Abstraction de sémantiques

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

Antoine Miné

Année 2016–2017

Cours 9
16 février 2017

Plan du cours

But :
construire une analyse statique par abstraction de la sémantique concrète

- notion intuitive d'abstraction
  les signes
- formalisation de la notion d'abstraction
  correspondance de Galois, abstraction sûre, abstraction optimale
  quelques preuves ⋆⋆⋆
- analyse non-relationnelle
  dérivation systématique depuis une abstraction de valeurs
- domaine non-relationnel des constantes
- domaine non-relationnel des intervalles
  gestion précise des tests ⋆⋆⋆
- TME : implantation de l'analyse non-relationnelle en OCaml

Interprétation abstraite

- Cadre unifié pour les sémantiques :
  sémantiques définies comme des points-fixes
  sémantiques définies par induction sur la syntaxe (interprétation)
- Comparer le pouvoir d'expression des sémantiques
  via des fonctions d'abstraction et de concrétisation
  déterminer ce qui peut et ne peut pas être prouvé par une sémantique
- Dériver systématiquement des analyses statiques
  par abstraction d'une sémantique concrète
- Deux aspects à la notion d'abstraction :
  - approximation
    $\{0,2\}$ n'est pas exprimable dans les intervalles, il est approximé par l'intervalle $[0,2]$
  - représentation
    $[0,2]$ n'est pas représenté par un ensemble $\{0,1,2\}$, mais par la paire $(0,2)$
- Assurer la sûreté (soundness)
  toute propriété prouvée dans l'abstrait est vraie dans le concret
  incomplétude : on ne peut pas tout prouver dans l'abstrait
- Assurer la calculabilité effective
  domaines finis (signes) ou accélération de point-fixes (cours 10)

Rappel : sémantique concrète collectrice (1/2)

$\mathsf{E}[\![\, expr \,]\!] : \mathcal{E} \to \mathcal{P}(\mathbb{Z})$ état mémoire → valeurs *possibles* de l'expression *expr*

$$\begin{array}{lcl}
\mathsf{E}[\![\, V \,]\!]\, \rho & \stackrel{\text{def}}{=} & \{\rho(V)\} \\
\mathsf{E}[\![\, c \,]\!]\, \rho & \stackrel{\text{def}}{=} & \{c\} \\
\mathsf{E}[\![\, \mathbf{rand}(a,b) \,]\!]\, \rho & \stackrel{\text{def}}{=} & \{\, x \in \mathbb{Z} \mid a \le x \le b \,\} \\
\mathsf{E}[\![\, e_1 + e_2 \,]\!]\, \rho & \stackrel{\text{def}}{=} & \{\, v_1 + v_2 \mid v_1 \in \mathsf{E}[\![\, e_1 \,]\!]\, \rho,\ v_2 \in \mathsf{E}[\![\, e_2 \,]\!]\, \rho \,\}
\end{array}$$

...

$\mathsf{C}[\![\, cond \,]\!] : \mathcal{P}(\mathcal{E}) \to \mathcal{P}(\mathcal{E})$ états mémoire qui *peuvent* passer le test *cond*

$$\begin{array}{lcl}
\mathsf{C}[\![\, \text{true} \,]\!]\, R & \stackrel{\text{def}}{=} & R \\
\mathsf{C}[\![\, \text{false} \,]\!]\, R & \stackrel{\text{def}}{=} & \emptyset \\
\mathsf{C}[\![\, c_1 \wedge c_2 \,]\!]\, R & \stackrel{\text{def}}{=} & \mathsf{C}[\![\, c_1 \,]\!]\, R \cap \mathsf{C}[\![\, c_2 \,]\!]\, R \\
\mathsf{C}[\![\, c_1 \vee c_2 \,]\!]\, R & \stackrel{\text{def}}{=} & \mathsf{C}[\![\, c_1 \,]\!]\, R \cup \mathsf{C}[\![\, c_2 \,]\!]\, R \\
\mathsf{C}[\![\, e_1 = e_2 \,]\!]\, R & \stackrel{\text{def}}{=} & \{\, \rho \in R \mid \exists v_1 \in \mathsf{E}[\![\, e_1 \,]\!]\, \rho, v_2 \in \mathsf{E}[\![\, e_2 \,]\!]\, \rho \colon v_1 = v_2 \,\} \\
\mathsf{C}[\![\, e_1 < e_2 \,]\!]\, R & \stackrel{\text{def}}{=} & \{\, \rho \in R \mid \exists v_1 \in \mathsf{E}[\![\, e_1 \,]\!]\, \rho, v_2 \in \mathsf{E}[\![\, e_2 \,]\!]\, \rho \colon v_1 < v_2 \,\}
\end{array}$$

...

Environnements : $\mathcal{E} \stackrel{\text{def}}{=} \mathbb{V} \to \mathbb{Z}$

états mémoire, associant à chaque variable dans $\mathbb{V}$ une valeur dans $\mathbb{Z}$

Rappel : sémantique concrète collectrice (2/2)

$\underline{\mathsf{S}[\![\, stat \,]\!] : \mathcal{P}(\mathcal{E}) \to \mathcal{P}(\mathcal{E})}$

$$
\begin{array}{lcl}
\mathsf{S}[\![\, \textbf{skip} \,]\!]\, R & \stackrel{\text{def}}{=} & R \\
\mathsf{S}[\![\, \textbf{halt} \,]\!]\, R & \stackrel{\text{def}}{=} & \emptyset \\
\mathsf{S}[\![\, X \leftarrow e \,]\!]\, R & \stackrel{\text{def}}{=} & \{\, \rho[X \mapsto v] \mid \rho \in R,\ v \in \mathsf{E}[\![\, e \,]\!]\, \rho \,\} \\
\mathsf{S}[\![\, s_1 ; s_2 \,]\!] & \stackrel{\text{def}}{=} & \mathsf{S}[\![\, s_2 \,]\!] \circ \mathsf{S}[\![\, s_1 \,]\!] \\
\mathsf{S}[\![\, \textbf{if}\ c\ \textbf{then}\ s_1\ \textbf{else}\ s_2 \,]\!]\, R & \stackrel{\text{def}}{=} & \mathsf{S}[\![\, s_1 \,]\!]\, (\mathsf{C}[\![\, c \,]\!]\, R) \cup \mathsf{S}[\![\, s_2 \,]\!]\, (\mathsf{C}[\![\, \neg c \,]\!]\, R) \\
\mathsf{S}[\![\, \textbf{assert}\ c \,]\!]\, R & \stackrel{\text{def}}{=} & \mathsf{C}[\![\, c \,]\!]\, R \\
\mathsf{S}[\![\, \textbf{while}\ c\ \textbf{do}\ s \,]\!]\, R & \stackrel{\text{def}}{=} & \mathsf{C}[\![\, \neg c \,]\!]\, (\mathrm{lfp}\, \lambda X. R \cup \mathsf{S}[\![\, s \,]\!]\, (\mathsf{C}[\![\, c \,]\!]\, X))
\end{array}
$$

états mémoire en entrée de *stat* $\to$ états mémoire *possibles* en sortie de *stat*

Limite à l'automatisation

La sémantique concrète n'est pas calculable car :

1. les éléments du domaine concret $\mathcal{P}(\mathcal{E})$
   ne sont pas tous représentables en mémoire ;

2. le point fixe $\mathrm{lfp}\, \lambda X.R \cup \mathsf{S}[\![\, s \,]\!]\, (\mathsf{C}[\![\, c \,]\!]\, X)$
   peut faire intervenir un nombre infini d'itérations
   théorème de Kleene

L'interprétation abstraite fournit une solution à ces deux problèmes :

1. remplacer $\mathcal{P}(\mathcal{E})$ par un domaine abstrait
   dont les éléments sont représentables en mémoire
   $\Longrightarrow$ ce cours

2. calculer les itérations de point fixe avec accélération de convergence
   $\Longrightarrow$ prochain cours

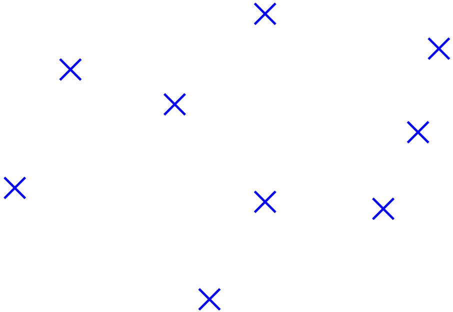
Analyse abstraite, analyse approchée

Plutôt que de raisonner sur le comportement réel des programmes nous raisonnons à un niveau d'abstraction.

