right.$$

$\mathcal{N}$ -domaine



Graphe de potentiel



Difference Bound Matrix

	$x$	$y$	$z$
$x$	$+\infty$	$+\infty$	$-1$
$y$	$1$	$+\infty$	$-1$
$z$	$4$	$3$	$+\infty$

# Relation d'ordre partiel sur les DBMs

## Définition

ordre partiel  $\trianglelefteq$  défini par :

$$m \trianglelefteq n \iff \forall i, j, m_{ij} \leq n_{ij}.$$

## Propriété

$$m \trianglelefteq n \implies \mathcal{D}(m) \subseteq \mathcal{D}(n).$$

La réciproque est fausse!

En particulier, **on n'a pas** :  $\mathcal{D}(m) = \mathcal{D}(n) \implies m = n.$

## Exemple

		$x$	$y$	$z$			$x$	$y$	$z$
(A)	$x$	$+\infty$	$+\infty$	$-1$	(B)	$x$	$\mathbf{0}$	$+\infty$	$-1$
	$y$	$1$	$0$	$-1$		$y$	$1$	$0$	$-1$
	$z$	$\mathbf{4}$	$3$	$0$		$z$	$\mathbf{5}$	$3$	$0$

# Opérations sur les DBMs

Définitions d'opérations sur les Difference Bound Matrices et interprétation en terme de  $\mathcal{N}$ -domaines.

- Test du vide.
- Clôture (mise en forme normale).
- Test d'inclusion et d'égalité.
- Intersection.
- Plus petit majorant (approximation de l'union).
- Élargissement (approximation de limite infinie).
- Oubli d'une variable.
- Garde.
- Affectation.

# Test du vide

## Théorème

$\mathcal{D}(m) = \emptyset$  si et seulement si il existe un **cycle simple de poids total strictement négatif** dans le graphe de potentiel associé à  $m$ .

## Algorithme

Algorithme de **Bellman-Ford** sur le graphe de potentiel.

Décrit dans la section 25.3 de [CLR90].

Coût en  $\mathcal{O}(n^3)$  au pire.

# Forme normale des DBMs

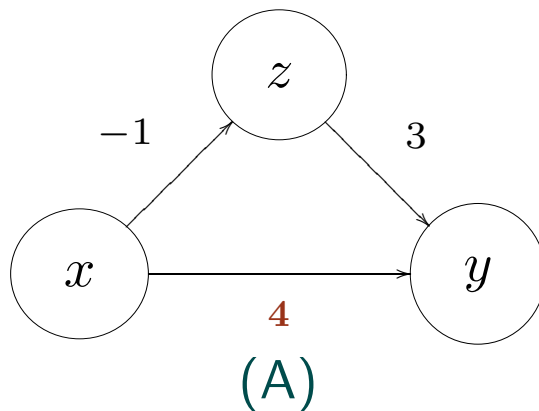
Des DBMs différentes peuvent avoir le même  $\mathcal{N}$ -domaine.

La forme normale, notée  $m^*$ , d'une DBM  $m$  de  $\mathcal{N}$ -domaine non vide est sa **clôture par plus-court chemin définie** par :

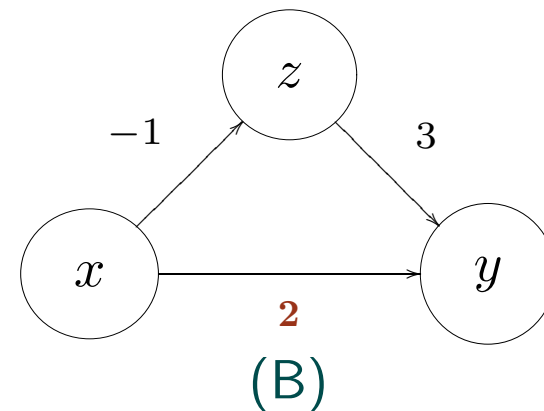
$$\begin{cases} m_{ii}^* = 0 \\ m_{ij}^* = \min_{\substack{1 \leq N \\ \langle i=i_1, i_2, \dots, i_N=j \rangle}} \sum_{k=1}^{N-1} m_{i_k i_{k+1}} \quad \text{si } i \neq j \end{cases}$$

