Domaines relationnels

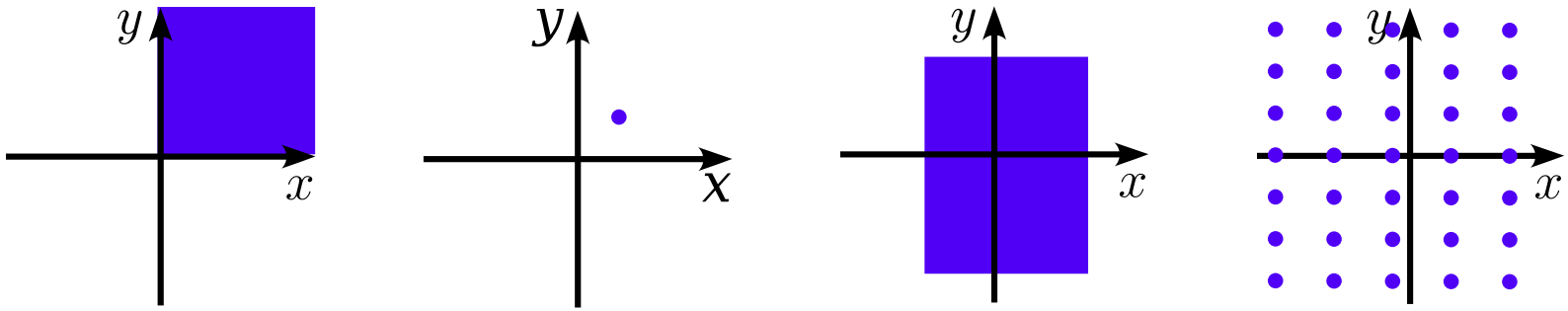
TAS : Typage et analyse statique
M2, Master STL INSTA, UPMC

Antoine Miné

Année 2016–2017

Cours 12
9 mars 2017

Rappel : domaines numériques non relationnels



signes constantes intervales parité

Principe :

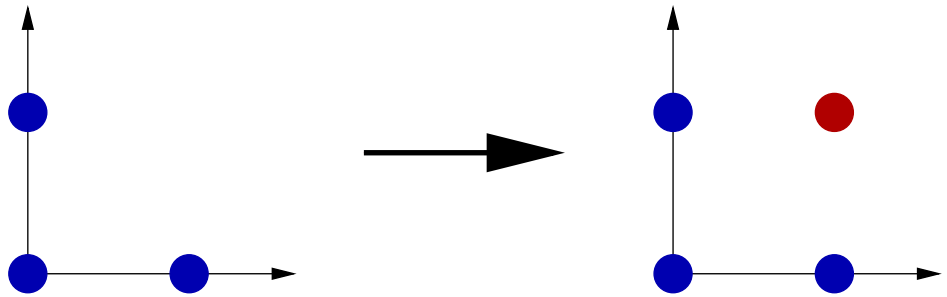
- définir une abstraction $\mathcal{D}^\sharp$ d'ensembles d'entiers $\mathcal{P}(\mathbb{Z})$
  et des abstractions $+^\sharp$, $-^\sharp$, $\times^\sharp$, $/^\sharp$, $\cup^\sharp$, $\subseteq^\sharp$, $\leq^\sharp$, ...
- représenter un ensemble d'environnements $X \in \mathcal{P}(\mathcal{E})$ en associant à chaque variable une valeur abstraite : $X^\sharp \in \mathcal{E}^\sharp \stackrel{\text{def}}{=} \mathbb{V} \to \mathcal{D}^\sharp$
  - $\mathcal{E}^\sharp$ a une structure de treillis
    extension de $\subseteq^\sharp$, $\cup^\sharp$, $\cap^\sharp$ point à point
  - $\mathsf{S}^\sharp[\![X \leftarrow e]\!]$ et $\mathsf{C}^\sharp[\![e_1 \bowtie e_2]\!]$ sont dérivés systématiquement par des algorithmes généraux
    paramétrés par l'implantation de $+^\sharp$, $-^\sharp$, ... sur $\mathcal{D}^\sharp$

Perte de précision : abstraction cartésienne

Perte de précision :

Les domaines non relationnels :

- abstraient les variables indépendemment les unes des autres ;
- effectuent implicitement une abstraction cartésienne :
  $X \in \mathcal{P}(\mathcal{E}) \mapsto \{\ \rho \in \mathbb{V} \to \mathbb{Z} \mid \forall V \in \mathbb{V},\ \exists \rho' \in X,\ \rho(V) = \rho'(V)\ \}$
- ils oublient donc les relations entre les variables.



$$X \in \{0,2\} \wedge Y \in \{0,2\} \wedge X + Y \leq 2 \quad \Longrightarrow \quad \{0,2\} \times \{0,2\}$$

Améliorer la précision

Deux types de domaines plus précis,
au delà des domaines non-relationnels.

- Les domaines disjonctifs
  vus au cours précédent.

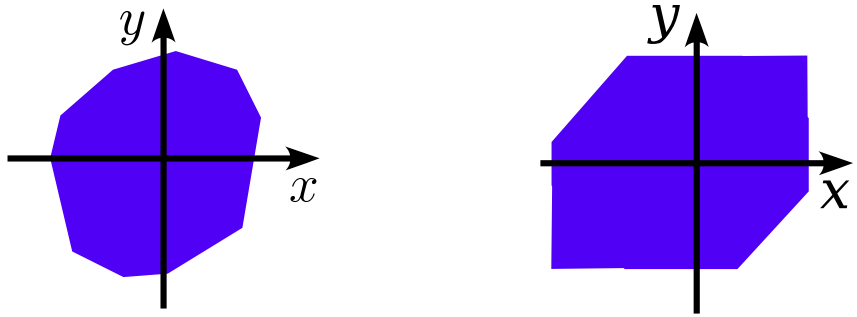
  e.g., permettent de représenter $(x = 1 \wedge y = 1) \vee (x = 2 \wedge y = 2)$, non convexe

- Les domaines relationnels convexes :
  domaines permettant de représenter des relations conjonctives entre variables

  - zones (e.g., $x \leq y + 10$)
  - polyèdres (e.g., $2x + 3y \leq z$)

Domaines numériques relationnels

Motivation



polyèdres zones

Représenter des relations entre variables est nécessaire :

- si l'assertion à prouver est relationnelle
  `i` $\leq$ `n` dans l'accès `a[i]` où la taille `n` du tableau n'est pas constante
- mais également pour éviter l'accumulation des pertes de précision
  même quand l'assertion à prouver est non-relationnelle
  Rappel : la composition d'abstractions optimales n'est pas forcément optimale

Les domaines relationnels sont plus coûteux
que les domaines non-relationnels.
$\Longrightarrow$ importance de trouver un compromis coût / précision, expressivité

Exemple : intervalles $\overset{\alpha}{\leftarrow}$ zones $\overset{\alpha}{\leftarrow}$ polyèdres

Exemple : affectations et tests relationnels

Exemple

$X \leftarrow \textbf{rand}(0, 10);$
$Y \leftarrow \textbf{rand}(0, 10);$
if $X \geq Y$ **then** $X \leftarrow Y$ **else skip**;
$D \leftarrow Y - X;$
assert $D \geq 0$

Analyse d'intervalles :

- $\mathrm{C}^\sharp[\![X \geq Y]\!]$ est abstrait par l'identité ;
  c'est l'abstraction optimale !
  si $R^\sharp \stackrel{\text{def}}{=} [X \mapsto [0, 10], Y \mapsto [0, 10]]$
  alors : $\mathrm{S}^\sharp[\![\textbf{if } X \geq Y \textbf{ then } X \leftarrow Y \textbf{ else skip}]\!] R^\sharp = R^\sharp$
- $D \leftarrow Y - X$ donne $D \in [0, 10] -^\sharp [0, 10] = [-10, 10]$;
- l'assertion $D \geq 0$ n'est pas prouvée.

Exemple : affectations et tests relationnels

Exemple

$X \leftarrow \textbf{rand}(0, 10)$;
$Y \leftarrow \textbf{rand}(0, 10)$;
if $X \geq Y$ **then** $X \leftarrow Y$ **else skip**;
$D \leftarrow Y - X$;
assert $D \geq 0$

Solution : utiliser un domaine relationnel, capable de :

- représenter explicitement l'information $X \leq Y$
- inférer que $X \leq Y$ est vrai après **if** $X \geq Y$ **then** $X \leftarrow Y$ **else skip**
  $X \leq Y$ est vrai après $X \leftarrow Y$ quand $X \geq Y$
  et après **skip** quand $X < Y$
- utiliser $X \leq Y$ pour en déduire que $Y - X \geq 0$, donc $D \geq 0$.

Note :

l'invariant recherché, $D \geq 0$, peut être représenté exactement dans le domaine des intervalles mais l'inférence et la preuve de $D \geq 0$ nécessitent localement un domaine plus expressif.

Invariant de boucle relationnel

Exemple

$I \leftarrow 1;\ X \leftarrow 0;$
while $I \leq 1000$ **do**
  $I \leftarrow I + 1;$
  $X \leftarrow X + 1;$
done;
assert $X \leq 1000$

Analyse d'intervalles :

- après deux itérations avec élargissement, nous trouvons :
  comme invariant de boucle : $I \in [1, +\infty]$ et $X \in [0, +\infty]$
  après la boucle : $I \in [1001, +\infty]$ et $X \in [0, +\infty] \Longrightarrow$ **assert** non prouvé
- en utilisant des itérations décroissantes après l'élargissement, nous avons :
  comme invariant de boucle : $I \in [1, 1001]$ et $X \in [0, +\infty]$
  après la boucle : $I = 1001$ et $X \in [0, +\infty] \Longrightarrow$ **assert** non prouvé
  le test $I \leq 1000$ raffine bien I, mais ne donne aucune information sur X
- sans élargissement, nous avons $I = 1001$ et $X = 1000$
  $\Longrightarrow$ **assert** est prouvé
  mais cela nécessite 1000 itérations ! ($\simeq$ calcul de point fixe concret)

Invariant de boucle relationnel

Exemple

$I \leftarrow 1;\ X \leftarrow 0;$
while $I \leq 1000$ **do**
  $I \leftarrow I + 1;$
  $X \leftarrow X + 1;$
done;
assert $X \leq 1000$

Solution : domaines relationnels

- inférer un invariant de boucle relationnel : $I = X + 1 \wedge 1 \leq I \leq 1001$
  qui représente de manière compacte l'ensemble de toutes les itérations
  - $I = X + 1$ est vrai avant d'entrer dans la boucle, car $1 = 0 + 1$
  - $I = X + 1$ est invariant par un tour de la boucle $I \leftarrow I + 1; X \leftarrow X + 1$
  - cet invariant peut être inféré en deux itérations avec élargissement dans le domaine des polyèdres !
- après propagation du test de sortie de la boucle $I > 1000$, nous trouvons :
  - $I = 1001$
  - $X = I - 1 = 1000 \Longrightarrow$ **assert** est prouvé

Note :

l'invariant recherché en fin de boucle est représentable dans les intervalles : $X \leq 1000$
mais nous avons besoin pour le trouver d'un invariant de boucle strictement plus expressif

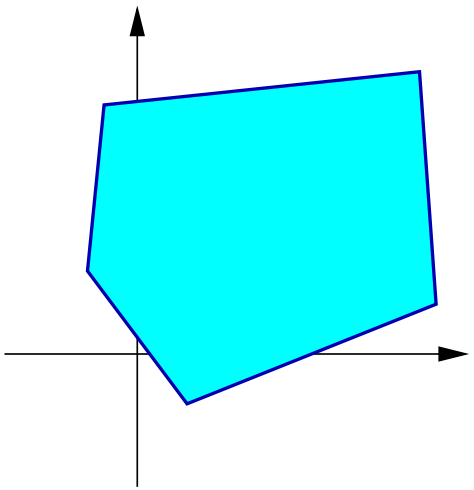
Domaine abstrait des polyèdres

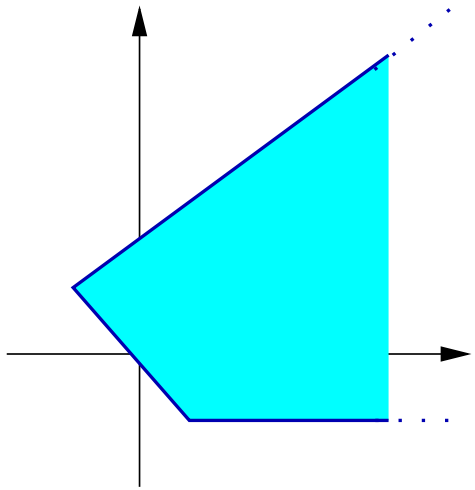
Le domaine des polyèdres

Nous cherchons comme invariants des conjonctions d'inégalités affines :
$\wedge_j \left(\sum_{i=1}^{n} \alpha_{ij} V_i \geq \beta_j\right)$ où les coefficients α_{ij} et β_j sont inférés automatiquement

Le domaine des polyèdres a été proposé par Cousot et Halbwachs en 1978.

$$\mathcal{E}^\sharp \simeq \{ \text{ polyèdres convexes clos de } \mathbb{V} \to \mathbb{R} \}$$





Notes :

- un polyèdre n'est pas nécessairement borné ;
- nous raisonnons dans $\mathbb{R}$ pour exploiter les résultats de l'algèbre linéaire, sans perte d'expressivité (un sous-ensemble de $\mathbb{Z}$ est aussi un sous-ensemble de $\mathbb{R}\ldots$).