Exemple : l'abstraction des signes

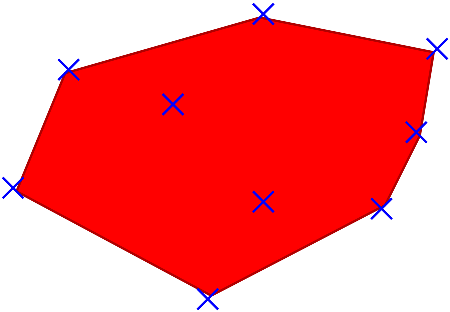
- oublier la valeur exacte des variables
  pour ne se souvenir que de leur signe : $+$, $-$, 0, $\top$ (aucune information)
  $\Longrightarrow$ 2 devient $+$
- la sémantique de valeur est plus expressive, c'est le monde concret,
  la sémantique de signe est moins expressive, c'est le monde abstrait ;
- dans le concret : $2 - 1 = 1$, qui est positif
  mais, après abstraction, cela donne $(+) - (+) = \top$
  $\Longrightarrow$ perte de précision
- si l'analyseur indique $X = +$ en fin d'analyse,
  alors nous ne connaissons pas l'ensemble des valeurs possibles de X,
  mais nous savons que X sera forcément positif.

Nous construirons donc des abstractions calculables du programme.



domaine concret $\mathcal{D}$: $\{(0,3),(6,0),(12,7),\ldots\}$

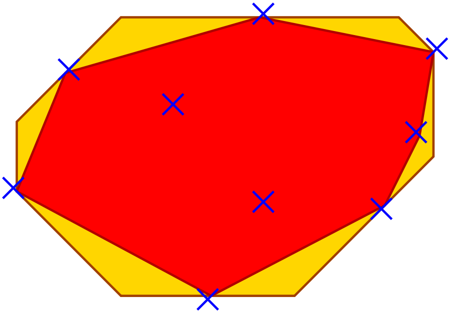
Exemples de domaines abstraits numériques



domaine concret $\mathcal{D}$: $\{(0,3),(6,0),(12,7),\ldots\}$

abstraction des polyèdres $\mathcal{D}^\sharp_p$: $6X + 11Y \geq 33 \wedge \cdots$

Exemples de domaines abstraits numériques

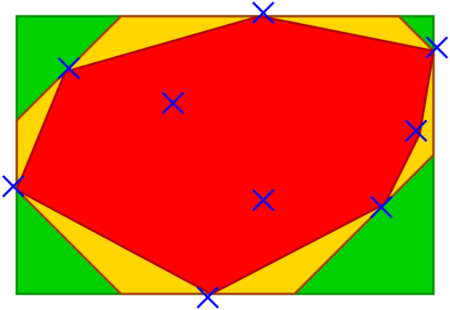


domaine concret $\mathcal{D}$: $\{(0,3),(6,0),(12,7),\ldots\}$

abstraction des polyèdres $\mathcal{D}_p^\sharp$: $6X + 11Y \geq 33 \wedge \cdots$

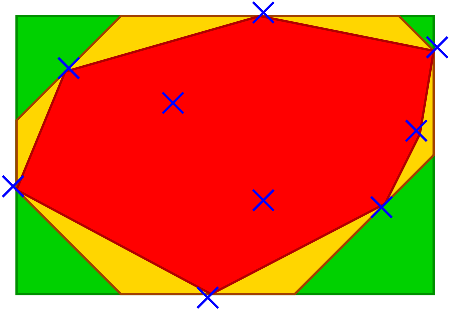
abstraction des octogones $\mathcal{D}_o^\sharp$: $X + Y \geq 3 \wedge Y \geq 0 \wedge \cdots$

Exemples de domaines abstraits numériques



domaine concret $\mathcal{D}$: $\{(0,3),(6,0),(12,7),\ldots\}$
abstraction des polyèdres $\mathcal{D}^\sharp_p$: $6X + 11Y \geq 33 \wedge \cdots$
abstraction des octogones $\mathcal{D}^\sharp_o$: $X + Y \geq 3 \wedge Y \geq 0 \wedge \cdots$
abstraction des intervalles $\mathcal{D}^\sharp_i$: $X \in [0,12] \wedge Y \in [0,8]$

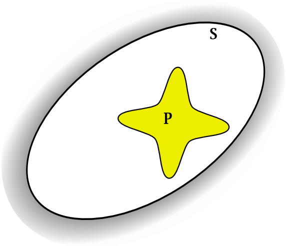
Exemples de domaines abstraits numériques



domaine concret $\mathcal{D}$:	$\{(0,3),(6,0),(12,7),\ldots\}$	non calculable
abstraction des polyèdres $\mathcal{D}^\sharp_p$:	$6X + 11Y \geq 33 \wedge \cdots$	coût exponentiel
abstraction des octogones $\mathcal{D}^\sharp_o$:	$X + Y \geq 3 \wedge Y \geq 0 \wedge \cdots$	coût cubique
abstraction des intervalles $\mathcal{D}^\sharp_i$:	$X \in [0,12] \wedge Y \in [0,8]$	coût linéaire

Compromis entre coût et expressivité / précision !

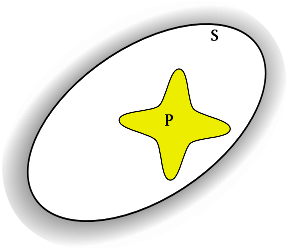
Sûreté et fausses alarmes



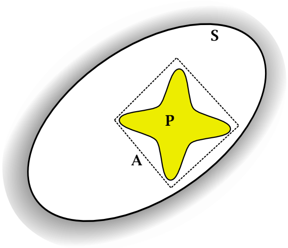
$P \subseteq S$

<u>But :</u> prouver qu'un programme P satisfait sa spécification S

Sûreté et fausses alarmes



$\Longleftarrow$



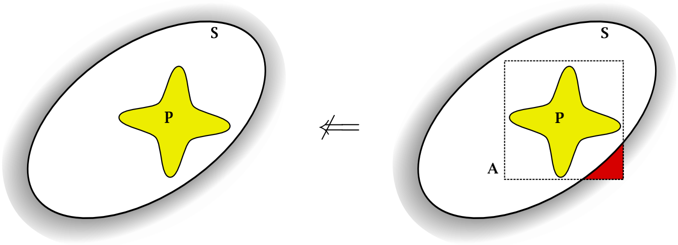
$P \subseteq S$

$A \subseteq S$
programme prouvé

<u>But :</u> prouver qu'un programme P satisfait sa spécification S

Une abstraction polyédrique A peut prouver la correction.

Sûreté et fausses alarmes



$P \subseteq S$

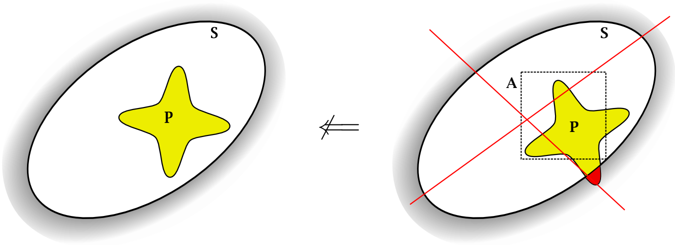
$A \not\subseteq S$
fausse alarme

<u>But :</u> prouver qu'un programme P satisfait sa spécification S

Une abstraction polyédrique A peut prouver la correction.

Une abstraction d'intervalle ne peut pas prouver la correction
$\Longrightarrow$ fausse alarme.