## Exemple

$$\begin{cases} y - z \leq 3 \\ z - x \leq -1 \end{cases}$$



$\Rightarrow$



$$y - x \leq 2$$

# Forme normale des DBMs (suite)

## Propriétés

- $m$  et  $m^*$  ont même  $\mathcal{N}$ -domaine.
- $m^*$  est **minimale** pour l'ordre  $\triangleright$ .
- $\mathcal{D}(m)$  **sature**  $m^*$  :  $\forall i, j, \exists (s_1, \dots, s_n) \in \mathcal{D}(m), s_j - s_i = m_{ij}^*$ .

## Calcul

Algorithme de **Floyd-Warshall** :

$$(A) \begin{cases} m_0 = m \\ m_k = m_{k-1}^k \\ m^* = m_n \end{cases} \quad (B) \begin{cases} n_{ii}^k = 0 \\ n_{ij}^k = \min(n_{ij}, n_{ik} + n_{kj}) \quad \text{si } i \neq j \end{cases}$$

L'algorithme de Floyd-Warshall est décrit dans la section 26.2 de [CLR90] et a un coût en  $\mathcal{O}(n^3)$ .

# Test d'inclusion et d'égalité

On considère le cas  $\mathcal{D}(m) \neq \emptyset$  et  $\mathcal{D}(n) \neq \emptyset$ .

## Théorème

- $\mathcal{D}(m) \subseteq \mathcal{D}(n) \iff m^* \triangleleft n.$
- $\mathcal{D}(m) = \mathcal{D}(n) \iff m^* = n^*.$

Coût en  $\mathcal{O}(n^2)$ , sans compter le coût des clôtures.

# Intersection et plus petit majorant

L'ensemble des  $\mathcal{N}$ -domaines est stable par intersection mais pas par union.

## Définitions

$$(m \wedge n)_{ij} = \min(m_{ij}, n_{ij})$$

$$(m \vee n)_{ij} = \max(m_{ij}, n_{ij})$$

## Propriétés

- $\mathcal{D}(m \wedge n) = \mathcal{D}(m) \cap \mathcal{D}(n)$  (**intersection**).
- $\mathcal{D}(m \vee n) \supseteq \mathcal{D}(m) \cup \mathcal{D}(n)$  (majorant).  
Si  $\mathcal{D}(o) \supseteq \mathcal{D}(m) \cup \mathcal{D}(n)$  alors  $\mathcal{D}(o) \supseteq \mathcal{D}(m^* \vee n^*)$  (**plus petit majorant**).

Coût en  $\mathcal{O}(n^2)$ , sans compter le coût des clôtures.

# Élargissement

Introduit dans [CC78] pour calculer en **temps fini** une approximation d'un plus-petit point fixe solution d'une équation de récurrence.

$$\begin{cases} b_0 = a_0 \\ b_{i+1} = b_i \nabla a_{i+1} \end{cases}$$

## Définition

$$(m \nabla n)_{ij} = \begin{cases} m_{ij} & \text{si } n_{ij} \leq m_{ij} \\ +\infty & \text{sinon.} \end{cases}$$

## Propriétés

- $\mathcal{D}(m \nabla n) \supseteq \mathcal{D}(m) \cup \mathcal{D}(n)$ .
- $\forall (a_i)_{i \in \mathbb{N}}$ , la chaîne  $(b_i)_{i \in \mathbb{N}}$  est **croissante** et **ultimement stationnaire**.

# Oubli d'une variable

$$(m \setminus x_k)_{ij} = \begin{cases} m_{ij} & \text{si } i \neq k \text{ et } j \neq k \\ 0 & \text{si } i = j = k \\ +\infty & \text{sinon.} \end{cases}$$

## Propriétés

- Si  $m$  est close, alors :

$$\mathcal{D}(m \setminus x_k) = \{ (x_1, \dots, x_n) \in \mathbb{I}^n \mid \exists x \in \mathbb{I}, (x_1, \dots, x_{k-1}, x, x_{k+1}, \dots, x_n) \in \mathcal{D}(m) \}$$

- Si  $m$  est close, alors  $m \setminus x_k$  est close.