Double description des polyèdres

Il existe deux manière duales de décrire un polyèdre.
(théorème de Weyl–Minkowski)

Représentation par contraintes

Sous forme matricielle : $\langle \mathbf{M}, \vec{C} \rangle$ où $\mathbf{M} \in \mathbb{Q}^{m \times n}$ et $\vec{C} \in \mathbb{Q}^m$
représente $\gamma(\langle \mathbf{M}, \vec{C} \rangle) \stackrel{\text{def}}{=} \{ \vec{V} \mid \mathbf{M} \times \vec{V} \geq \vec{C} \}$

De manière équivalente, sous forme d'ensemble de contraintes
$\{ \sum_i \alpha_{ij} V_i \geq \beta_j \}$.
(chaque contrainte est une ligne de la matrice...)

Représentation par générateurs

$[\mathbf{P}, \mathbf{R}]$ où

- $\mathbf{P} \in \mathbb{Q}^{n \times p}$ est un ensemble de p sommets : $\vec{P}_1, \ldots, \vec{P}_p$
- $\mathbf{R} \in \mathbb{Q}^{n \times r}$ est un ensemble de r rayons : $\vec{R}_1, \ldots, \vec{R}_r$

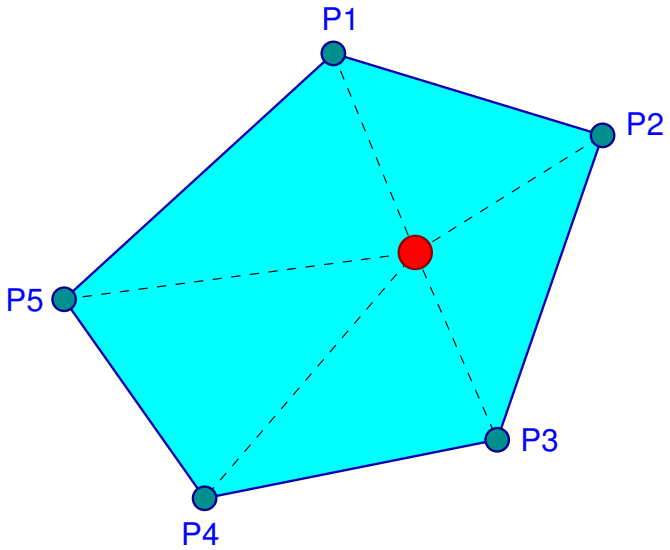
$$\gamma([\mathbf{P}, \mathbf{R}]) \stackrel{\text{def}}{=} \{ (\textstyle\sum_{j=1}^{p} \alpha_j \vec{P}_j) + (\sum_{j=1}^{r} \beta_j \vec{R}_j) \mid \forall j, \alpha_j, \beta_j \geq 0 \colon \sum_{j=1}^{p} \alpha_j = 1 \}$$

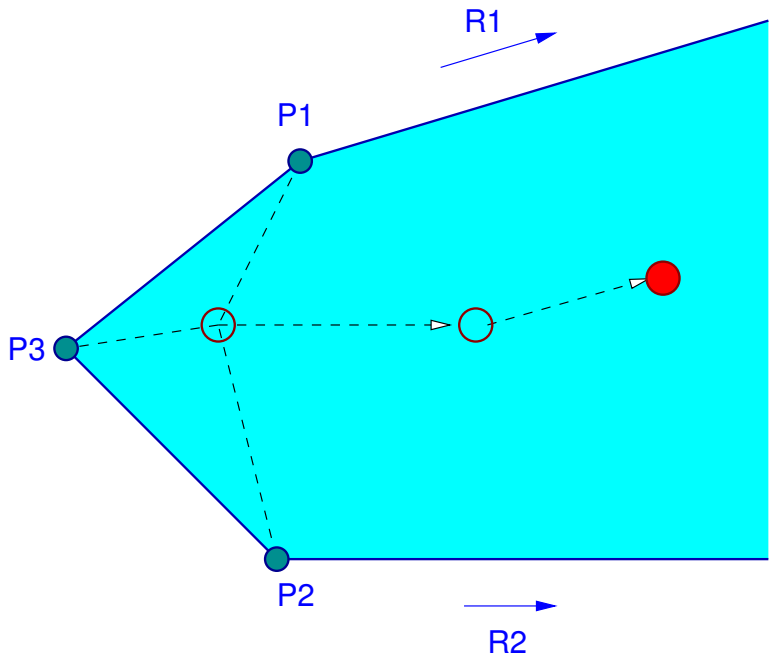
Note : les coordonnées et les coefficients sont dans $\mathbb{Q}$, pour être représentables en machine.

Double description des polyèdres (suite)

Exemples de représentations par générateurs :

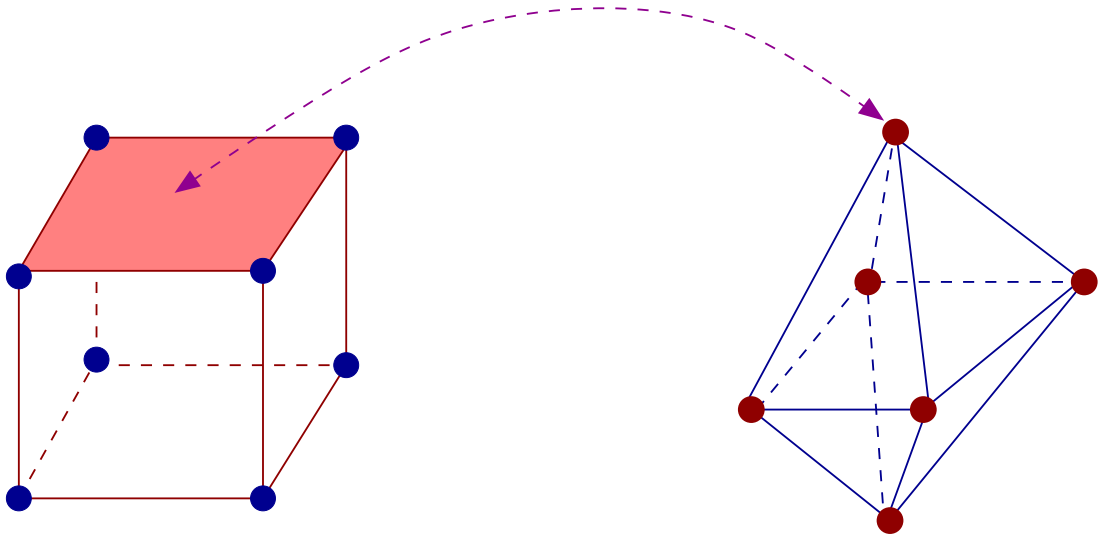
$$\gamma([\mathbf{P},\mathbf{R}]) \stackrel{\text{def}}{=} \{\, (\textstyle\sum_{j=1}^{p} \alpha_j \vec{P}_j) + (\sum_{j=1}^{r} \beta_j \vec{R}_j) \,|\, \forall j, \alpha_j, \beta_j \geq 0 : \sum_{j=1}^{p} \alpha_j = 1 \,\}$$





- les sommets définissent une enveloppe convexe bornée ;
- les rayons permettent de représenter des polyèdres non bornés.

Notion de dualité dans les polyèdres ⋆ ⋆⋆



P $\qquad\qquad$ P^*

$0x + 0y + 1z \leq 1 \quad \Longleftrightarrow \quad (0, 0, 1)$

Dualité : P^* est le dual de P

- générateurs et contraintes peuvent être vus comme des vecteurs ;
- les générateurs de P^* correspondent alors aux contraintes de P ;
- et les contraintes de P^* aux générateurs de P ;
- idempotence : $P^{**} = P$.

Double description : avantage et inconvénient

Avantage :

Les opérations abstraites sont généralement très faciles
en partant de la bonne représentation

(exemples : contraintes pour $\cap^\sharp$, générateurs pour $\cup^\sharp$)

$\Longrightarrow$ l'algorithmique des polyèdres se réduit à un seul algorithme complexe et coûteux : le passage d'une représentation à l'autre.

Inconvénient :

Passer d'une représentation à l'autre peut générer
une explosion exponentielle de la taille de la représentation !

Exemple : un hypercube dans $\mathbb{R}^n$, avec des faces parallèles aux axes

- a $2n$ contraintes ;
- mais 2^n générateurs (les sommets de l'hypercube) ;
- forme abstraite rencontrée fréquemment en analyse de programmes !

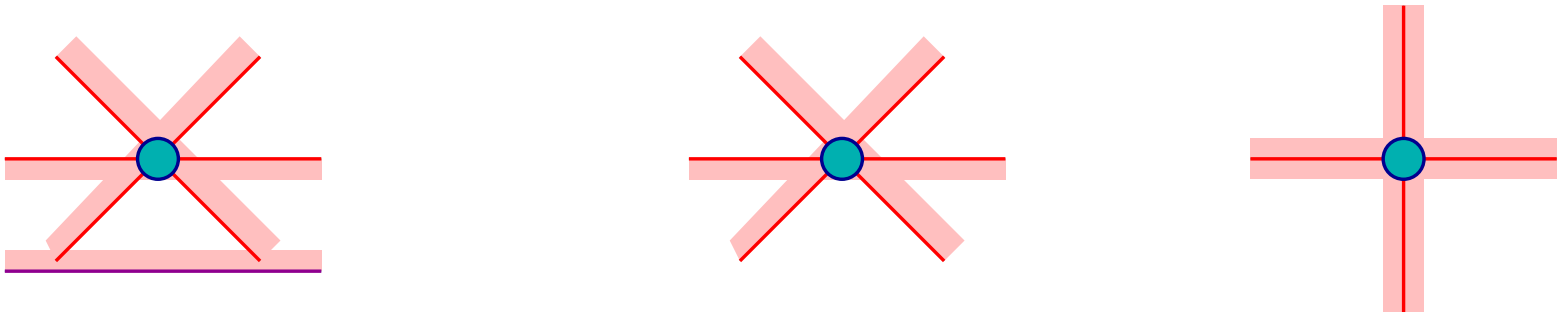
Nous ne sommes pas libres de choisir la représentation la plus compacte ;
le choix de représentation est dicté par les besoins des opérations abstraites...