Sûreté et fausses alarmes



$P \subseteq S$

impossible
fausse preuve

<u>But :</u> prouver qu'un programme P satisfait sa spécification S

Une abstraction polyédrique A peut prouver la correction.

Une abstraction d'intervalle ne peut pas prouver la correction
$\Longrightarrow$ fausse alarme.

L'analyse est pas sûre $\Longrightarrow$ jamais de faux négatif !

Intuition : les signes, ensembles et propriétés

Comment approximer efficacement des ensembles d'entiers $\mathcal{P}(\mathbb{Z})$?

Exemple : garder uniquement l'information de signe.

Raisonnement dans $\mathcal{D}^\sharp \stackrel{\text{def}}{=} \{0, \geq 0, \leq 0, \top, \bot\}$ où

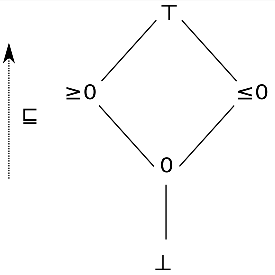
- ≥ 0 dénote un ensemble d'entiers positifs ;
- ≤ 0 dénote un ensemble d'entiers négatifs ;
- 0 dénote un ensemble d'entiers nuls ;
- $\top$ dénote un ensemble d'entiers arbitraires ;
- $\bot$ indique un ensemble vide.

Un élément de $\mathcal{D}^\sharp$ dénote une propriété d'un ensemble d'entiers ;

- un ensemble peut avoir plusieurs propriétés :
  $\{1, 2, 3\}$ a la propriété ≥ 0, mais aussi la propriété $\top$;
- une propriété peut aussi être vue comme un ensemble d'entiers :
  0 est $\{0\}$, ≥ 0 est $\{0, 1, 2, \ldots\}$.
  l'ensemble des entiers qui satisfont la propriété

Quizz : quelles sont *les* propriétés de $\{-1, 0, 1\}$? de 0 ? de $\emptyset$?

Intuition : ordre d'information, meilleur abstraction



Intuitivement, les propriétés peuvent être ordonnées par un ordre d'information $\sqsubseteq$: $\bot \sqsubseteq 0 \sqsubseteq \geq 0 \sqsubseteq \top$.

C'est un ordre partiel : ≤ 0 et ≥ 0 ne sont pas comparables.

Pour les signes, nous avons même une structure de treillis complet.

L'ordre est compatible avec l'interprétation ensembliste,
en effet : $\emptyset \subseteq \{0\} \subseteq \mathbb{N} \subseteq \mathbb{Z}$.

Tout ensemble d'entiers a une meilleur représentation sous forme d'information de signe (i.e., plus petite pour $\sqsubseteq$)

0 est une meilleur représentation que ≥ 0 ou $\top$ pour $\{0\}$
elle donne plus d'information, elle représente un ensemble plus petit d'entiers

Intuition : opérateurs abstraits

Exemple : la règle des signes

$\times$	≥ 0	≤ 0	0	$\top$	$\bot$
≥ 0	≥ 0	≤ 0	0	$\top$	$\bot$
≤ 0	≤ 0	≥ 0	0	$\top$	$\bot$
0	0	0	0	0	$\bot$
$\top$	$\top$	$\top$	0	$\top$	$\bot$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$

$+$	≥ 0	≤ 0	0	$\top$	$\bot$
≥ 0	≥ 0	$\top$	≥ 0	$\top$	$\bot$
≤ 0	$\top$	≤ 0	≤ 0	$\top$	$\bot$
0	≥ 0	≤ 0	0	$\top$	$\bot$
$\top$	$\top$	$\top$	$\top$	$\top$	$\bot$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$

Principe :

Remplacer un calcul d'opérateur $\circ$ sur $\mathbb{Z}$ par un calcul abstrait $\circ^\sharp$ sur $\mathcal{D}^\sharp$.

Raisonnement dans les signes $\mathcal{D}^\sharp$ compatible avec celui dans $\mathcal{P}(\mathbb{Z})$:

- si $a, b \in \mathbb{Z}$ ont la propriété $a^\sharp, b^\sharp \in \mathcal{D}^\sharp$
  alors $a \circ b$ a bien la propriété $a^\sharp \circ^\sharp b^\sharp$
  $\Longrightarrow$ sûreté
- parfois $\circ^\sharp$ donne la plus forte propriété dans $\mathcal{D}^\sharp$
  $\Longrightarrow$ optimalité

La sûreté sera toujours garantie ;
l'optimalité ne sera pas toujours garantie.

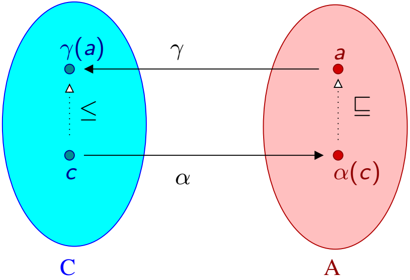
Formalisation de la correspondance concret/abstrait

Correspondance de Galois

Étant donnés deux posets $(C, \leq)$ et $(A, \sqsubseteq)$,
la paire $(\alpha : C \to A, \gamma : A \to C)$ est une correspondance de Galois si :

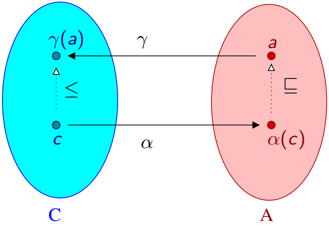
$$\forall a \in A, c \in C, \alpha(c) \sqsubseteq a \iff c \leq \gamma(a)$$

ce que nous notons : $(C, \leq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (A, \sqsubseteq)$.



- α est l'adjoint supérieur, ou abstraction ; A est le domaine abstrait.
- γ est l'adjoint inférieur, ou concrétisation ; C est le domaine concret.

Exemple : correspondance de Galois pour les signes



Les signes représentent des ensembles d'entiers :

- $C \stackrel{\text{def}}{=} \mathcal{P}(\mathbb{Z})$;
- $A \stackrel{\text{def}}{=} \{\bot, 0, \geq 0, \leq 0, \top\}$;
- l'ordre concret $\leq$ est $\subseteq$.

$$\begin{array}{lcl} \gamma(\bot) & \stackrel{\text{def}}{=} & \emptyset \\ \gamma(0) & \stackrel{\text{def}}{=} & \{0\} \\ \gamma(\geq 0) & \stackrel{\text{def}}{=} & \mathbb{N} \\ \gamma(\leq 0) & \stackrel{\text{def}}{=} & -\mathbb{N} \\ \gamma(\top) & \stackrel{\text{def}}{=} & \mathbb{Z} \end{array}$$

$$\alpha(S) \stackrel{\text{def}}{=} \begin{cases} \bot & \text{si } S = \emptyset \\ 0 & \text{si } S = \{0\} \\ \geq 0 & \text{sinon, si } \forall s \in S,\ s \geq 0 \\ \leq 0 & \text{sinon, si } \forall s \in S,\ s \leq 0 \\ \top & \text{sinon} \end{cases}$$

À vérifier : $\alpha(c) \sqsubseteq a \iff c \subseteq \gamma(a)\ldots$

Caractérisation alternative

Une paire $(\alpha : C \to A, \gamma : A \to C)$ est une correspondance de Galois si et seulement si elle satisfait :

1. γ est croissante
   $\forall a, a',\ a \sqsubseteq a' \implies \gamma(a) \leq \gamma(a')$
2. α est croissante
   $\forall c, c',\ c \leq c' \implies \alpha(c) \sqsubseteq \alpha(c')$
3. $\gamma \circ \alpha$ est extensive
   $\forall c,\ c \leq \gamma(\alpha(c))$
4. $\alpha \circ \gamma$ est réductrice
   $\forall a,\ \alpha(\gamma(a)) \sqsubseteq a$

C'est à dire :

- les ordres abstraits et concrets sont compatibles (croissance)
- passer par l'abstrait est une sur-approximation vis à vis du concret
  ($\gamma \circ \alpha$ est extensive)

Note : généralement, $\alpha \circ \gamma$ est l'identité... voir un peu plus loin.