# Garde et affectation

Image  $m_{(g)}$  d'une DBM  $m$  par une **garde**  $g$  (égalité ou inégalité) :

$$\mathcal{D}(m_{(g)}) \supseteq \{ s \in \mathcal{D}(m) \mid s \text{ satisfait } g \}.$$

Image  $m_{(x_i \leftarrow e)}$  d'une DBM  $m$  par une **affectation**  $x_i \leftarrow e$  :

$$\mathcal{D}(m_{(x_i \leftarrow e)}) \supseteq \{ (s_1, \dots, s, \dots, s_n) \mid \exists s_i, (s_1, \dots, s_n) \in \mathcal{D}(m), s = e(s_1, \dots, s_n) \}.$$

## Exemples

- $(m_{(x_{j_0} - x_{i_0} \leq c)})_{ij} = \begin{cases} \min(m_{ij}, c) & \text{si } i = i_0 \text{ et } j = j_0 \\ m_{ij} & \text{sinon} \end{cases} \quad \text{si } c \in \mathbb{I}.$
- $m_{(x_{i_0} \leftarrow x_{j_0} + c)} = (((m^*) \setminus x_{i_0})_{(x_{i_0} - x_{j_0} \leq c)})_{(x_{j_0} - x_{i_0} \leq -c)} \quad \text{si } i_0 \neq j_0 \text{ et } c \in \mathbb{I}.$
- $m_{(g)} = m$  et  $m_{(x_{i_0} \leftarrow e)} = (m^*) \setminus x_{i_0}$  dans le cas général.

# Représentation des contraintes de somme

Soit un ensemble fini de variables  $\mathcal{V} = \{v_1, \dots, v_n\}$ .

On considère les DBMs définies sur  $\mathcal{N} = \{v_1^+, v_1^-, \dots, v_n^+, v_n^-\}$ .

La contrainte	est représentée par
$v - w \leq c \quad (v \neq w)$	$v^+ - w^+ \leq c \quad \text{et} \quad w^- - v^- \leq c$
$v + w \leq c \quad (v \neq w)$	$v^+ - w^- \leq c \quad \text{et} \quad w^+ - v^- \leq c$
$-v - w \leq c \quad (v \neq w)$	$w^- - v^+ \leq c \quad \text{et} \quad v^- - w^+ \leq c$
$v \leq c$	$v^+ - v^- \leq 2c$
$v \geq c$	$v^- - v^+ \leq -2c$

Le  $\mathcal{V}$ -**domaine** de  $m$  est  $\Pi(\mathcal{D}(m))$  avec :

$$\Pi(D) = \{ (t_1, \dots, t_n) \mid (t_1, -t_1, \dots, t_n, -t_n) \in D \} \subseteq (\mathcal{V} \rightarrow \mathbb{I}).$$

# Cohérence

## Définitions

- $\begin{cases} \text{variable de } \mathcal{N} & v_1^+ & v_1^- & \cdots & v_n^+ & v_n^- \\ \text{indice dans } m & 1 & 2 & \cdots & 2n-1 & 2n \end{cases}$
- $\bar{i}$  est défini par :

$$\bar{i} = \begin{cases} i + 1 & \text{si } i \text{ est impair} \\ i - 1 & \text{si } i \text{ est pair} \end{cases}$$

- $m$  est dite **cohérente** si :

$$\forall i, j \quad m_{ij} = m_{j\bar{i}}$$

$$(\text{exemple : } v_i^+ - v_j^+ \leq c \iff v_j^- - v_i^- \leq c)$$

Désormais, on ne s'intéresse qu'aux DBM cohérentes sur  $\mathcal{N}$ .

# Forme normale forte

Étant donnée une DBM, il peut exister des DBMs de même  $\mathcal{V}$ -domaine mais de  $\mathcal{N}$ -domaine différent.