Unicité des représentations ⋆⋆⋆

Représentations minimales

- Un système de contraintes / de générateurs est minimal si aucune contrainte / générateur ne peut être omise sans changer la concrétisation.
- Les représentations, même minimales, ne sont pas uniques !
  et deux représentations minimales d'un même polyèdre n'ont pas forcément la même taille

Exemple : trois systèmes de contraintes représentant un point

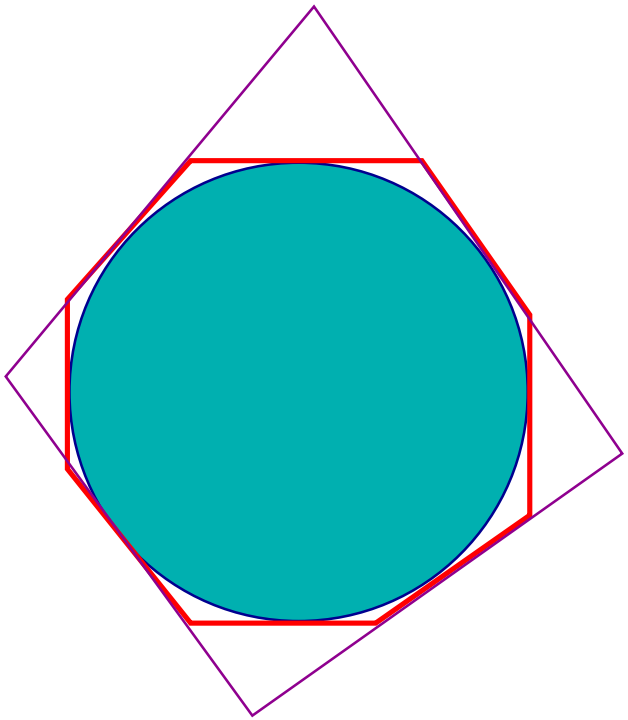


(a) (b) (c)

- (a) $y + x \geq 0, y - x \geq 0, y \leq 0, y \geq -5$ non minimal
- (b) $y + x \geq 0, y - x \geq 0, y \leq 0$ minimal
- (c) $x \leq 0, x \geq 0, y \leq 0, y \geq 0$ minimal

Borne sur les représentations des polyèdres

- Il n'y a pas de borne sur la taille des représentations de polyèdres.
  même sur les représentations minimales
- Il n'y a pas d'opérateur d'abstraction α ;
  pas de représentation optimale pour certains ensembles de points
  $\Longrightarrow$ il n'y a donc pas de correspondance de Galois
  ni de meilleur abstraction pour tout opérateur.



Exemple :

un disque a une infinité de sur-approximations par des polyèdres ;
aucune sur-approximation n'est la meilleur.

Passage de représentation : algorithme de Cherknikova ⋆ ⋆⋆

Algorithme de Chernikova (1968), amélioré par LeVerge (1992) :

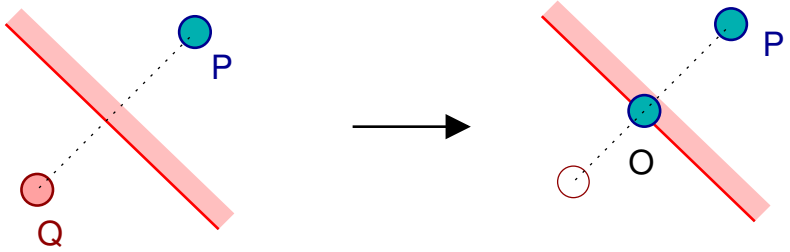
- passe d'un système de contraintes à un système de générateurs équivalent ;
- par dualité, convertit également des générateurs en contraintes ;
- minimise la représentation à la volée.

Intuition : algorithme incrémental

- part d'une représentation en générateurs de $\mathbb{R}^n$;
- ajoute les contraintes une par une ;
- filtre les générateurs pour ne garder que ceux qui satisfont chaque nouvelle contrainte ;
- déplace les autres générateurs jusqu'à ce qu'ils satisfassent la contrainte.
  i.e., qu'ils *saturent* la contrainte

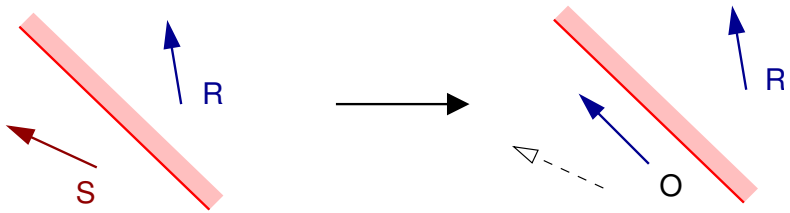
Algorithme de Cherknikova : illustration ⋆ ⋆⋆

Exemple : déplacement de sommets et de rayons.



Pour chaque paire P, Q de sommets :

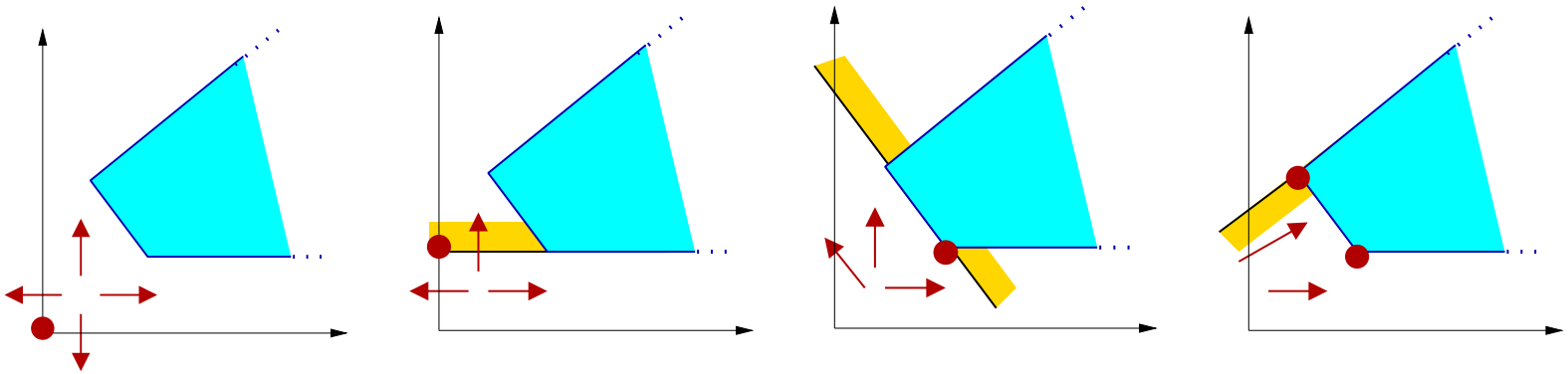
- si P satisfait la contrainte
- et Q ne satisfait pas la contrainte
- Q est déplacé vers P jusqu'à toucher l'hyper-plan supportant la contrainte



Pour chaque paire R, S de rayons :

- si R satisfait la contrainte
- et S ne satisfait pas la contrainte
- S est tourné vers R jusqu'à être parallèle à l'hyper-plan de la contrainte

Algorithme de Cherknikova : exemple ⋆ ⋆⋆



Ajout de trois contraintes, une par une :

	$\mathbf{P}_0 = \{(0,0)\}$	$\mathbf{R}_0 = \{(1,0), (-1,0), (0,1), (0,-1)\}$
$Y \geq 1$	$\mathbf{P}_1 = \{(0,1)\}$	$\mathbf{R}_1 = \{(1,0), (-1,0), (0,1)\}$
$X + Y \geq 3$	$\mathbf{P}_2 = \{(2,1)\}$	$\mathbf{R}_2 = \{(1,0), (-1,1), (0,1)\}$
$X - Y \leq 1$	$\mathbf{P}_3 = \{(2,1), (1,2)\}$	$\mathbf{R}_3 = \{(0,1), (1,1)\}$

nous avons omis les générateurs redondants générés par cette version naïve de l'algorithme ; l'algoroithme complet sait supprimer ces générateurs automatiquement

Opérateurs abstraits sur les polyèdres

Opérations ensemblistes :

Si $X^\sharp, Y^\sharp \neq \bot$, nous définissons :

$$X^\sharp \subseteq^\sharp Y^\sharp \quad \overset{\text{def}}{\iff} \quad \begin{cases} \forall \vec{P} \in \mathbf{P}_{X^\sharp} : \mathbf{M}_{Y^\sharp} \times \vec{P} \geq \vec{C}_{Y^\sharp} \\ \forall \vec{R} \in \mathbf{R}_{X^\sharp} : \mathbf{M}_{Y^\sharp} \times \vec{R} \geq \vec{0} \end{cases}$$

chaque générateur de $X^\sharp$ satisfait toutes les contraintes de $Y^\sharp$

$$X^\sharp =^\sharp Y^\sharp \quad \overset{\text{def}}{\iff} \quad X^\sharp \subseteq^\sharp Y^\sharp \text{ et } Y^\sharp \subseteq^\sharp X^\sharp$$

double inclusion

$$X^\sharp \cap^\sharp Y^\sharp \quad \overset{\text{def}}{=} \quad \left\langle \begin{bmatrix} \mathbf{M}_{X^\sharp} \\ \mathbf{M}_{Y^\sharp} \end{bmatrix}, \begin{bmatrix} \vec{C}_{X^\sharp} \\ \vec{C}_{Y^\sharp} \end{bmatrix} \right\rangle$$

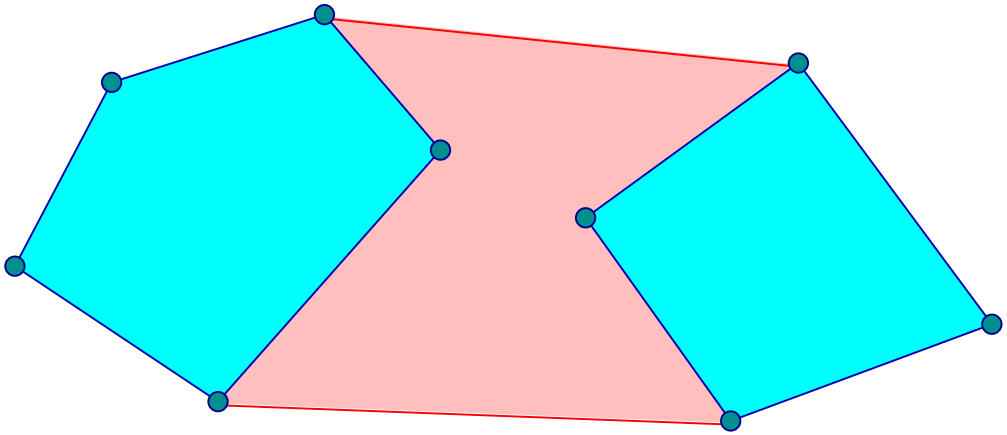
union des ensembles de contraintes

$\subseteq^\sharp$, $=^\sharp$ et $\cap^\sharp$ sont exacts dans $\mathcal{P}(\mathbb{V} \to \mathbb{R})$

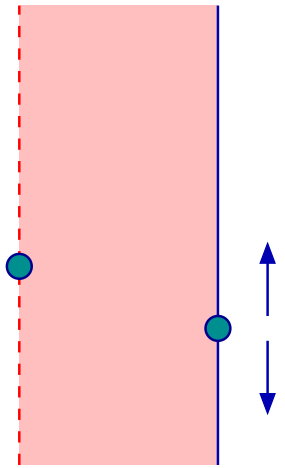
Opérateurs abstraits sur les polyèdres (suite)

Union : $X^\sharp \cup^\sharp Y^\sharp \stackrel{\text{def}}{=} [\,[\mathbf{P}_{X^\sharp}\ \mathbf{P}_{Y^\sharp}],\ [\mathbf{R}_{X^\sharp}\ \mathbf{R}_{Y^\sharp}]\,]$ union des ensembles de générateurs

Exemples :



deux polyèdres bornés



un point et une ligne

$\cup^\sharp$ est optimal dans $\mathcal{P}(\mathbb{V} \to \mathbb{R})$:

(α n'est pas toujours défini, mais $\alpha(\gamma(X^\sharp) \cup \gamma(Y^\sharp))$ existe toujours)

$\Longrightarrow$ clôture topologique de l'enveloppe convexe de $\gamma(X^\sharp) \cup \gamma(Y^\sharp)$:

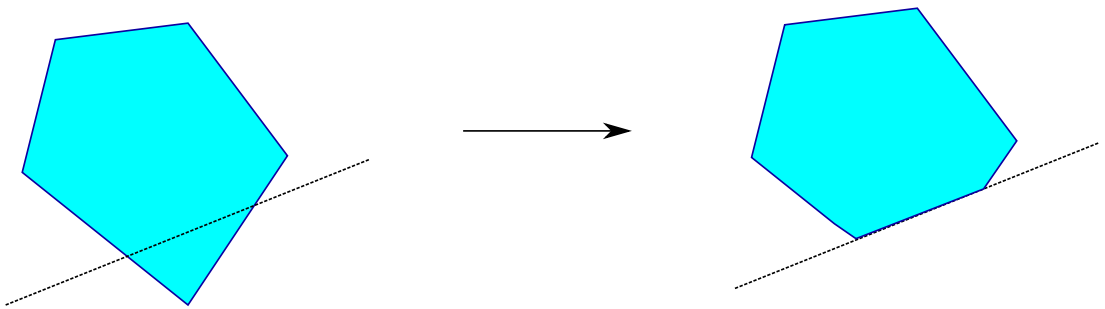
- tous nos polyèdres sont convexes ;
- la clôture sert en cas de polyèdres infinis et permet une représentation avec des contraintes non-strictes $\overset{\star}{\star}$

Opérateurs abstraits sur les polyèdres (suite)

Test affine :

$$\mathrm{C}^\sharp[\![\textstyle\sum_i \alpha_i V_i \geq \beta]\!] X^\sharp \stackrel{\text{def}}{=} \left\langle \begin{bmatrix} \mathbf{M}_{X^\sharp} \\ \alpha_1 \cdots \alpha_n \end{bmatrix}, \begin{bmatrix} \vec{C}_{X^\sharp} \\ \beta \end{bmatrix} \right\rangle$$

$$\mathrm{C}^\sharp[\![\textstyle\sum_i \alpha_i V_i = \beta]\!] X^\sharp \stackrel{\text{def}}{=} \mathrm{C}^\sharp[\![\textstyle\sum_i \alpha_i V_i \geq -\beta]\!] (\mathrm{C}^\sharp[\![\textstyle\sum_i (-\alpha_i) V_i \geq \beta]\!] X^\sharp)$$

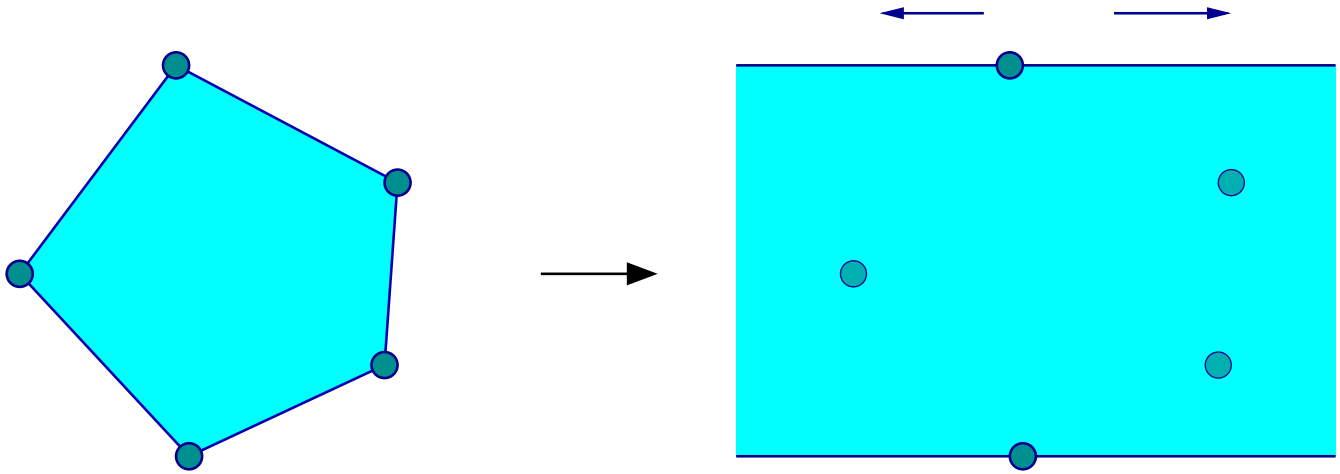


- simple ajout de contraintes ;
- ces opérateurs sont exacts ;
- les autres tests peuvent être abstraits par $\mathrm{C}^\sharp[\![c]\!] X^\sharp \stackrel{\text{def}}{=} X^\sharp$.
  (sûr mais très peu précis)

Opérateurs abstraits sur les polyèdres (suite)

Affectation non-déterministe :

$$S^\sharp[\![V_j \leftarrow \textbf{rand}(-\infty, +\infty)]\!] X^\sharp \stackrel{\text{def}}{=} [\, \mathbf{P}_{X^\sharp},\ [\, \mathbf{R}_{X^\sharp}\ \ \vec{x}_j\ \ (-\vec{x}_j)\,]\,]$$



- dans le concret :
  $S[\![V_j \leftarrow \textbf{rand}(-\infty, +\infty)]\!] R = \{\, \rho[V_j \mapsto v] \mid \rho \in R,\ v \in \mathbb{R} \,\}$;
- dans l'abstrait :
  ajout de deux rayons, dans la direction de la variable "oubliée" ;
- cet opérateur est exact dans $\mathcal{P}(\mathbb{V} \to \mathbb{R})$.

Opérateurs abstraits sur les polyèdres (suite)

Affectation affine :

$$S^\sharp[\![V_j \leftarrow \sum_i \alpha_i V_i + \beta]\!] X^\sharp \stackrel{\text{def}}{=}$$

si $\alpha_j \neq 0, \langle \mathbf{M}, \vec{C} \rangle$ où V_j est remplacé par $\frac{1}{\alpha_j}(V_j - \sum_{i \neq j} \alpha_i V_i - \beta)$

si $\alpha_j = 0, C^\sharp[\![V_j = \sum_i \alpha_i V_i + \beta]\!] (S^\sharp[\![V_j \leftarrow \textbf{rand}(-\infty, +\infty)]\!] X^\sharp)$

- si $\alpha_j \neq 0$, nous effectuons une substitution par l'inverse
  Exemple : $X \leftarrow 2X + Y$
  X est remplacé pas $(X - Y)/2$ dans le système de contraintes
  c.f., logique de Hoare
- si $\alpha_j = 0$, l'affectation n'est pas inversible
  Exemple : $X \leftarrow Y$
  oubli de l'ancienne valeur de X
  puis ajout de la contrainte $X = Y$
- l'opération est exacte dans $\mathcal{P}(\mathbb{V} \to \mathbb{R})$
- les affectations non-affines peuvent être modélisées par :
  $S^\sharp[\![V \leftarrow e]\!] \stackrel{\text{def}}{=} S^\sharp[\![V \leftarrow \textbf{rand}(-\infty, +\infty)]\!]$
  (sûr mais peu précis)

Opérateurs abstraits sur les polyèdres (suite)

Affectation affine :

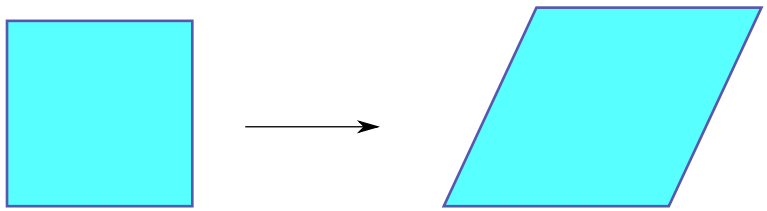
$$S^\sharp[\![V_j \leftarrow \sum_i \alpha_i V_i + \beta]\!] X^\sharp \stackrel{\text{def}}{=}$$

si $\alpha_j \neq 0, \langle \mathbf{M}, \vec{C} \rangle$ où V_j est remplacé par $\frac{1}{\alpha_j}(V_j - \sum_{i \neq j} \alpha_i V_i - \beta)$

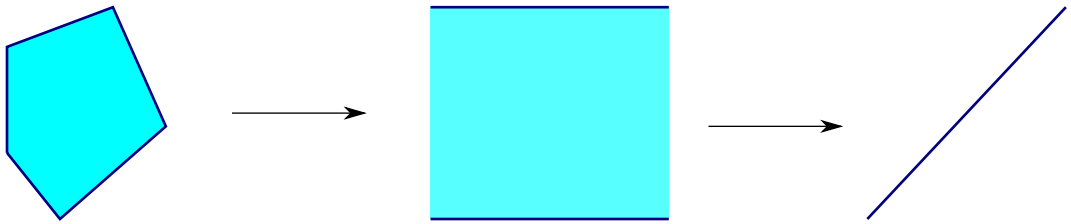
si $\alpha_j = 0, C^\sharp[\![V_j = \sum_i \alpha_i V_i + \beta]\!] (S^\sharp[\![V_j \leftarrow \textbf{rand}(-\infty, +\infty)]\!] X^\sharp)$

Exemples :

$X \leftarrow X + Y$



$X \leftarrow Y$



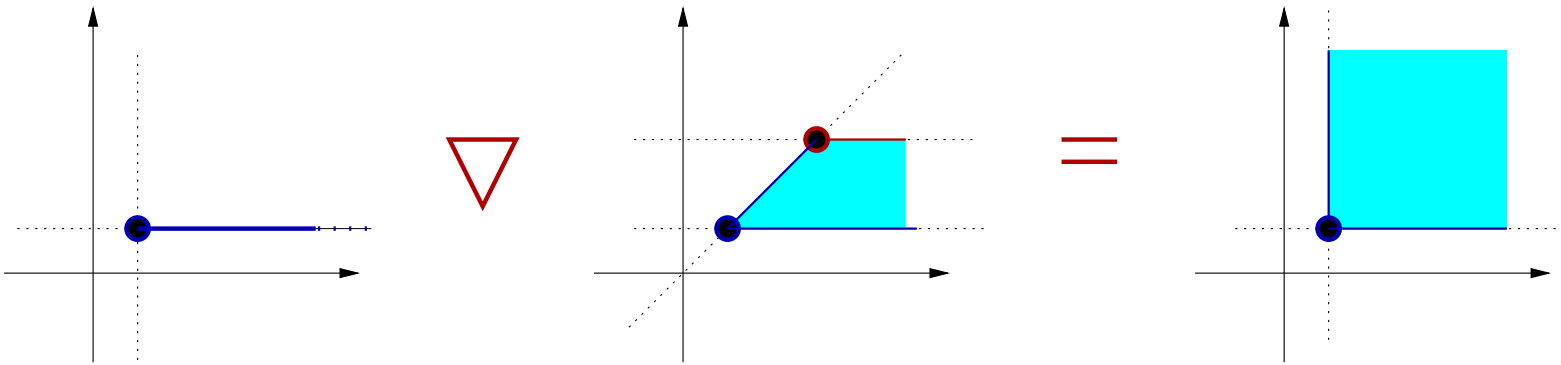
Élargissement naïf sur les polyèdres

$\mathcal{E}^\sharp$ a des chaînes infinies strictement croissantes
$\Longrightarrow$ nous avons besoin d'un élargissement

Définition : $X^\sharp \triangledown Y^\sharp \stackrel{\text{def}}{=} \{ c \in X^\sharp \mid Y^\sharp \subseteq^\sharp \{c\} \}$

- garde les contraintes de $X^\sharp$ satisfaites par $Y^\sharp$;
- contrairement à $\cup^\sharp$, pas de création de nouvelle contrainte ;
- $\triangledown$ réduit l'ensemble des contraintes
  $\Longrightarrow$ la terminaison est garantie.