Quelques preuves sur les correspondances de Galois ⋆⋆⋆

Si $\forall a, c,\ \alpha(c) \sqsubseteq a \iff c \leq \gamma(a)$, alors :

1. $\gamma \circ \alpha$ est extensive : $\forall c,\ c \leq \gamma(\alpha(c))$
   ⋆⋆⋆preuve : $\alpha(c) \sqsubseteq \alpha(c) \implies c \leq \gamma(\alpha(c))$
2. $\alpha \circ \gamma$ est réductrice : $\forall a,\ \alpha(\gamma(a)) \sqsubseteq a$
3. α est croissante
   ⋆⋆⋆preuve : $c \leq c' \implies c \leq \gamma(\alpha(c')) \implies \alpha(c) \sqsubseteq \alpha(c')$
4. γ est croissante
5. $\gamma \circ \alpha \circ \gamma = \gamma$
   ⋆⋆⋆preuve :
   $\alpha(\gamma(a)) \sqsubseteq \alpha(\gamma(a)) \implies \gamma(a) \leq \gamma(\alpha(\gamma(a)))$ et
   $a \sqsupseteq \alpha(\gamma(a)) \implies \gamma(a) \geq \gamma(\alpha(\gamma(a)))$
6. $\alpha \circ \gamma \circ \alpha = \alpha$
7. $\alpha \circ \gamma$ est idempotente : $\alpha \circ \gamma \circ \alpha \circ \gamma = \alpha \circ \gamma$
8. $\gamma \circ \alpha$ est idempotente : $\gamma \circ \alpha \circ \gamma \circ \alpha = \gamma \circ \alpha$

Meilleur abstraction, unicité des adjoints

Si $(C, \leq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (A, \sqsubseteq)$
alors chaque adjoint peut être défini de manière unique
en fonction de l'autre :

1. $\alpha(c) = \sqcap \{ a \mid c \leq \gamma(a) \}$
2. $\gamma(a) = \vee \{ c \mid \alpha(c) \sqsubseteq a \}$

Conséquence importante

$\alpha(c) = \sqcap \{ a \mid c \leq \gamma(a) \}$ signifie :
$\alpha(c)$ est la meilleur abstraction de c dans A
la sur-approximation la plus précise

Preuve : de 1

$\forall a,\ c \leq \gamma(a) \implies \alpha(c) \sqsubseteq a$, donc $\alpha(c)$ est un minorant de $\{ a \mid c \leq \gamma(a) \}$.
Supposons que a' est un autre minorant.
Alors, $\forall a,\ c \leq \gamma(a) \implies a' \sqsubseteq a$.
Par correspondance de Galois, il vient $\forall a,\ \alpha(c) \sqsubseteq a \implies a' \sqsubseteq a$.
Ceci implique : $a' \sqsubseteq \alpha(c)$.
Donc, le plus grand minorant de $\{ a \mid c \leq \gamma(a) \}$ existe, et vaut $\alpha(c)$.
La preuve de 2 est similaire.

Propriétés additionnelles des correspondances de Galois

Si $(\alpha : C \to A, \gamma : A \to C)$ est une correspondance de Galois, alors :

1. $\forall X \subseteq C$, si $\vee X$ existe, alors $\alpha(\vee X) = \sqcup \{ \alpha(x) \mid x \in X \}$.
2. $\forall X \subseteq A$, si $\sqcap X$ existe, alors $\gamma(\sqcap X) = \wedge \{ \gamma(x) \mid x \in X \}$.

Conséquence importante

Le domaine abstrait est clos par conjonction
la conjonction de deux propriétés expressibles dans l'abstrait est expressible dans l'abstrait

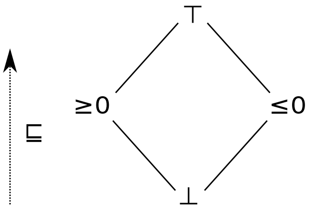
Preuve : de 1

Par définition des lubs, $\forall x \in X$, $x \leq \vee X$.
Par croissance, $\forall x \in X$, $\alpha(x) \sqsubseteq \alpha(\vee X)$.
Or, $\alpha(\vee X)$ est un majorant de $\{ \alpha(x) \mid x \in X \}$.
Supposons que y soit un autre majorant de $\{ \alpha(x) \mid x \in X \}$.
Alors, $\forall x \in X$, $\alpha(x) \sqsubseteq y$.
Par correspondance de Galois, $\forall x \in X$, $x \leq \gamma(y)$.
Par définition des lubs, $\vee X \leq \gamma(y)$.
Par correspondance de Galois, $\alpha(\vee X) \sqsubseteq y$.
Donc $\{ \alpha(x) \mid x \in X \}$ a un lub, égal à $\alpha(\vee X)$.
La preuve de 2 est similaire.

Optimalité et clôture par conjonction de propriétés

Nous avons : $\gamma(a \sqcap a') = \gamma(a) \wedge \gamma(a')$

Contre-exemple : un domaine des signes imparfait



$C \stackrel{\text{def}}{=} \mathcal{P}(\mathbb{Z})$

dans C la conjonction de propriétés
est l'intersection d'ensembles d'entiers

$A \stackrel{\text{def}}{=} \{\bot, \leq 0, \geq 0, \top\}$

$\gamma(\leq 0) \cap \gamma(\geq 0) = \{0\} \notin \gamma(A)$

pas de meilleur abstraction pour $\{0\}$

$\implies$ pas de correspondance de Galois

Corrections possibles :

- compléter A par $\cap$: $A \stackrel{\text{def}}{=} \{\bot, 0, \leq 0, \geq 0, \top\}$;
- vider A, en enlevant des éléments : $A \stackrel{\text{def}}{=} \{\bot, \geq 0, \top\}$;
- modifier des éléments : $A \stackrel{\text{def}}{=} \{\bot, < 0, \geq 0, \top\}$.

Opérations concrètes sur les ensembles entiers

Rappel : le monde concret est $C \stackrel{\text{def}}{=} \mathcal{P}(\mathbb{Z})$.

But : définir les opérations sémantiques élémentaires sur C

Briques de base pour la définition de la sémantique concrète $\mathsf{E}[\![\ ]\!]$, $\mathsf{C}[\![\ ]\!]$, $\mathsf{S}[\![\ ]\!]$.

- opérations arithmétiques :
  $+$, $-$, $\times$, $/$, étendues aux ensembles $(\mathcal{P}(\mathbb{Z}))^n \to \mathcal{P}(\mathbb{Z})$

$$
\begin{array}{lcl}
\overline{-}\, X & \stackrel{\text{def}}{=} & \{\, -x \mid x \in X \,\} \\
X \mathbin{\overline{+}} Y & \stackrel{\text{def}}{=} & \{\, x + y \mid x \in X, y \in Y \,\} \\
X \mathbin{\overline{-}} Y & \stackrel{\text{def}}{=} & \{\, x - y \mid x \in X, y \in Y \,\} \\
X \mathbin{\overline{\times}} Y & \stackrel{\text{def}}{=} & \{\, x \times y \mid x \in X, y \in Y \,\} \\
X \mathbin{\overline{/}} Y & \stackrel{\text{def}}{=} & \{\, x / y \mid x \in X, y \in Y, y \neq 0 \,\}
\end{array}
$$

- opérations ensemblistes : $\cup$, $\cap$
- relation d'ordre : $\subseteq$
- filtres : $\leq$, $\geq$, ... présentés plus tard

Abstraction d'opérateurs : abstraction sûre

Étant donnée une correspondance de Galois $(C, \leq) \xleftarrow[\alpha]{\gamma} (A, \sqsubseteq)$
et un opérateur concret $F : C \to C$
comment modéliser F dans l'abstrait ?

Sûreté :

$a \in A$ est une abstraction sûre de $c \in C$ si $c \leq \gamma(a)$
ou, de manière équivalente : $\alpha(c) \sqsubseteq a$.