La clôture  $m^*$  d'une DBM n'est donc pas une forme normale acceptable pour représenter un  $\mathcal{V}$ -domaine.

## Exemple



## Définition

$m$  est en **forme normale forte** si et seulement si :

$$\left\{ \begin{array}{ll} \forall i, j, & m_{ij} = m_{\bar{j}\bar{i}} \quad (\text{coherence}) \\ \forall i, j, k, & m_{ij} \leq m_{ik} + m_{kj} \quad (\text{cl\^oture}) \\ \forall i, j & m_{ij} \leq (m_{i\bar{i}} + m_{\bar{j}j})/2 \quad (\text{forte coherence}) \end{array} \right.$$

# Clôture forte

## Algorithme

La **clôture forte**  $m^*$  de  $m$  est obtenue par l'algorithme de **Floyd-Warshall modifié** :

$$(B) \quad \left\{ \bar{n}_{ij} = \min( n_{ij}, (n_{i\bar{i}} + n_{\bar{j}j})/2 ) \right.$$

$$(A) \quad \left\{ \begin{array}{l} m_0 = m \\ m_k = \overline{m_{k-1}^{2k-1}}, \quad 1 \leq k \leq n \\ m^* = m_n \end{array} \right.$$

$$(C) \quad \left\{ \begin{array}{l} n_{ii}^k = 0 \\ n_{ij}^k = \min( \begin{array}{l} n_{ij}, \\ n_{ik} + n_{kj}, \\ n_{i\bar{k}} + n_{\bar{k}j}, \\ n_{ik} + n_{k\bar{k}} + n_{\bar{k}j}, \\ n_{i\bar{k}} + n_{\bar{k}k} + n_{kj} \end{array} ) \quad \text{si } i \neq j \end{array} \right.$$

## Propriétés

- $m$  et  $m^*$  ont même  $\mathcal{V}$ -domaine.
- $m^*$  est **close**, **cohérente**, **fortement cohérente** et **minimale** pour  $\triangleleft$ .
- $\Pi(\mathcal{D}(m^*))$  **sature**  $m^*$ .
- Coût en  $\mathcal{O}(n^3)$ .

# Adaptation des opérations

- Test du vide :  $\Pi(\mathcal{D}(m)) = \emptyset \iff m$  a un cycle strictement négatif.
- Test d'égalité :  $\Pi(\mathcal{D}(m)) = \Pi(\mathcal{D}(n)) \iff m^* = n^*$ .
- Test d'inclusion :  $\Pi(\mathcal{D}(m)) \subseteq \Pi(\mathcal{D}(n)) \iff m^* \trianglelefteq n$ .
- Intersection :  $m \wedge n$ .
- Plus petit majorant :  $(m^*) \vee (n^*)$ .
- Élargissement :  $m \nabla n$ .
- Oubli d'une variable :  $m_{\parallel v_k} = m_{\setminus v_k^+ \setminus v_k^-}$ .
- Garde : représentation exacte de  $x + y \leq c$  et  $x \leq c$  en plus de  $x - y \leq c$ .
- Affectation : représentation exacte de  $x \leftarrow c$  et  $x \leftarrow c - y$  en plus de  $x \leftarrow y + c$ .

# Treillis des DBMs fortement closes

## Définition

- Ensemble  $\mathcal{M}_{\mathcal{N}}^*(\mathbb{I})$  des DBMs **fortement closes** sur  $\mathcal{N} = \{v_1^+, \dots, v_n^-\}$ .
- On y ajoute  $\perp^*$  qui représente le  $\mathcal{V}$ -domaine vide.
- $\top^*$  est définie par :  $(\top^*)_{ii} = 0$ ,  $(\top^*)_{ij} = +\infty$  si  $i \neq j$ .
- $\preceq$ ,  $\vee$  et  $\wedge$  sont étendues à  $\perp^*$  pour donner  $\sqsubseteq^*$ ,  $\sqcup^*$  et  $\sqcap^*$ .  
 $\sqsubseteq^*$  est un ordre partiel.  
 $\sqcup^*$  et  $\sqcap^*$  sont les plus petit majorant et plus grand minorant vis à vis de  $\sqsubseteq^*$ .
- Tout ensemble  $X$  a un plus petit majorant  $\sqcup^* X$  et plus grand minorant  $\sqcap^* X$ .