Exemple :



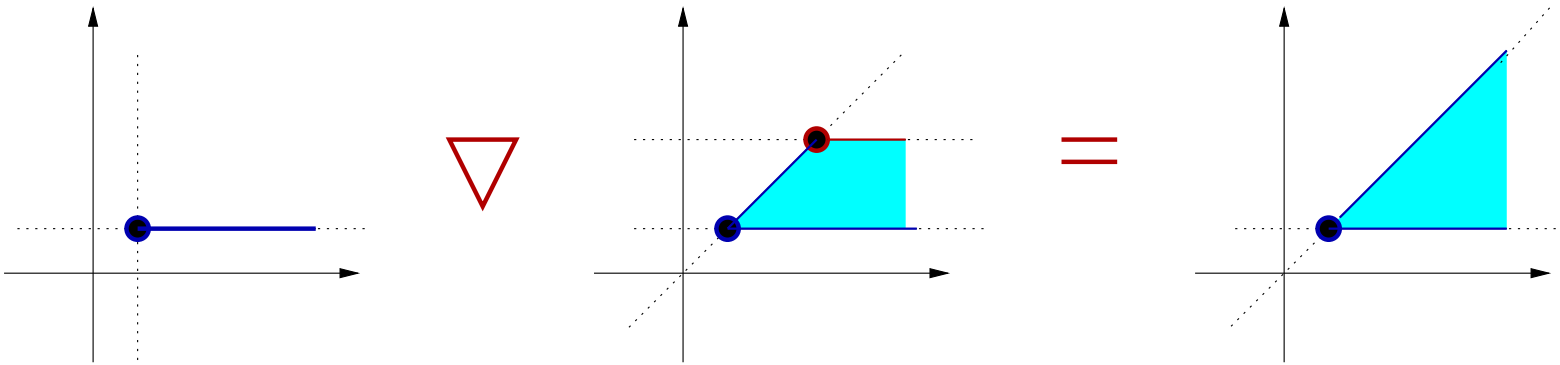
$\{X \geq 1, Y \geq 1, Y \leq 1\} \triangledown \{X \geq 1, Y \geq 1, Y \leq 2, X \geq Y\} = \{X \geq 1, Y \geq 1\}$

Élargissements avancés sur les polyèdres ⋆ ⋆⋆

Prise en compte des contraintes de $Y^\sharp$

$$X^\sharp \triangledown Y^\sharp \overset{\text{def}}{=} \{ c \in X^\sharp \mid Y^\sharp \subseteq^\sharp \{c\} \} \cup \{ c \in Y^\sharp \mid \exists c' \in X^\sharp : X^\sharp =^\sharp (X^\sharp \setminus c') \cup \{c\} \}$$

garde aussi les contraintes de $Y^\sharp$ équivalentes à des contraintes de $X^\sharp$;
c'est un remède à l'absence d'unicité des représentations de polyèdres



$\{X \geq 1, Y \geq 1, Y \leq 1\} \triangledown \{X \geq 1, Y \geq 1, Y \leq 2, X \geq Y\} = \{X \geq 1, X \geq Y\}$

Élargissement étagé

paramétré par un ensemble fini de contraintes T :

$$X^\sharp \triangledown Y^\sharp \overset{\text{def}}{=} \{ c \in X^\sharp \mid Y^\sharp \subseteq^\sharp \{c\} \} \cup \{ c \in T \mid X^\sharp \subseteq^\sharp \{c\} \wedge Y^\sharp \subseteq^\sharp \{c\} \}$$

ajoute les contraintes de T stables, comme pour les intervalles...

Exemple d'analyse avec les polyèdres

Exemple

$X \leftarrow 2; I \leftarrow 0;$
while $I < 10$ **do**
  if rand$(0,1) = 0$ **then** $X \leftarrow X + 2$ **else** $X \leftarrow X - 3;$
  $I \leftarrow I + 1$
done

Invariant de boucle :

itérations croissantes avec élargissement :

$$\begin{array}{lcl} X_1^\sharp & = & \{X = 2, I = 0\} \\ X_2^\sharp & = & \{X = 2, I = 0\} \triangledown (\{X = 2, I = 0\} \cup^\sharp \{X \in [-1, 4],\ I = 1\}) \\ & = & \{X = 2, I = 0\} \triangledown \{\ I \in [0, 1],\ 2 - 3I \leq X \leq 2I + 2\ \} \\ & = & \{I \geq 0,\ 2 - 3I \leq X \leq 2I + 2\} \end{array}$$

itérations décroissantes : pour retrouver $I \leq 10$

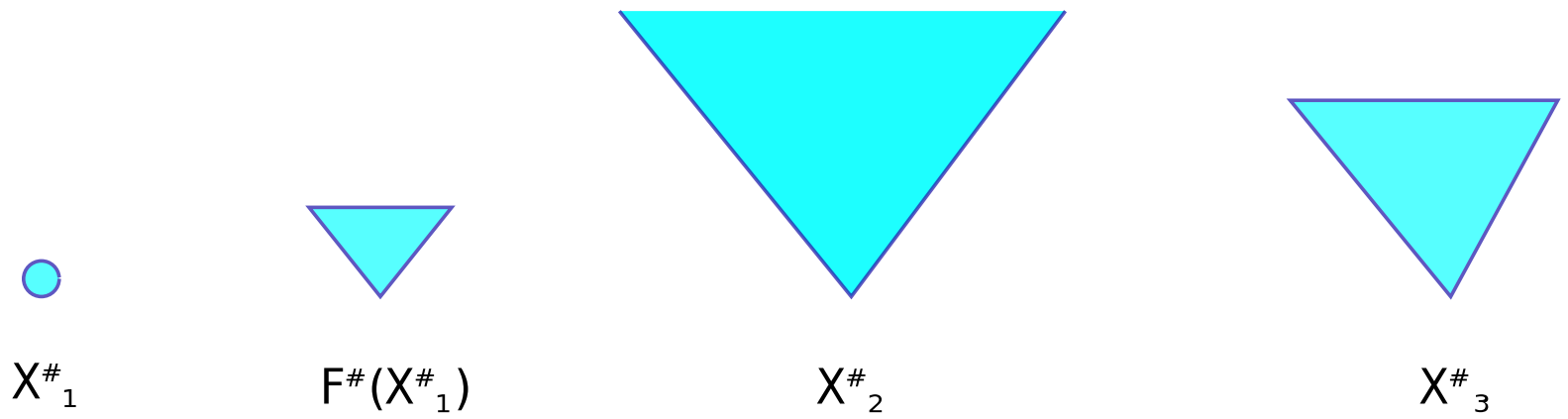
$$\begin{array}{lcl} X_3^\sharp & = & \{X = 2, I = 0\} \cup^\sharp \{\ I \in [1, 10],\ 2 - 3I \leq X \leq 2I + 2\ \} \\ & = & \{I \in [0, 10],\ 2 - 3I \leq X \leq 2I + 2\} \end{array}$$

nous trouvons, en sortie de boucle : $I = 10 \wedge X \in [-28, 22]$

Exemple d'analyse avec les polyèdres (illustration)

Exemple

$X \leftarrow 2; I \leftarrow 0;$
while $I < 10$ **do**
  if rand$(0,1) = 0$ **then** $X \leftarrow X + 2$ **else** $X \leftarrow X - 3;$
  $I \leftarrow I + 1$
done



$$
\begin{array}{lcl}
X_1^\sharp & = & \{X = 2, I = 0\} \\
X_2^\sharp & = & \{X = 2, I = 0\} \triangledown (\{X = 2, I = 0\} \cup^\sharp \{X \in [-1, 4],\ I = 1\}) \\
 & = & \{I \geq 0,\ 2 - 3I \leq X \leq 2I + 2\} \\
X_3^\sharp & = & \{X = 2, I = 0\} \cup^\sharp \{\ I \in [1, 10],\ 2 - 3I \leq X \leq 2I + 2\ \} \\
 & = & \{I \in [0, 10],\ 2 - 3I \leq X \leq 2I + 2\}
\end{array}
$$

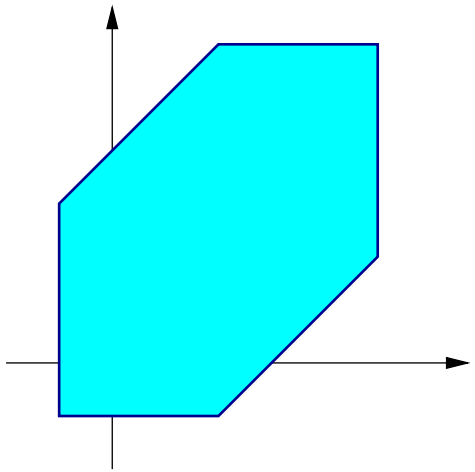
Domaine abstrait des zones

Le domaine des zones

Restriction des polyèdres à des contraintes de la forme :

$$\bigwedge V_i - V_j \leq c \text{ ou } \pm V_i \leq c, \quad c \in \mathbb{Q}$$

- contraintes d'intervalles $V_i \in [c, c']$;
- mais aussi, bornes sur les différences de variables $V_i - V_j \leq c$;
  $\Longrightarrow$ relations utiles dans les boucles, les accès de tableau, etc.
- un tel polyèdre est appelé une **zone** ;
- les calculs sur les zones sont plus efficaces que sur les polyèdres généraux.



on retrouve aussi les zones dans le *model-checking* des automates temporisés

Représentation sous forme de graphe

Les contraintes $V_j - V_i \leq c$ sont appelées contraintes de potentiel.
(car invariantes par l'ajout d'une constante à toutes les variables)

Graphe de potentiel : $\mathcal{G}$

Représentation d'un ensemble de contraintes de potentiel par :

- un graphe pondéré $\mathcal{G}$;
- dont les nœuds sont étiquetés par les variables du programme $\mathbb{V}$;
- un arc va de V_i à V_j pour chaque contrainte $V_j - V_i \leq c$;
- l'arc est annoté avec le poids c ;

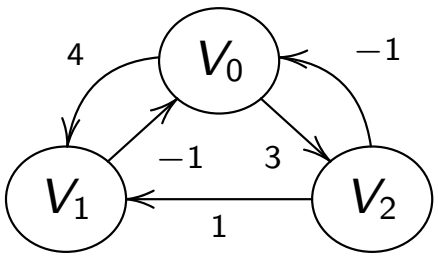
$\Longrightarrow$ nous allons utiliser des algorithmes de graphe.

Contraintes d'intervalles :

Représentées sous forme de contraintes de potentiel,
grâce à une variable spéciale, V_0, égale à la constante zéro :

- $V_i \leq c$ est encodé par $V_i - V_0 \leq c$
- $V_i \geq c$ est encodé par $V_0 - V_i \leq -c$

Exemple :



$V_1 \in [1, 4], V_2 \in [1, 3], V_1 - V_2 \leq 1$

Représentation sous forme de matrice

Matrices de différences bornées (*Difference Bound Matrices*, DBM)

Matrice d'adjacence **m** d'un graphe de potentiel $\mathcal{G}$:

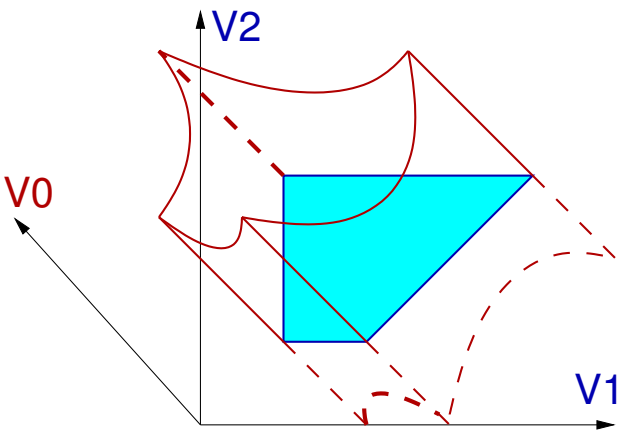
- **m** est carrée, de taille $n \times n$, à éléments dans $\mathbb{Q} \cup \{+\infty\}$;
- $m_{ij} = c < +\infty$ dénote la contrainte $V_j - V_i \leq c$;
- $m_{ij} = +\infty$ si il n'y a pas de borne supérieure à $V_j - V_i$;

$\Longrightarrow$ représentation plus compacte et plus pratique que les graphes.

Concrétisation :

$$\gamma(\mathbf{m}) \stackrel{\text{def}}{=} \{ (v_0, v_1, \ldots, v_n) \in \mathbb{R}^n \mid v_0 = 0 \wedge \forall i, j : v_j - v_i \leq m_{ij} \}$$

Exemple :



	V_0	V_1	V_2
V_0	$+\infty$	4	3
V_1	-1	$+\infty$	$+\infty$
V_2	-1	1	$+\infty$

Le treillis des matrices

$\mathcal{E}^\sharp$ contient les matrices DBMs, ainsi que $\bot$.

L'ordre naturel $\leq$ sur $\mathbb{Q} \cup \{+\infty\}$ est étendu point à point.