Donc, $F^\sharp : A \to A$ est une abstraction sûre de $F : C \to C$
si $\forall a$: $F(\gamma(a)) \leq \gamma(F^\sharp(a))$
ou, de manière équivalente : $\alpha(F(\gamma(a))) \sqsubseteq F^\sharp(a)$.

Nous le notons en raccourci : $F \circ \gamma \leq \gamma \circ F^\sharp$
un pas dans l'abstrait sur-approxime un pas dans le concret

Se généralise aux opérateurs $n-$aires : $F(\gamma(a_1), \ldots, \gamma(a_n)) \leq \gamma(F^\sharp(a_1, \ldots, a_n))$

Note : pour définir la sûreté, il nous suffit de la concrétisation γ
l'existence d'une abstraction α, donc d'une correspondance de Galois, est superflue

Abstraction d'opérateurs : abstraction optimale

Optimalité :

$F^\sharp$ définie par $F^\sharp \stackrel{\text{def}}{=} \alpha \circ F \circ \gamma$ est **optimale**.

En effet, $F^\sharp$ est sûre $\iff (\alpha \circ F \circ \gamma)(a) \sqsubseteq F^\sharp(a)$
donc $\alpha \circ F \circ \gamma$ est l'abstraction sûre de F la plus précise !

Conséquences :

- la sémantique abstraite $F^\sharp$ peut être dérivée systématiquement de la sémantique concrète F, étant donnée une correspondance de Galois (α, γ) ;
- mais $\alpha \circ F \circ \gamma$ n'est qu'une définition mathématique, encore faut-il trouver un algorithme effectif pour l'implanter...

Exemples :

domaine des signes $\mathcal{D}^\sharp \stackrel{\text{def}}{=} \{\bot, 0, \leq 0, \geq 0, \top\}$

- dans le concret $X \mathrel{\overline{\times}} Y \stackrel{\text{def}}{=} \{\, a \times b \mid a \in X, b \in Y \,\}$
- $\times^\sharp \stackrel{\text{def}}{=} \alpha(\gamma(X^\sharp) \mathrel{\overline{\times}} \gamma(Y^\sharp))$ redonne la règle des signes
  $(\geq 0) \times^\sharp (\geq 0) = (\geq 0)$, $(\geq 0) \times^\sharp (\leq 0) = (\leq 0)$, $(\geq 0) \times^\sharp 0 = 0$, ...
- $X^\sharp \mathbin{/^\sharp} Y^\sharp = \top$ est sûr, mais n'est pas optimal !

Abstraction d'opérateurs : abstraction exacte

Exactitude :

$F^\sharp$ est une abstraction exacte de F si $F \circ \gamma = \gamma \circ F^\sharp$
$\Longrightarrow$ aucune perte à effectuer l'opération dans l'abstrait

$F^\sharp$ exacte implique $F^\sharp$ sûre et optimale
mais les fonctions optimales ne sont pas toujours exactes !
$\Longrightarrow$ l'abstraction α génère une perte d'information.

Exemples : domaine des signes alternatif $\mathcal{D}^\sharp \stackrel{\text{def}}{=} \{\bot, 0, <0, >0, \top\}$

- la règle des signes pour $\times^\sharp$ reste sûre, optimale, et exacte ;
- l'union abstraite $X^\sharp \cup^\sharp Y^\sharp \stackrel{\text{def}}{=} \alpha(\gamma(X^\sharp) \cup \gamma(Y^\sharp))$ est optimale
  mais pas exacte : $(>0) \cup^\sharp (<0) = \top$
  or $\gamma(>0) \cup \gamma(<0) = \mathbb{Z} \setminus \{0\} \neq \mathbb{Z}$!

Dans la plus part des domaines, $\cup^\sharp$ n'est pas exact...

Quizz : comment définir $\cap^\sharp$ optimale ? est-elle exacte ?

Abstraction d'opérateurs : composition

- la composition d'abstractions sûres est une abstraction sûre :

  si F est croissante
  et $F^\sharp$, $G^\sharp$ abstraient F et G de manière sûre
  alors $F^\sharp \circ G^\sharp$ est une abstraction sûre de $F \circ G$
  Preuve : $\forall a,\ (F \circ G \circ \gamma)(a) \leq (F \circ \gamma \circ G^\sharp)(a) \leq (\gamma \circ F^\sharp \circ G^\sharp)(a)$

- la composition d'abstractions exactes est une abstraction exacte :

  si $F \circ \gamma = \gamma \circ F^\sharp$ et $G \circ \gamma = \gamma \circ G^\sharp$,
  alors $(F \circ G) \circ \gamma = \gamma \circ (F^\sharp \circ G^\sharp)$.

Principe

- réduire la sémantique concrète du langage en une composition d'un petit nombre d'opérations élémentaires
- abstraire chaque opération élémentaire
- composer les opérations abstraites pour obtenir une analyse

Abstraction d'opérateurs : composition (suite)

- la composition d'abstractions optimales est sûre
  mais pas forcément optimale
  $(\alpha \circ F \circ \gamma) \circ (\alpha \circ G \circ \gamma)$ n'est pas $\alpha \circ (F \circ G) \circ \gamma$
  (le $\gamma \circ \alpha$ peut générer une perte de précision)

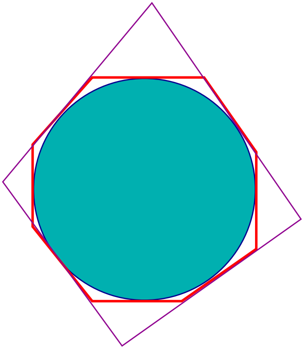
Exemple :

- dans $\mathcal{P}(\mathbb{Z})$,
  prenons $F(X) \stackrel{\text{def}}{=} \{\, x+1 \mid x \in X \,\}$ et $G(X) \stackrel{\text{def}}{=} \{\, x-1 \mid x \in X \,\}$
  on a donc $(G \circ F)(\{0\}) = \{0\}$;
- dans les signes :
  $F^\sharp(0) = (\alpha \circ F \circ \gamma)(0) = (\geq 0)$ et $G^\sharp(\geq 0) = (\alpha \circ G \circ \gamma)(\geq 0) = \top$
  donc $(G^\sharp \circ F^\sharp)(0) = \top$;
- pourtant $(\alpha \circ (G \circ F) \circ \gamma)(0) = \alpha(\{0\}) = 0$!

Conclusion

La granularité des opérations élémentaire compte.
Une décomposition trop fine cause une perte de précision !

Absence de correspondance de Galois



Exemple : le domaine des polyèdres.

Il n'existe pas de meilleurs sur-approximation d'un cercle par un polygone.

nous pouvons toujours raffiner le polygone
en ajoutant des arrêtes

L'emploi d'un opérateur optimal $\alpha \circ F \circ \gamma$ n'est pas toujours possible :

- certains domaines abstraits n'ont pas de correspondance de Galois ;
- $\alpha \circ F \circ \gamma$ peut être difficile ou coûteux à implanter ;

$\Longrightarrow$ nous nous contentons alors d'abstractions sûres, non optimales.

En pratique, l'analyse statique par interprétation abstraite avec seulement γ est fréquente !

Analyse non-relationnelle

Abstraction des environnements

- nous avons vu des abstractions de $\mathcal{P}(\mathbb{Z})$;
- mais notre sémantique concrète $\mathsf{S}[\![\, stat \,]\!]$ manipule des ensembles d'environnements $\mathcal{P}(\mathcal{E}) = \mathcal{P}(\mathbb{V} \to \mathbb{Z})$.

Principe de l'analyse non-relationnelle :

- associer une valeur abstraite à chaque variable ;
  (exemple : X est positif et Y est négatif)
- $\mathcal{P}(\mathbb{V} \to \mathbb{Z})$ est abstrait par $\mathcal{E}^\sharp \stackrel{\text{def}}{=} \mathbb{V} \to \mathcal{D}^\sharp$
  où $\mathcal{D}^\sharp$ est une abstraction arbitraire de $\mathcal{P}(\mathbb{Z})$
  (exemple : $\mathcal{D}^\sharp$ est le domaine des signes)

Les opérations sur $\mathcal{E}^\sharp$ seront systématiquement dérivées des opérations sur $\mathcal{D}^\sharp$.