## Propriété fondamentale

$(\mathcal{M}_{\mathcal{N}}^*(\mathbb{I}), \sqsubseteq^*, \perp^*, \top^*, \sqcup^*, \sqcap^*)$  est un treillis complet.

# Correspondance de Galois

## Définition

$$(\mathcal{P}(\mathcal{V} \mapsto \mathbb{I}), \subseteq, \emptyset, \mathcal{V} \mapsto \mathbb{I}, \cup, \cap) \xrightleftharpoons[\alpha]{\gamma} (\mathcal{M}_{\mathcal{N}}^*(\mathbb{I}), \sqsubseteq^*, \perp^*, \top^*, \sqcup^*, \sqcap^*).$$

$$\begin{cases} \gamma(\perp^*) = \emptyset \\ \gamma(m) = \Pi(\mathcal{D}(m)) \quad \text{si } m \neq \perp^* \end{cases}$$

$$\alpha(D) = \sqcap^* \{ m \in \mathcal{M}_{\mathcal{N}}^*(\mathbb{I}) \mid D \subseteq \gamma(m) \}.$$

## Propriétés

- $(\alpha, \gamma)$  est une **correspondance de Galois** :  $\alpha(D) \sqsubseteq^* m \iff D \subseteq \gamma(m)$ .
- $\gamma$  est **injective** et  $\alpha \circ \gamma = id$ .
- $\gamma(\sqcap^* X) = \bigcap_{x \in X} \gamma(x)$  ( **morphisme complet pour  $\cap$**  ).
- $\gamma(\sqcup^* X) \supseteq \bigcup_{x \in X} \gamma(x)$  (sous-morphisme complet pour  $\cup$ ).

# Syntaxe du langage $\mathcal{I}$

Langage impératif minimal, mais Turing-complet :

	instructions		tests
$\mathcal{I} ::=$	$()$	$\mathcal{T} ::=$	$\mathcal{T}$ <b>and</b> $\mathcal{T}$
	$\mathcal{I}; \mathcal{I}$		$\mathcal{T}$ <b>or</b> $\mathcal{T}$
	$x = \alpha x + \beta y + c$		$\alpha x + \beta y + c \leq 0$
	<b>if</b> ( $\mathcal{T}$ ) <b>then</b> $\mathcal{I}$ <b>else</b> $\mathcal{I}$		
	<b>while</b> ( $\mathcal{T}$ ) <b>do</b> $\mathcal{I}$		

où  $x, y \in \mathcal{V}$  et  $\alpha, \beta, c \in \mathbb{Z}$ .

Un **état** du programme est une fonction de  $\mathcal{V} \mapsto \mathbb{Z}$ .

Étant donné un **état initial**, un programme  $p$  peut :

- terminer après un nombre fini d'étapes dans un certain **état final**;
- boucler indéfiniment.

# Sémantique concrète

## Définition

$\llbracket p \rrbracket \in \mathcal{P}(\mathcal{V} \mapsto \mathbb{Z}) \mapsto^m \mathcal{P}(\mathcal{V} \mapsto \mathbb{Z})$  associe à un ensemble d'états initiaux l'ensemble des états finaux atteints par  $p$  après un **temps fini** :