Si $\mathbf{m}, \mathbf{n} \neq \bot$:

$$\begin{array}{rcl} \mathbf{m} \sqsubseteq \mathbf{n} & \overset{\text{def}}{\iff} & \forall i, j : m_{ij} \leq n_{ij} \\ \mathbf{m} = \mathbf{n} & \overset{\text{def}}{\iff} & \forall i, j : m_{ij} = n_{ij} \\ [\mathbf{m} \sqcap \mathbf{n}]_{ij} & \overset{\text{def}}{=} & \min(m_{ij}, n_{ij}) \\ [\mathbf{m} \sqcup \mathbf{n}]_{ij} & \overset{\text{def}}{=} & \max(m_{ij}, n_{ij}) \\ [\top]_{ij} & \overset{\text{def}}{=} & +\infty \end{array}$$

$(\mathcal{E}^\sharp, \sqsubseteq, \sqcup, \sqcap, \bot, \top)$ est un treillis.

Notes :

- $\mathbf{m} \sqsubseteq \mathbf{n} \Longrightarrow \gamma(\mathbf{m}) \subseteq \gamma(\mathbf{n})$, mais pas la réciproque ;
- $\mathbf{m} = \mathbf{n} \Longrightarrow \gamma(\mathbf{m}) = \gamma(\mathbf{n})$, mais pas la réciproque.

Forme normale et test d'inclusion

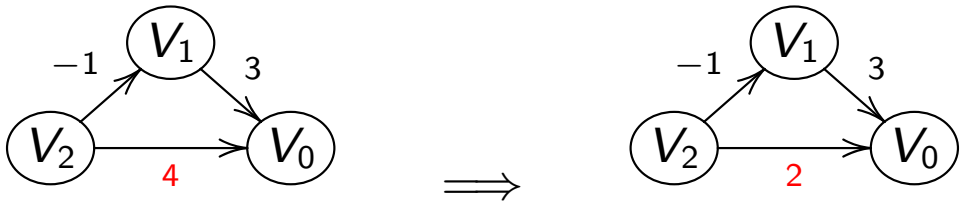
Problème : comment comparer $\gamma(\mathbf{m})$ et $\gamma(\mathbf{n})$?

Solution : définir une forme normale

Principe : propagation de contraintes

Ajouter deux contraintes permet de dériver une nouvelle contrainte et raffiner une borne existante.

$$\begin{cases} V_0 - V_1 \leq 3 \\ V_1 - V_2 \leq -1 \\ V_0 - V_2 \leq 4 \end{cases} \qquad \begin{cases} V_0 - V_1 \leq 3 \\ V_1 - V_2 \leq -1 \\ V_0 - V_2 \leq 2 \end{cases}$$



Généralisation : clôture par plus courts chemins $\mathbf{m}^*$

$$m^*_{ij} \stackrel{\text{def}}{=} \min_{\substack{N \\ \langle i = i_1, \ldots, i_N = j \rangle}} \sum_{k=1}^{N-1} m_{i_k\, i_{k+1}}$$

Bien défini uniquement si $\mathbf{m}$ n'a pas de cycle avec un poids total strictement négatif.

Algorithme de Floyd–Warshall

Propriétés :

- $\gamma(\mathbf{m}) = \emptyset \iff \mathcal{G}$ a un cycle de poids total strictement négatif
- si $\gamma(\mathbf{m}) \neq \emptyset$, la clôture par plus courts chemins $\mathbf{m}^*$ est une forme normale :

  $$\mathbf{m}^* = \min_{\sqsubseteq} \{ \mathbf{n} \mid \gamma(\mathbf{m}) = \gamma(\mathbf{n}) \}$$

  (chaque borne supérieure de contrainte est la plus stricte possible)
- si $\gamma(\mathbf{m}), \gamma(\mathbf{n}) \neq \emptyset$, alors
  - $\gamma(\mathbf{m}) = \gamma(\mathbf{n}) \iff \mathbf{m}^* = \mathbf{n}^*$
  - $\gamma(\mathbf{m}) \subseteq \gamma(\mathbf{n}) \iff \mathbf{m}^* \sqsubseteq \mathbf{n}$

Algorithme de Floyd–Warshall

$$\begin{cases} m_{ij}^0 & \stackrel{\text{def}}{=} & m_{ij} \\ m_{ij}^{k+1} & \stackrel{\text{def}}{=} & \min(m_{ij}^k, m_{ik}^k + m_{kj}^k) \end{cases}$$

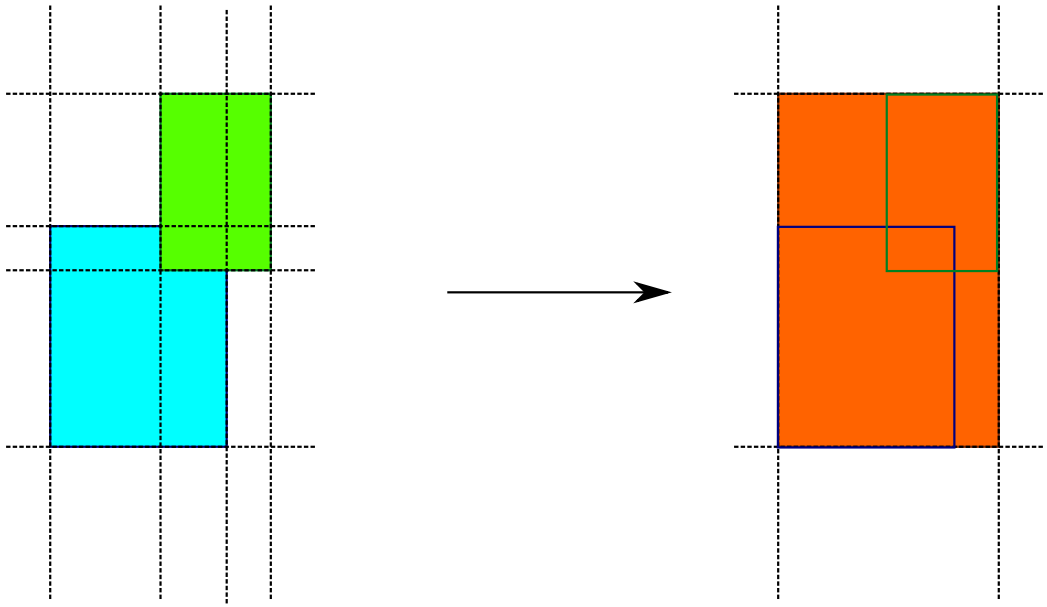
- algorithme classique, qui itère des propagations locales
- si $\gamma(\mathbf{m}) \neq \emptyset$, alors $\mathbf{m}^* = \mathbf{m}^{n+1}$ (forme normale)
- $\gamma(\mathbf{m}) = \emptyset \iff \exists i : m_{ii}^{n+1} < 0$ (teste du vide)
- $\mathbf{m}^{n+1}$ peut être calculé en temps $\mathcal{O}(n^3)$

Opérateurs abstraits

Union abstraite $\cup^\sharp$: version naïve : $\sqcup$ (rappel : $[\mathbf{m} \sqcup \mathbf{n}]_{ij} = \max(m_{ij}, n_{ij})$)

- $\sqcup$ est une abstraction sûre de $\cup$,

  mais $\gamma(\mathbf{m} \sqcup \mathbf{n})$ n'est pas forcément la plus petite zone contenant $\gamma(\mathbf{m})$ et $\gamma(\mathbf{n})$!



l'union de deux boîtes avec $\sqcup$ n'est pas plus précise dans les zones que dans les intervalles !

Opérateurs abstraits (suite)

Union abstraite $\cup^\sharp$: version précise : $\sqcup$ après clôture

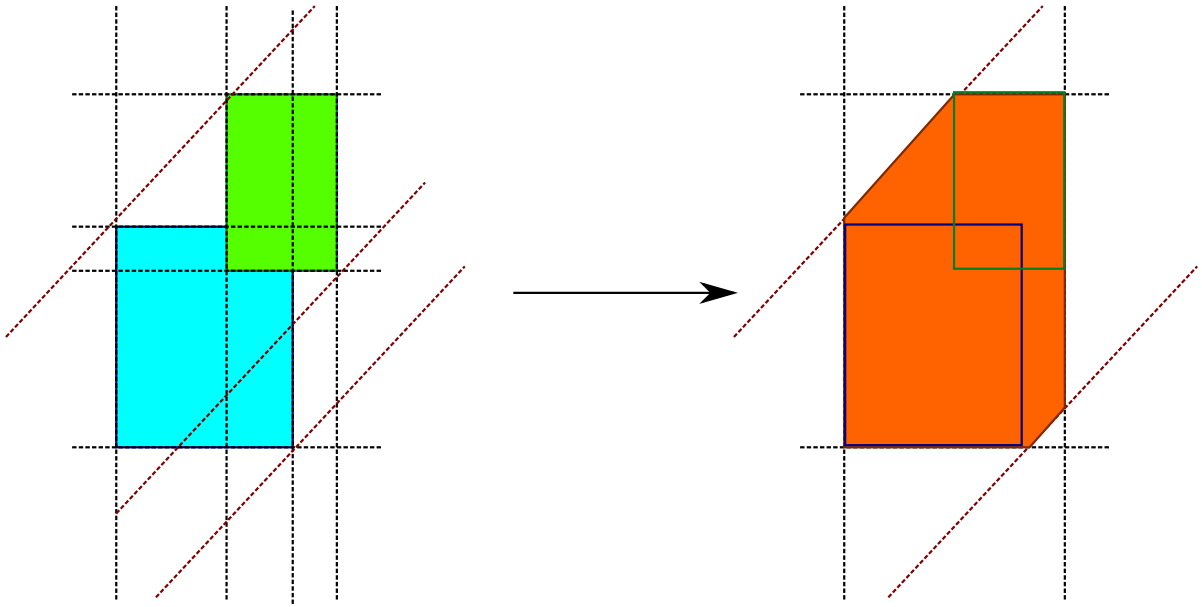
- $(\mathbf{m}^*) \sqcup (\mathbf{n}^*)$ est par contre optimal

  nous avons :

  $$(\mathbf{m}^*) \sqcup (\mathbf{n}^*) = \min_{\sqsubseteq} \{ \mathbf{o} \mid \gamma(\mathbf{o}) \supseteq \gamma(\mathbf{m}) \cup \gamma(\mathbf{n}) \}$$

  ce qui implique :

  $$\gamma((\mathbf{m}^*) \sqcup (\mathbf{n}^*)) = \min_{\subseteq} \{ \gamma(\mathbf{o}) \mid \gamma(\mathbf{o}) \supseteq \gamma(\mathbf{m}) \cup \gamma(\mathbf{n}) \}$$



après clôture, des contraintes $c \leq X - Y \leq d$ ajoutées permettent une union plus précise

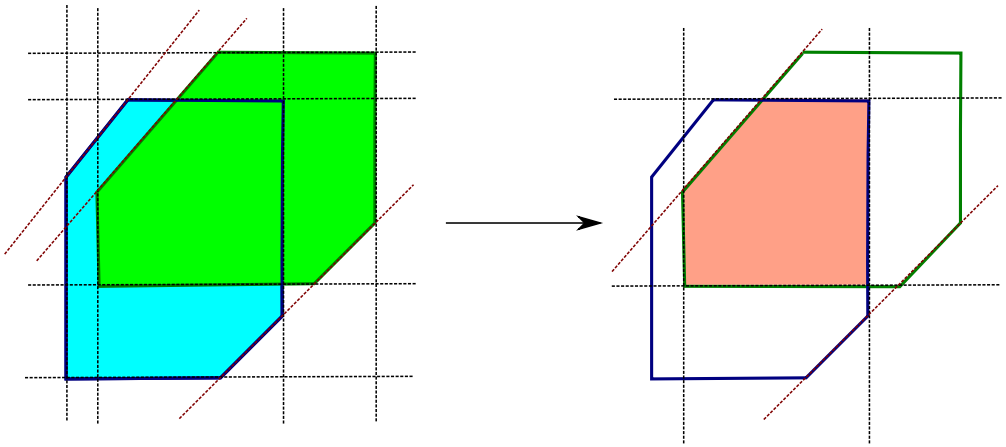
- $(\mathbf{m}^*) \sqcup (\mathbf{n}^*)$ est toujours close.

Opérateurs abstraits (suite)

Intersection abstraite $\cap^\sharp$: $\sqcap$ (rappel : $[\mathbf{m} \sqcap \mathbf{n}]_{ij} = \min(m_{ij}, n_{ij})$)

- $\mathcal{E}^\sharp$ est clos par intersection
  (Rappel : c'est le cas de la plus part des domaines)
- $\sqcap$ est toujours une abstraction exacte de $\cap$:

$$\gamma(\mathbf{m} \sqcap \mathbf{n}) = \gamma(\mathbf{m}) \cap \gamma(\mathbf{n})$$



- $(\mathbf{m}^*) \sqcap (\mathbf{n}^*)$ n'est pas forcément close...