Signature de l'abstraction des valeurs

Abstraction de valeurs $\mathcal{D}^\sharp$:

$\mathcal{D}^\sharp$	ensemble de valeurs abstraites représentables en mémoire
$\gamma : \mathcal{D}^\sharp \to \mathcal{P}(\mathbb{Z})$	concretisation
$\alpha : \mathcal{P}(\mathbb{Z}) \to \mathcal{D}^\sharp$	abstraction (optionnelle)
$\sqsubseteq$	ordre partiel compatible avec γ (γ croissant)
$\bot$, $\top$	représentation de $\emptyset$ et $\mathbb{Z}$
$+^\sharp$, $-^\sharp$, $\times^\sharp$, $/^\sharp$	abstractions sûres de $\overline{+}$, $\overline{-}$, $\overline{\times}$, $\overline{/}$
$c^\sharp$, $[a, b]^\sharp$	abstractions sûres de $\{c\}$, $[a, b]$
$\cup^\sharp$, $\cap^\sharp$	abstractions sûres de $\cup$ et $\cap$

Dérivation systématique de $\mathcal{D}^\sharp$

$\mathcal{E}^\sharp$ est défini par :

- $\mathcal{E}^\sharp \stackrel{\text{def}}{=} \mathbb{V} \to \mathcal{D}^\sharp$
- ordre point à point : $X_1^\sharp \mathrel{\dot\sqsubseteq} X_2^\sharp \iff \forall V \in \mathbb{V}: X_1^\sharp(V) \sqsubseteq X_2^\sharp(V)$
- union : $X_1^\sharp \mathrel{\dot\cup^\sharp} X_2^\sharp \stackrel{\text{def}}{=} \lambda V. X_1^\sharp(V) \cup^\sharp X_2^\sharp(V)$
- intersection : $X_1^\sharp \mathrel{\dot\cap^\sharp} X_2^\sharp \stackrel{\text{def}}{=} \lambda V. X_1^\sharp(V) \cap^\sharp X_2^\sharp(V)$
- $\dot\bot(V) \stackrel{\text{def}}{=} \lambda V.\bot$
- $\dot\top(V) \stackrel{\text{def}}{=} \lambda V.\top$

Note : la structure de $\mathcal{D}^\sharp$ se retrouve sur $\mathcal{E}^\sharp$

- $(\mathcal{E}^\sharp, \dot\sqsubseteq)$ est un ordre partiel
- si $\mathcal{D}^\sharp$ est un treillis (complet), alors $\mathcal{E}^\sharp$ est aussi un treillis complet
- si $\mathcal{D}^\sharp$ a une correspondance de Galois
  on peut définir une correspondance de Galois sur $\mathcal{E}^\sharp$:
  - $(\mathcal{P}(\mathbb{V} \to \mathbb{Z}), \subseteq) \underset{\dot\alpha}{\overset{\dot\gamma}{\leftrightarrows}} (\mathbb{V} \to \mathcal{D}^\sharp, \dot\sqsubseteq)$
  - $\dot\alpha(E) \stackrel{\text{def}}{=} \lambda V. \alpha(\{\, \rho(V) \mid \rho \in R \,\})$
  - $\dot\gamma(X^\sharp) \stackrel{\text{def}}{=} \{\, \rho \mid \forall V \in \mathbb{V}: \rho(V) \in \gamma(X^\sharp(V)) \,\}$

Notation : un point $\dot{}$ sert à distinguer les opérations sur $\mathcal{D}^\sharp$ de celles sur $\mathbb{V} \to \mathcal{D}^\sharp$.

Sémantique abstraite des expressions

$\mathsf{E}^\sharp[\![\, expr \,]\!] : \mathcal{E}^\sharp \to \mathcal{D}^\sharp$ est une abstraction de $\mathsf{E}[\![\, expr \,]\!] : \mathcal{E} \to \mathcal{P}(\mathbb{Z})$

$$
\begin{array}{lcl}
\mathsf{E}^\sharp[\![\, V \,]\!]\, X^\sharp & \stackrel{\text{def}}{=} & X^\sharp(V) \\
\mathsf{E}^\sharp[\![\, c \,]\!]\, X^\sharp & \stackrel{\text{def}}{=} & c^\sharp \\
\mathsf{E}^\sharp[\![\, \textbf{rand}(a,b) \,]\!]\, X^\sharp & \stackrel{\text{def}}{=} & [a,b]^\sharp \\
\mathsf{E}^\sharp[\![\, -e \,]\!]\, X^\sharp & \stackrel{\text{def}}{=} & -^\sharp\; \mathsf{E}^\sharp[\![\, e \,]\!]\, X^\sharp \\
\mathsf{E}^\sharp[\![\, e_1 + e_2 \,]\!]\, X^\sharp & \stackrel{\text{def}}{=} & \mathsf{E}^\sharp[\![\, e_1 \,]\!]\, X^\sharp \;+^\sharp\; \mathsf{E}^\sharp[\![\, e_2 \,]\!]\, X^\sharp \\
\mathsf{E}^\sharp[\![\, e_1 - e_2 \,]\!]\, X^\sharp & \stackrel{\text{def}}{=} & \mathsf{E}^\sharp[\![\, e_1 \,]\!]\, X^\sharp \;-^\sharp\; \mathsf{E}^\sharp[\![\, e_2 \,]\!]\, X^\sharp \\
\mathsf{E}^\sharp[\![\, e_1 \times e_2 \,]\!]\, X^\sharp & \stackrel{\text{def}}{=} & \mathsf{E}^\sharp[\![\, e_1 \,]\!]\, X^\sharp \;\times^\sharp\; \mathsf{E}^\sharp[\![\, e_2 \,]\!]\, X^\sharp \\
\mathsf{E}^\sharp[\![\, e_1 / e_2 \,]\!]\, X^\sharp & \stackrel{\text{def}}{=} & \mathsf{E}^\sharp[\![\, e_1 \,]\!]\, X^\sharp \;/^\sharp\; \mathsf{E}^\sharp[\![\, e_2 \,]\!]\, X^\sharp
\end{array}
$$

Définition par induction sur la syntaxe en suivant le modèle de $\mathsf{E}[\![\, expr \,]\!]$ mais dans l'abstrait !

Sûreté

$\cup_{\rho \in \dot\gamma(X^\sharp)} \mathsf{E}[\![\, e \,]\!]\, \rho \subseteq \gamma(\mathsf{E}^\sharp[\![\, e \,]\!]\, X^\sharp)$

par composition de la sûreté de $+^\sharp$, $-^\sharp$, etc.

Sémantique abstraite (partielle)

$\mathsf{S}^\sharp[\![\, \textit{stat} \,]\!] : \mathcal{E}^\sharp \to \mathcal{E}^\sharp$ version abstraite de $\mathsf{S}[\![\, \textit{stat} \,]\!] : \mathcal{P}(\mathcal{E}) \to \mathcal{P}(\mathcal{E})$

- $\mathsf{S}[\![\, \textbf{skip} \,]\!] R \stackrel{\text{def}}{=} R$

  $\mathsf{S}^\sharp[\![\, \textbf{skip} \,]\!] X^\sharp \stackrel{\text{def}}{=} X^\sharp$ (identité)

- $\mathsf{S}[\![\, \textbf{halt} \,]\!] R \stackrel{\text{def}}{=} \emptyset$

  $\mathsf{S}^\sharp[\![\, \textbf{halt} \,]\!] X^\sharp \stackrel{\text{def}}{=} \dot\bot$ (arrêt)

- $\mathsf{S}[\![\, s_1;\ s_2 \,]\!] R \stackrel{\text{def}}{=} \mathsf{S}[\![\, s_2 \,]\!] (\mathsf{S}[\![\, s_1 \,]\!] R)$