- $\llbracket () \rrbracket = \lambda D. D$
- $\llbracket a; b \rrbracket = \llbracket b \rrbracket \circ \llbracket a \rrbracket$
- $\llbracket a \textbf{ and } b \rrbracket = \lambda D. \llbracket a \rrbracket(D) \cap \llbracket b \rrbracket(D)$
- $\llbracket a \textbf{ or } b \rrbracket = \lambda D. \llbracket a \rrbracket(D) \cup \llbracket b \rrbracket(D)$
- $\llbracket \textbf{if } (t) \textbf{ then } a \textbf{ else } b \rrbracket = \lambda D. \llbracket a \rrbracket(\llbracket t \rrbracket(D)) \cup \llbracket b \rrbracket(D \setminus \llbracket t \rrbracket(D))$
- $\llbracket \textbf{while } (t) \textbf{ do } a \rrbracket = \lambda D. (\lambda E. E \setminus \llbracket t \rrbracket(E))(\text{lfp}(\lambda F. D \cup \llbracket a \rrbracket(\llbracket t \rrbracket(F))))$
- $\llbracket \alpha v_i + \beta v_j + c \leq 0 \rrbracket = \lambda D. \{ (t_1, \dots, t_n) \in D \mid \alpha t_i + \beta t_j + c \leq 0 \}$
- $\llbracket v_i = \alpha v_i + \beta v_j + c \rrbracket = \lambda D. \{ (t_1, \dots, t_{i-1}, t_i, t_{i+1}, \dots, t_n) \in \mathcal{V} \mapsto \mathbb{Z} \mid \exists t, (t_1, \dots, t_{i-1}, t, t_{i+1}, \dots, t_n) \in D, t_i = \alpha t + \beta t_j + c \}$

Cette sémantique est **non-calculable**!

# Sémantique abstraite

$\llbracket p \rrbracket^\# \in \mathcal{M}_{\mathcal{N}}^*(\mathbb{Z}) \mapsto^m \mathcal{M}_{\mathcal{N}}^*(\mathbb{Z}) :$

- $\llbracket () \rrbracket^\# = \lambda m. m$
- $\llbracket a; b \rrbracket^\# = \llbracket b \rrbracket^\# \circ \llbracket a \rrbracket^\#$
- $\llbracket a \text{ and } b \rrbracket^\# = \lambda m. \llbracket a \rrbracket^\#(m) \sqcap^* \llbracket b \rrbracket^\#(m)$
- $\llbracket a \text{ or } b \rrbracket^\# = \lambda m. \llbracket a \rrbracket^\#(m) \sqcup^* \llbracket b \rrbracket^\#(m)$
- $\llbracket \text{if } (t) \text{ then } a \text{ else } b \rrbracket^\# = \lambda m. \llbracket a \rrbracket^\#(\llbracket t \rrbracket^\#(m)) \sqcup^* \llbracket b \rrbracket^\#(\llbracket \neg t \rrbracket^\#(m))$
- $\llbracket \text{while } (t) \text{ do } a \rrbracket^\# = \lambda m. \llbracket \neg t \rrbracket^\#(\text{lfp}_m(\lambda n. n \nabla \llbracket a \rrbracket^\#(\llbracket t \rrbracket^\#(n^*))))^*$
- $\llbracket \alpha v_i + \beta v_j + c \leq 0 \rrbracket^\# = \lambda m. (\mathbf{m}_{(\alpha \mathbf{v}_i + \beta \mathbf{v}_j + \mathbf{c} \leq \mathbf{0})})^*$
- $\llbracket v_i = \alpha v_i + \beta v_j + c \rrbracket^\# = \lambda m. (\mathbf{m}_{(\mathbf{v}_i \leftarrow \alpha \mathbf{v}_i + \beta \mathbf{v}_j + \mathbf{c})})^*$

avec les règles de réécriture pour  $\neg$  :

$$\begin{array}{ll} \neg(a \text{ and } b) & \rightarrow (\neg a) \text{ or } (\neg b) \\ \neg(a \text{ or } b) & \rightarrow (\neg a) \text{ and } (\neg b) \\ \neg(\alpha v_i + \beta v_j + c \leq 0) & \rightarrow (-\alpha)v_i + (-\beta)v_j + (1 - c) \leq 0 \end{array}$$

# Sémantique abstraite (suite)

## Propriétés

- La sémantique abstraite  $\llbracket \cdot \rrbracket^\#$  est **calculable**.
- La sémantique abstraite est **sûre** par rapport à la sémantique concrète :

$$\llbracket p \rrbracket(D) \subseteq \gamma(\llbracket p \rrbracket^\#(\alpha(D))).$$

Elle calcule un **sur-ensemble** des états finaux atteints après un temps fini par  $p$ .