Note :

L'ensemble des matrices closes avec $\bot$, et les opérations $\sqsubseteq$, $\sqcup$, $\lambda \mathbf{m}, \mathbf{n}.(\mathbf{m} \sqcap \mathbf{n})^*$ forme un sous-treillis.
γ est injective dans ce sous-treillis.

Opérateurs abstraits (suite)

Tests simples :

Si la contrainte peut être représentée exactement dans les zones,
il suffit de l'ajouter,
en modifiant un élément de la matrice.

$$\left[\mathrm{C}^\sharp[\![\, V_{j_0} - V_{i_0} \leq c \,]\!]\, \mathbf{m}\right]_{ij} \stackrel{\text{def}}{=} \begin{cases} \min(m_{ij}, c) & \text{si } (i,j) = (i_0, j_0), \\ m_{ij} & \text{sinon.} \end{cases}$$

$$\mathrm{C}^\sharp[\![\, V_{j_0} - V_{i_0} = [a,b] \,]\!]\, \mathbf{m} \stackrel{\text{def}}{=} \mathrm{C}^\sharp[\![\, V_{j_0} - V_{i_0} \leq b \,]\!]\, (\mathrm{C}^\sharp[\![\, V_{i_0} - V_{j_0} \leq -a \,]\!]\, \mathbf{m})$$

- ces opérations sont exactes ;
- pour les autres tests, nous pouvons utiliser l'identité :
  $\mathrm{C}^\sharp[\![\, e_1 \bowtie e_2 \,]\!]\, \mathbf{m} = \mathbf{m}$.

Opérateurs abstraits (suite)

Affectations simples :

Cas où la contrainte $X - e$ peut être représentée exactement dans les zones :

$$S^\sharp[\![V_{j_0} \leftarrow V_{i_0} + [a,b]]\!]\,\mathbf{m} \stackrel{\text{def}}{=} S^\sharp[\![V_{j_0} - V_{i_0} = [a,b]]\!](S^\sharp[\![V_{j_0} \leftarrow \mathbf{rand}(-\infty,+\infty)]\!]\,\mathbf{m}) \quad \text{si } i_0 \neq j_0$$

$$\left[S^\sharp[\![V_{j_0} \leftarrow V_{j_0} + [a,b]]\!]\,\mathbf{m}\right]_{ij} \stackrel{\text{def}}{=} \begin{cases} m_{ij} - a & \text{si } i = j_0 \text{ et } j \neq j_0 \\ m_{ij} + b & \text{si } i \neq j_0 \text{ et } j = j_0 \\ m_{ij} & \text{sinon.} \end{cases}$$

Cas non-déterministe : oubli de contraintes

$$\left[S^\sharp[\![V_{j_0} \leftarrow \mathbf{rand}(-\infty,+\infty)]\!]\,\mathbf{m}\right]_{ij} \stackrel{\text{def}}{=} \begin{cases} +\infty & \text{si } i = j_0 \text{ ou } j = j_0, \\ m^*_{ij} & \text{sinon.} \end{cases}$$

(non optimal si l'argument n'est pas clos !)

- ces opérations sont exactes ;
- pour les autres affectations, nous pouvons utiliser le cas non-déterministe :
  $S^\sharp[\![X \leftarrow e]\!]\,\mathbf{m} = S^\sharp[\![X \leftarrow \mathbf{rand}(-\infty,+\infty)]\!]\,\mathbf{m}$

Opérateurs abstraits (suite) ⋆ ⋆⋆

Cas général ⋆ ⋆⋆ : affectation approchée $V \leftarrow e$

Idée : utiliser l'évaluation dans les intervalles

- de e pour trouver des nouvelles bornes de V
- de $e - W$ pour des bornes de $V - W$ pour tout $W \neq V$

Exemple :

argument

$$\begin{cases} 0 \leq Y \leq 10 \\ 0 \leq Z \leq 10 \\ 0 \leq Y - Z \leq 10 \end{cases}$$

$$\Downarrow X \leftarrow Y - Z$$

$$\begin{cases} -10 \leq X \leq 10 \\ -20 \leq X - Y \leq 10 \\ -20 \leq X - Z \leq 10 \end{cases}$$

bornes pour V seul

$$\begin{cases} -10 \leq X \leq 10 \\ -10 \leq X - Y \leq 0 \\ -10 \leq X - Z \leq 10 \end{cases}$$

bornes pour tous les $V - W$

$$\begin{cases} 0 \leq X \leq 10 \\ -10 \leq X - Y \leq 0 \\ -10 \leq X - Z \leq 10 \end{cases}$$

affectation optimale $\alpha \circ \mathrm{S}[\![V \leftarrow e]\!] \circ \gamma$

$\Longrightarrow$ bon compromis entre coût et précision !

Élargissement

Le domaine des zones a des chaînes infinies strictement croissantes
$\Longrightarrow$ un élargissement ∇ est nécessaire.

Élargissement ∇

$$[\mathbf{m} \nabla \mathbf{n}]_{ij} \stackrel{\text{def}}{=} \begin{cases} m_{ij} & \text{si } n_{ij} \leq m_{ij} \\ +\infty & \text{sinon} \end{cases}$$

Comme pour les intervalles, les bornes non stables sont supprimées.

Itérations avancées $\stackrel{\star}{\star}\stackrel{\star}{\star}$:

Les améliorations connues des intervalles s'appliquent également :

- élargissement à étages

$$[\mathbf{m} \nabla_T \mathbf{n}]_{ij} \stackrel{\text{def}}{=} \begin{cases} m_{ij} & \text{si } n_{ij} \leq m_{ij} \\ \min \{ t \in T \mid t \geq n_{ij} \} & \text{sinon} \end{cases}$$

- itérations décroissantes avec rétrécissement $\triangle$

$$[\mathbf{m} \triangle \mathbf{n}]_{ij} \stackrel{\text{def}}{=} \begin{cases} n_{ij} & \text{si } m_{ij} = +\infty \\ m_{ij} & \text{sinon} \end{cases}$$

Autres domaines faiblement relationnels

Domaine faiblement relationnel = restriction des polyèdres.

compromis entre coût et précision

De nombreux domaines ont été proposés.

Domaines basés sur une clôture

- **Octogones** : $\pm X \pm Y \leq c$
  extension des zones par symétrie x / $-x$
  basée sur une adaptation légère de l'algorithme de Floyd-Warshall
- **Deux variables par inégalité** : $\alpha x + \beta y \leq c$
  algorithme de clôture de Nelson
- **Octaèdres** : $\sum \alpha_i V_i \leq c$, $\alpha_i \in \{-1, 0, 1\}$
  propagation incomplète, pour éviter un coût exponentiel
- **Pentagones** : $X - Y \leq 0$
  propagation incomplète, pour atteindre un coût linéaire

Domaines basés sur la programmation linéaire :

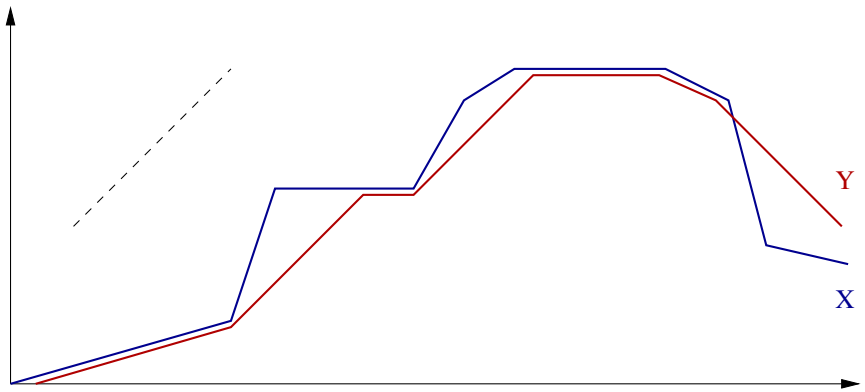
- **Template** : $\mathbf{M} \times \vec{V} \geq \vec{C}$ en fixant la matrice $\mathbf{M}$

Exemple d'analyse avec les octogones

Limiteur de vitesse

$Y \leftarrow 0$; **while** • true **do**
  $X \leftarrow [-128, 128]$; $D \leftarrow [0, 16]$;
  $S \leftarrow Y$; $Y \leftarrow X$; $R \leftarrow X - S$;
  if $R <= -D$ **then** $Y \leftarrow S - D$;
  if $R >= D$ **then** $Y \leftarrow S + D$
done

X :	signal d'entrée
Y :	signal de sortie
S :	dernière entée
R :	delta Y-S
D :	max. permis pour $\|R\|$



Prend un flux X d'entrée et calcule un flux de sortie Y ;
X, et donc Y, sont mis à jour à chaque tour de boucle (*tick* d'horloge).

La sortie Y suit l'entrée X,
mais deux valeurs successives de Y ne doivent pas s'écarter de plus de D.

Exemple d'analyse avec les octogones

Limiteur de vitesse

$Y \leftarrow 0$; **while** • true **do**
  $X \leftarrow [-128, 128]$; $D \leftarrow [0, 16]$;
  $S \leftarrow Y$; $Y \leftarrow X$; $R \leftarrow X - S$;
  if $R <= -D$ **then** $Y \leftarrow S - D$;
  if $R >= D$ **then** $Y \leftarrow S + D$
done

X :	signal d'entrée
Y :	signal de sortie
S :	dernière entée
R :	delta Y-S
D :	max. permis pour $\|R\|$

L'analyse utilise :

- le domaine des octogones ;
- une affectation approchée pour $V_{j_0} \leftarrow a_0 + \sum_k a_k \times V_k$;
- un élargissement avec étages T.

Résultat : nous prouvons que $|Y|$ est borné par : $\min \{ t \in T \mid t \geq 144 \}$

Note :

le domaine des polyèdres trouverait $|Y| \leq 128$, sans avoir besoin d'utiliser d'étage d'élargissement, mais l'analyse serait plus coûteuses.

Domaines numériques : résumé

Résumé

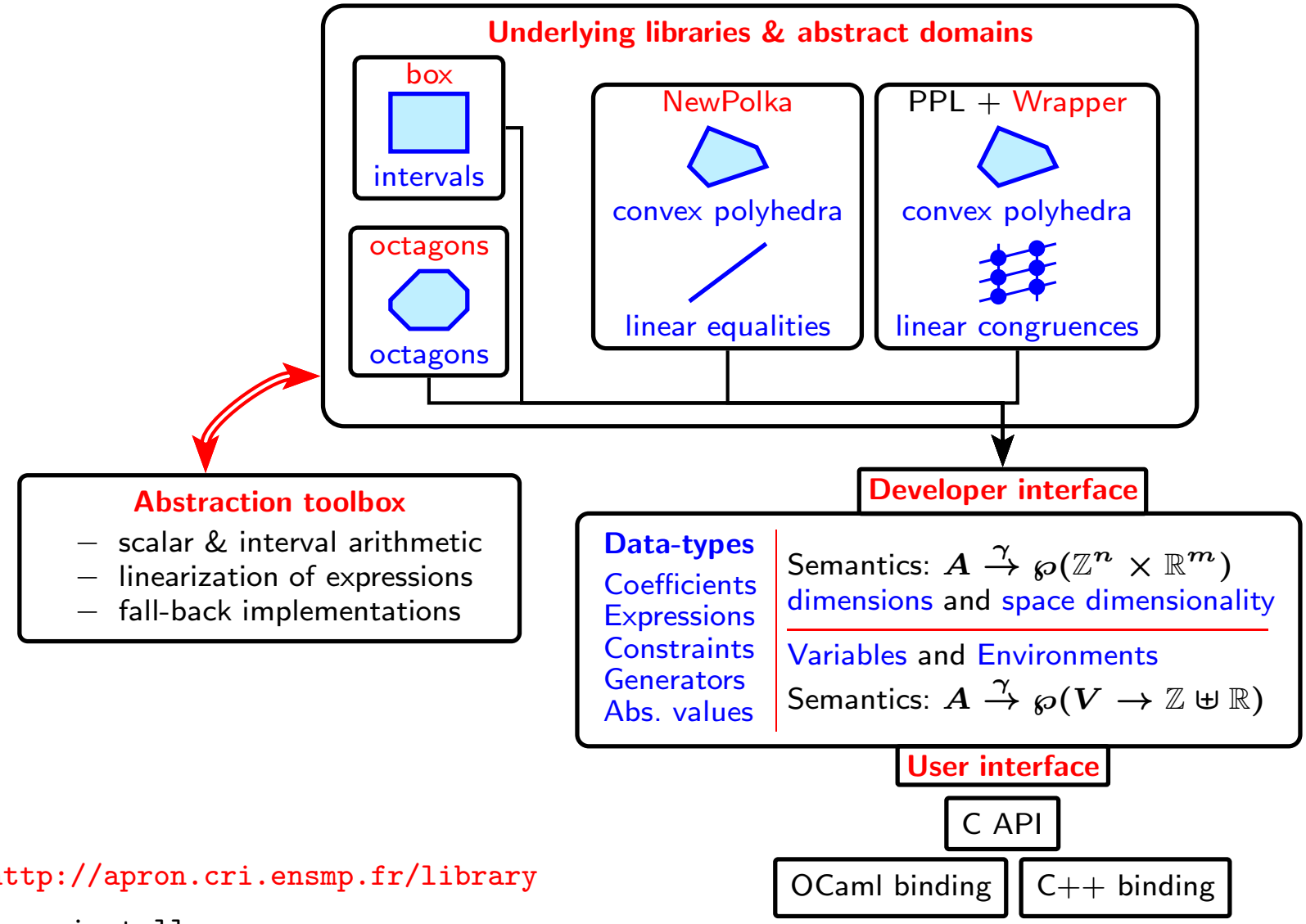
Coût et précision :