  $\mathsf{S}^\sharp[\![\, s_1; s_2 \,]\!] X^\sharp \stackrel{\text{def}}{=} \mathsf{S}^\sharp[\![\, s_2 \,]\!] (\mathsf{S}^\sharp[\![\, s_1 \,]\!] X^\sharp)$ (composition)

- $\mathsf{S}[\![\, V \leftarrow e \,]\!] R \stackrel{\text{def}}{=} \{\, \rho[V \mapsto v] \mid \rho \in R,\ v \in \mathsf{E}[\![\, e \,]\!] \rho \,\}$

  $$\mathsf{S}^\sharp[\![\, V \leftarrow e \,]\!] X^\sharp \stackrel{\text{def}}{=} \begin{cases} X^\sharp[V \mapsto \mathsf{E}^\sharp[\![\, e \,]\!] X^\sharp] & \text{si } \mathsf{E}^\sharp[\![\, e \,]\!] X^\sharp \neq \bot \\ \dot\bot & \text{si } \mathsf{E}^\sharp[\![\, e \,]\!] X^\sharp = \bot \end{cases}$$

  Réduction : si $\mathsf{E}^\sharp[\![\, e \,]\!] X^\sharp = \bot$, alors $\mathsf{S}[\![\, V \leftarrow e \,]\!]$ retourne $\emptyset$
  $\implies$ nous renvoyons donc $\dot\bot$, qui est la représentation la plus précise de $\emptyset$ pour $\dot\sqsubseteq$

Sémantique abstraite (partielle) : tests

$\mathsf{C}^\sharp[\![\, cond \,]\!] : \mathcal{E}^\sharp \to \mathcal{E}^\sharp$ version abstraite de $\mathsf{C}[\![\, cond \,]\!] : \mathcal{P}(\mathcal{E}) \to \mathcal{P}(\mathcal{E})$

Cas de base et par induction :

- $\mathsf{C}^\sharp[\![\, \text{true} \,]\!]\, X^\sharp \stackrel{\text{def}}{=} \dot{\top}$
- $\mathsf{C}^\sharp[\![\, \text{false} \,]\!]\, X^\sharp \stackrel{\text{def}}{=} \dot{\bot}$
- $\mathsf{C}^\sharp[\![\, c_1 \vee c_2 \,]\!]\, X^\sharp \stackrel{\text{def}}{=} (\mathsf{C}^\sharp[\![\, c_1 \,]\!]\, X^\sharp) \mathbin{\dot{\cup}^\sharp} (\mathsf{C}^\sharp[\![\, c_2 \,]\!]\, X^\sharp)$
- $\mathsf{C}^\sharp[\![\, c_1 \wedge c_2 \,]\!]\, X^\sharp \stackrel{\text{def}}{=} (\mathsf{C}^\sharp[\![\, c_1 \,]\!]\, X^\sharp) \mathbin{\dot{\cap}^\sharp} (\mathsf{C}^\sharp[\![\, c_2 \,]\!]\, X^\sharp)$

La comparaison d'expressions $\mathsf{C}^\sharp[\![\, e_1 \bowtie e_2 \,]\!]$, $\bowtie \in \{=, \neq, <, >, \leq, \geq\}$ sera présentée plus tard !

Sûreté

$\mathsf{C}[\![\, c \,]\!]\, \gamma(X^\sharp) \subseteq \gamma(\mathsf{C}^\sharp[\![\, c \,]\!]\, X^\sharp)$

Sémantique abstraite (partielle)

- $\mathsf{S}[\![\, \textbf{assert } c \,]\!]\, R \stackrel{\text{def}}{=} \mathsf{C}[\![\, c \,]\!]\, R$

  $\mathsf{S}^\sharp[\![\, \textbf{assert } c \,]\!]\, X^\sharp \stackrel{\text{def}}{=} \mathsf{C}^\sharp[\![\, c \,]\!]\, X^\sharp$

- $\mathsf{S}[\![\, \textbf{if } c \textbf{ then } s_1 \textbf{ else } s_2 \,]\!]\, R$
  $\stackrel{\text{def}}{=} \mathsf{S}[\![\, s_1 \,]\!]\, (\mathsf{C}[\![\, c \,]\!]\, R) \cup \mathsf{S}[\![\, s_2 \,]\!]\, (\mathsf{C}[\![\, \neg c \,]\!]\, R)$

  $\mathsf{S}^\sharp[\![\, \textbf{if } c \textbf{ then } s_1 \textbf{ else } s_2 \,]\!]\, X^\sharp$
  $\stackrel{\text{def}}{=} \mathsf{S}^\sharp[\![\, s_1 \,]\!]\, (\mathsf{C}^\sharp[\![\, c \,]\!]\, X^\sharp) \mathbin{\dot\cup^\sharp} \mathsf{S}^\sharp[\![\, s_2 \,]\!]\, (\mathsf{C}^\sharp[\![\, \neg c \,]\!]\, X^\sharp)$

- $\mathsf{S}[\![\, \textbf{while } c \textbf{ do } s \,]\!]\, R \stackrel{\text{def}}{=} \mathsf{C}[\![\, \neg c \,]\!]\, (\mathrm{lfp}\, \lambda X. R \cup \mathsf{S}[\![\, s \,]\!]\, (\mathsf{C}[\![\, c \,]\!]\, X))$

  pour l'instant, nous pouvons écrire naïvement
  $\mathsf{S}^\sharp[\![\, \textbf{while } c \textbf{ do } s \,]\!]\, X^\sharp \stackrel{\text{def}}{=} \mathsf{C}^\sharp[\![\, \neg c \,]\!]\, (\mathrm{lfp}\, \lambda Y^\sharp. X^\sharp \cup \mathsf{S}^\sharp[\![\, s \,]\!]\, (\mathsf{C}^\sharp[\![\, c \,]\!]\, Y^\sharp))$

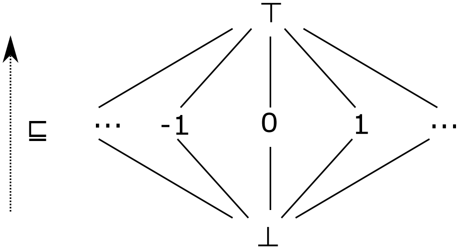
  le prochain cours expliquera le traitement des boucles en détail...

Sûreté

$\mathsf{S}[\![\, c \,]\!]\, \gamma(X^\sharp) \subseteq \gamma(\mathsf{S}^\sharp[\![\, c \,]\!]\, X^\sharp)$

Le domaine des constantes

Treillis des constantes



propriétés abstraites :

- $c \in \mathbb{Z}$: variable constante ;
- $\perp$: code non accessible ;
- $\top$: variable non constante.

Treillis complet, infini en largeur, mais “plat”.

Opérations sur les constantes

Correspondance de Galois :

$$\begin{aligned} \gamma(\bot) &\stackrel{\text{def}}{=} \emptyset \\ \gamma(c) &\stackrel{\text{def}}{=} \{c\} \\ \gamma(\top) &\stackrel{\text{def}}{=} \mathbb{Z} \end{aligned}$$

$$\alpha(S) \stackrel{\text{def}}{=} \begin{cases} \bot & \text{si } S = \emptyset \\ c & \text{si } S = \{c\} \\ \top & \text{sinon} \end{cases}$$

Opérateurs optimaux dérivés :

- $\cup^\sharp$ et $\cap^\sharp$ sont $\sqcup$ et $\sqcap$ pour l'ordre partiel ;
- $c^\sharp \stackrel{\text{def}}{=} c$;
- $[a, b]^\sharp \stackrel{\text{def}}{=} a$ si $a = b$, $\top$ sinon ;
- $X^\sharp +^\sharp Y^\sharp \stackrel{\text{def}}{=} \begin{cases} \bot & \text{si } X^\sharp \text{ ou } Y^\sharp = \bot \\ \top & \text{sinon si } X^\sharp \text{ ou } Y^\sharp = \top \\ X^\sharp + Y^\sharp & \text{sinon} \end{cases}$
- $X^\sharp \times^\sharp Y^\sharp \stackrel{\text{def}}{=} \begin{cases} \bot & \text{si } X^\sharp \text{ ou } Y^\sharp = \bot \\ 0 & \text{sinon si } X^\sharp \text{ ou } Y^\sharp = 0 \\ \top & \text{sinon si } X^\sharp \text{ ou } Y^\sharp = \top \\ X^\sharp \times Y^\sharp & \text{sinon} \end{cases}$
- ...