## Précision

- Union non exacte.
- Opérateurs de garde et d'affectation non exacts dans le cas général.
- Approximation du point fixe (while) par élargissement.
- Au moins aussi précis que les intervalles.

# Exemples

Les opérations sur les DBMs ainsi que le calcul de la sémantique approchée d'un programme  $\mathcal{I}$  ont été implantés en OCAML.

```
 $x = 0;$   
while  $x < 40$  do  
  if  $z = 0$   
  then  $x = x + 1$   
  else  $x = x + 2$   
done
```

$\Rightarrow$

```
 $40 \leq x \leq 41$ 
```

```
 $x = 12;$   
 $y = 0;$   
while  $z > 0$  do  
   $x = x + 1;$   
   $y = y - 1;$   
   $z = z - 1$   
done
```

$\Rightarrow$

```
 $x \geq 12$   
 $y \leq 0$   
 $z \leq 0$   
 $x - y \leq 12$   
 $x + y = 12$   
 $x - z \geq 12$   
 $y + z \leq 0$ 
```

## Exemples (suite)

Extrait du **tri en tas** : vérification des bornes de tableau uniquement.

```
heap (A: array [1,size], i: integer)
```

```
l = 2i;
```

```
r = 2i + 1;
```

```
if l ≤ size and A.[l] > A.[i]
```

```
  then max = l
```

```
  else max = i
```

```
if r ≤ size and A.[r] > A.[max]
```

```
  then max = r
```

```
if max ≠ i
```

```
  then swap(A.[i],A.[max]); heap(A, max)
```

# Exemples (fin)

**Bakery Algorithm** pour la synchronisation de processus parallèles : vérification que les deux processus n'exécutent jamais simultanément leur section critique.

```
y1 = 0;
```

```
y2 = 0;
```

```
p1 ()
```

---

```
y1 = y2 + 1;
```

```
while y2 ≠ 0 and y1 > y2 do done;
```

```
- - - section critique - - -
```

```
y1 = 0;
```

```
p2 ()
```

---

```
y2 = y1 + 1;
```

```
while y1 ≠ 0 and y2 ≥ y1 do done;
```

```
- - - section critique - - -
```

```
y2 = 0;
```

# Améliorations possibles

- Utilisation d'une représentation plus compacte que les matrices (matrice creuse, [LLPY]).
- Amélioration des opérateurs de garde et d'affectation.
- Utilisation d'une sémantique abstraite plus précise utilisant les mêmes opérations sur les DBMs (sémantique en arrière).
- Utilisation du treillis des DBMs fortement closes dans d'autres types d'analyses (analyse d'alias paramétrée par un treillis numérique).

# Conclusion

Nous avons présenté une analyse statique de programmes basée sur un nouveau domaine abstrait numérique qui permet de représenter et de manipuler des invariants de la forme  $\pm x \pm y \leq c$  avec un coût en mémoire de  $\mathcal{O}(n^2)$  par état abstrait un coût en temps de  $\mathcal{O}(n^3)$  par opération abstraite.

Cette analyse est sûre et au moins aussi précise que l'analyse d'intervalles.

## Remerciements

Je remercie Jérôme Feret, Charles Hymans, David Monniaux et Laurent Mauborgne pour leur aide lors de mes recherches et lors de la rédaction de mon rapport.

# Bibliographie

- [Bel58] R. Bellman. **On a routing problem.**
- [CLR90] Thomas Cormen, Charles Leiserson, and Ronald Rivest. **Introduction to Algorithms.** The MIT Press, 1990.
- [CH78] P. Cousot and N. Halbwachs. **Automatic discovery of linear restraints among variables of a program.**
- [LLPY] Kim Larsen, Fredrik Larsson, Paul Pettersson, and Wang Yi. **Efficient verification of real-time systems : Compact data structure and state-space reduction.**
- [CC77] P. Cousot and R. Cousot. **Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints.**