domaine	invariants	coût mémoire	coût en temps par opération
intervalles	$V \in [\ell, h]$	$\mathcal{O}(\lvert\mathbb{V}\rvert)$	$\mathcal{O}(\lvert\mathbb{V}\rvert)$
polyèdres	$\sum_i \alpha_i V_i \geq \beta_i$	non borné, exponentiel en pratique	
zones	$V_i - V_j \leq c$	$\mathcal{O}(\lvert\mathbb{V}\rvert^2)$	$\mathcal{O}(\lvert\mathbb{V}\rvert^3)$

- différents domaines offrent différents compromis entre coût et précision, entre coût et expressivité
- des invariants relationnels sont parfois nécessaires
  même pour prouver des propriétés non-relationnelles
- un domaine abstrait est défini par :
  - le choix de propriétés abstraites et d'opérateurs — aspect sémantique
  - des structures de données et des algorithmes — aspect algorithmique
- une analyse mêle deux sortes d'approximations :
  - des approximations statiques — choix des propriétés abstraites
  - des approximations dynamiques — élargissement

Implantation : utilisation de la bibliothèque Apron

Bibliothèque Apron



http://apron.cri.ensmp.fr/library

```
opam install apron
```

Modules de la bibliothèque Apron

Le module `Apron` contient des sous-modules, dont les plus utiles sont :

- `Abstract1`
  éléments abstraits
- `Manager`
  instances de domaine abstrait (passé en argument à toutes les fonctions d'`Abstract1`)
- `Polka`
  fabrique d'objets `Manager.t` pour la création d'éléments abstraits polyèdres
- `Var`
  variable entière ou réelle (dénotée par une chaîne de caractères)
- `Environment`
  ensemble de variables entières et réelles
- `Texpr1`
  arbre d'expression arithmétique
- `Tcons1`
  expression booléenne simple (basée sur `Texpr1`)
- `Coeff`
  coefficient numérique (utilisé dans `Texpr1`, `Tcons1`)

Variables et environnements

Variables : `type Var.t`

les variables sont identifiées par leur nom (chaîne de caractères) :

on suppose donc que les variables du programme ont des noms distincts

- `Var.of_string: string -> Var.t`

Environnements : `type Environment.t`

un élément abstrait représente un ensemble d'environnements $\mathbb{V} \to \mathbb{R}$
$\mathbb{V}$ est l'environnement, contenant des variables à valeur entière et des variables à valeur réelle

- `Environment.make: Var.t array -> Var.t array -> t`
  `make ivars rvars` crée un environnement avec les variables entières `ivars` et les variables réelles `rvars` ;
  `make [||] [||]` est l'environnement vide
- `Environment.add: Environment.t -> Var.t array -> Var.t array -> t`
  `add env ivars rvars` ajoute des variables entières et réelles à `env`
- `Environment.remove: t -> Var.t array -> t`
  `remove env vars` enlève des variables (entières ou réelles) de `env`

en interne, un élément abstrait représente un ensemble de points dans $\mathbb{R}^n$;
l'environnement garde une association entre nom de variable et dimension dans $[1, n]$

Expressions

Arbres concrets d'expression : `type Texpr1.expr`

```
type expr =   | Cst of Coeff.t                                  constantes
              | Var of Var.t                                    variables
              | Unop of unop * expr * typ * round               opérations unaires
              | Binop of binop * expr * expr * typ * round      opérations binaires
```

- opérateurs unaires :

  `type Texpr1.unop = Neg | ···`

- opérateurs binaires :

  `type Texpr1.binop = Add | Sub | Mul | Div | ···`

- types numériques :
  nous utilisons ici uniquement les entiers, mais Apron dispose aussi de types réels et flottants

  `type Texpr1.typ = Int | ···`

- arrondi :
  utile seulement lors de la division d'entiers ; nous utilisons l'arrondi vers zéro, i.e., la troncature

  `type Texpr1.round = Zero | ···`

Expressions (suite)

Forme d'expression interne à la bibliothèque : `type Texpr1.t`

les arbres concrets d'expression doivent être convertis dans une forme interne avant d'être acceptés par les opérations abstraites

- `Texpr1.of_expr: Environment.t -> Texpr1.expr -> Texpr1.t`
  l'environnement est nécessaire pour convertir les noms de variables en dimensions dans $\mathbb{R}^n$

Coefficients : `type Coeff.t`

peuvent être des scalaires $\{c\}$ ou des intervalles $[a, b]$

le module `Mpqf` permet la conversion de chaînes en entiers de précision arbitraire, avant de les convertir vers `Coeff.t` :

- pour un scalaire $\{c\}$:
  ```
  Coeff.s_of_mpqf (Mpqf.of_string c)
  ```
- pour un intervalle $[a, b]$:
  ```
  Coeff.i_of_mpqf (Mpqf.of_string a) (Mpqf.of_string b)
  ```

Expressions booléennes, contraintes numériques

Contraintes : `type Tcons1.t`

constructeur pour une contrainte $expr \bowtie 0$:

- `Tcons1.make: Texpr1.t -> TCons1.typ -> Tcons1.t`

  où :

  `type Tcons1.typ =` `SUPEQ` ($\geq$) | `SUP` ($>$) | `EQ` ($=$) | `DISEQ` ($\neq$) | $\cdots$

Note : évitez d'utiliser `DISEQ`, qui est peu précis ;
utilisez à la place une disjonction de deux contraintes `SUP`

Tableau de contraintes : `type Tcons1.earray`

les opérateurs abstraits n'utilisent pas des contraintes,
mais plutôt des tableaux de contraintes (pour être plus efficace)

Exemple : construction d'un tableau `ar` réduit à une seule contrainte :

```
let c = Tcons1.make texpr1 typ in
let ar = Tcons1.array_make env 1 in
Tcons1.array_set ar 0 c
```

Opérateurs abstraits

Éléments abstraits : `type Abstract1.t`

- `Abstract1.top: Manager.t -> Environment.t -> t`
  crée un élément abstrait où les variables sont non initialisées
  (elles ont toutes les valeurs possibles)

- `Abstract1.env: t -> Environment.t`
  retrouve l'environnement (ensemble de variables) associé à un élément abstrait

- `Abstract1.change_environment: Manager.t -> t -> Environment.t -> bool -> t`
  modifie l'environnement d'un élément abstrait, en ajoutant ou en enlevant des variables si nécessaire ; le paramètre `bool` doit être mis à `false` pour préciser que les variables éventuellement ajoutées ne sont pas initialisées

- `Abstract1.assign_texpr: Manager.t -> t -> Var.t -> Texpr1.t -> t option -> t`
  affectation abstraite : l'argument `option` doit être mis à `None`

- `Abstract1.forget_array: Manager.t -> t -> Var.t array -> bool -> t`
  affectation non-déterministe : oublie la valeur des variables indiquées (si `bool` est `false`)

- `Abstract1.meet_tcons_array: Manager.t -> t -> Tcons1.earray -> t`
  test abstrait : ajoute une ou plusieurs contraintes

Opérateurs abstraits (suite)

- `Abstract1.join: Manager.t -> t -> t -> t`
  union abstraite $\cup^\sharp$
- `Abstract1.meet: Manager.t -> t -> t -> t`
  intersection abstraite $\cap^\sharp$
- `Abstract1.widen: Manager.t -> t -> t -> t`
  élargissement $\triangledown$
- `Abstract1.is_leq: Manager.t -> t -> t -> bool`
  $\subseteq^\sharp$: renvoie `true` si le premier argument est inclus dans le second
- `Abstract1.is_bottom: Manager.t -> t -> t bool`
  renvoie `true` si l'élément abstrait représente $\emptyset$
- `Abstract1.print: Format.formatter -> t -> unit`
  affiche l'élément abstrait

Contrat :

- les opérateurs retournent un nouvel élément abstrait immuable (style fonctionnel)
- les opérateurs peuvent retourner une sur-approximation
  (pas toujours optimal, par exemple pour les expressions non-linéaires)
- les prédicats retournent `true` (propriété satisfaite) ou `false` ("ne sait pas")

Managers

Managers : `type Manager.t`

Un manager représente un choix de domaine abstrait
Pour utiliser les polyèdres, le manager peut être créé par la commande :

- `let manager = Polka.manager_alloc_loose ()`

le même objet `manager` sera ensuite passé en argument à toutes les fonctions de `Abstract1`

pour sélectionner un autre domaine, il suffit de changer la ligne qui définit la variable `manager`

Autres choix possibles :

- `Polka.manager_alloc_equalities` — égalités affines
- `Polka.manager_alloc_strict` — inégalités affines larges ($\geq$) et strictes ($>$)
- `Box.manager_alloc` — intervalles
- `Oct.manager_alloc` — octogones

Erreurs

Compatibilité des arguments : il faut s'assurer que :

- le même manager est utilisé pour créer
  et pour utiliser un élément abstrait

  le système de types vérifie la compatibilité
  entre `'a Manager.t` et `'a Abstract1.t`
- les expressions et les éléments abstraits
  portent sur le même environnement
- les variables destination d'une affectation existent bien
  dans l'environnement de l'élément abstrait
- les deux éléments abstraits arguments d'une opération binaire ($\cup$, $\cap$, $\triangledown$, $\subseteq$)
  sont définis sur le même environnement

sinon, une exception `Manager.Error` sera signalée...

Squelette de domaine abstrait utilisant Apron

```
open Apron

module RelationalDomain = (struct
  (* manager *)
  type man = Polka.loose Polka.t
  let manager = Polka.manager_alloc_loose ()

  (* éléments abstraits *)
  type t = man Abstract1.t

  (* utilitaires *)
  val expr_to_texpr: int_expr -> Texpr1.expr

  (* implantation *)
  ...

end: DOMAIN)
```

Derniers recours dans les affectations et les tests

```
let rec expr_to_texpr = function
| AST_int_binary (op, e1, e2) ->
  match op with
    | AST_PLUS -> Texpr1.Binop ···
    | ···
    | _ -> raise Top

let assign env var expr =
  try
    let e = expr_to_texpr expr in
    Abstract1.assign_texpr ···
  with Top -> Abstract1.forget_array ···

let compare abs e1 e2 =
  try
    ···
    Abstract1.meet_tcons_array ···
  with Top -> abs
```

Principe :

signaler une exception Top pour interrompre le calcul ;
la récupérer pour traiter l'affectation ou le test de manière sûr
en considérant le pire des cas (expression non-déterministe)