Opérations sur les constantes (suite)

Exemples de test :

- $C^\sharp[\![\, X - c = 0 \,]\!]\, R^\sharp \stackrel{\text{def}}{=} \begin{cases} \dot\perp & \text{si } R^\sharp(X) \notin \{c, \top\} \\ R^\sharp[X \mapsto c] & \text{sinon} \end{cases}$

- $C^\sharp[\![\, X - Y - c = 0 \,]\!]\, R^\sharp \stackrel{\text{def}}{=}$

$$\left(\begin{cases} C^\sharp[\![\, X - (R^\sharp(Y) + c) = 0 \,]\!]\, R^\sharp & \text{si } R^\sharp(Y) \notin \{\perp, \top\} \\ R^\sharp & \text{sinon} \end{cases} \right) \dot\cap^\sharp$$

$$\left(\begin{cases} C^\sharp[\![\, Y - (R^\sharp(X) - c) = 0 \,]\!]\, R^\sharp & \text{si } R^\sharp(X) \notin \{\perp, \top\} \\ R^\sharp & \text{sinon} \end{cases} \right)$$

une variable constante détermine l'autre variable constante

- Note :
$C^\sharp[\![\, c \,]\!]\, R^\sharp \stackrel{\text{def}}{=} R^\sharp$ est toujours un choix possible, sûr mais peu précis.

Exemple d'analyse

Exemple :

$$
\begin{array}{l}
X \leftarrow 0;\ Y \leftarrow 10; \\
\textbf{while } X < 100 \textbf{ do} \\
\quad Y \leftarrow Y - 3; \\
\quad X \leftarrow X + Y; \bullet \\
\quad Y \leftarrow Y + 3 \\
\textbf{done}
\end{array}
$$

Nous supposons ici que le **while** est calculé avec des itérations de Kleene, comme dans la sémantique concrète.

(c.f. cours suivant pour l'analyse détaillée des boucles)

L'analyse dans le domaine des constante trouve à $\bullet$: $\begin{cases} X = \top \\ Y = 7 \end{cases}$

Note :

l'analyse découvre des constantes qui n'apparaissent pas syntaxiquement dans le programme.

Le domaine des intervalles

Les intervalles entiers

Idée :

abstraire les comportements du programme
par la borne supérieure et la borne inférieure de chaque variable.

$$\mathcal{D}^\sharp \stackrel{\text{def}}{=} \{\, [a,b] \mid a \in \mathbb{Z} \cup \{-\infty\}, b \in \mathbb{Z} \cup \{+\infty\}, a \leq b \,\} \cup \{\bot\}$$

- Les valeurs de borne $-\infty$, $+\infty$ sont nécessaires ;
  elles permettent de représenter des ensembles non-bornés ;

  $[-\infty, +\infty] = \top$ représente $\mathbb{Z}$,
  $[0, +\infty]$ représente $\mathbb{N}$, etc.

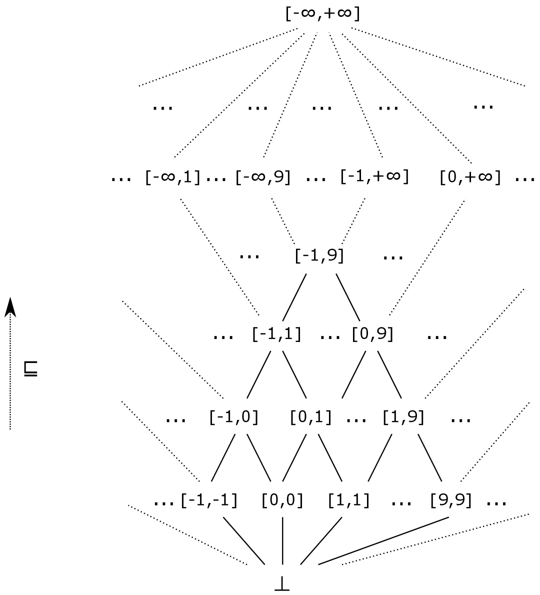
  $\Longrightarrow$ tout ensenble d'entiers a une sur-approximation dans $\mathcal{D}^\sharp$

  au pire, cette sur-approximation est $\top = [-\infty, +\infty]$

- $\bot$ est l'unique représentant de $\emptyset$

  dans $[a, b]$, nous imposons $a \leq b$; les intervalles non$-\bot$ ne sont jamais vides

Le treillis des intervalles



Structure algébrique

Ordre partiel : $\sqsubseteq$

- $\forall I \in \mathcal{D}^\sharp \colon \bot \sqsubseteq I$
- $[a,b] \sqsubseteq [c,d] \iff a \geq c \wedge b \leq d$

  où $\leq$ est étendu naturellement à $\mathbb{Z} \cup \{-\infty, +\infty\}$ par : $\forall c \in \mathbb{Z} \colon -\infty < c < +\infty$

Treillis : $\sqcup, \sqcap$

- plus petit majorant $\sqcup$ pour $\sqsubseteq$
  - $\forall I \in \mathcal{D}^\sharp \colon \bot \sqcup I = I \sqcup \bot = I$
  - $[a,b] \sqcup [c,d] = [\min(a,c), \max(b,d)]$
- plus grand minorant $\sqcap$:
  - $\forall I \colon \bot \sqcap I = I \sqcap \bot = \bot$
  - $[a,b] \sqcap [c,d] = \begin{cases} [\max(a,c), \min(b,d)] & \text{si } \max(a,c) \leq \min(b,d) \\ \bot & \text{si } \max(a,c) > \min(b,d) \end{cases}$

Structure algébrique (suite)

Notes :

- le treillis est complet ;

  $\forall I \subseteq \mathcal{D}^\sharp$: $\sqcup\ I$ et $\sqcap\ I$ existent

  $\sqcup \{\, [a_j, b_j] \mid j \in J \,\} = [\min_{j \in J} a_j, \max_{j \in J} b_j]$

  $\sqcap \{\, [a_j, b_j] \mid j \in J \,\} = [\max_{j \in J} a_j, \min_{j \in J} b_j]$ si max $\leq$ min, ou $\perp$ sinon

- $\cap^\sharp = \sqcap$

  l'ensemble des intervalles est clos par $\cap$;

  $[a, b] \cap [c, d] = [a, b] \sqcap [c, d]$

- $\cup^\sharp = \sqcup$

  l'ensemble des intervalles n'est pas clos par $\cup$.

  $[0, 0] \cup [2, 2] = \{0, 2\}$, qui n'est pas un intervalle ;

  $[0, 0] \sqcup [2, 2] = [0, 2]$

$\Longrightarrow$ perte de précision potentielle dans l'analyse !

Correspondance de Galois pour les intervalles

Correspondance de Galois : $(\mathcal{P}(\mathbb{Z}), \subseteq) \xleftarrow[\alpha]{\gamma} (\mathcal{D}^\sharp, \sqsubseteq)$

- $\begin{cases} \gamma(\bot) \stackrel{\text{def}}{=} \emptyset \\ \gamma([a,b]) \stackrel{\text{def}}{=} \{\, x \in \mathbb{Z} \mid a \leq x \leq b \,\} \end{cases}$

- $\alpha(X) \stackrel{\text{def}}{=} \begin{cases} \bot & \text{si } X = \emptyset \\ [\min X, \max X] & \text{si } X \neq \emptyset \end{cases}$

Preuve : on a bien $\alpha(X) \sqsubseteq I \iff X \subseteq \gamma(I)$, car

$$\begin{array}{lll} & \alpha(X) \sqsubseteq (a,b) & \\ \iff & \min X \geq a \wedge \max X \leq b & (\text{def. } \alpha, \sqsubseteq) \\ \iff & \forall x \in X: a \leq x \leq b & (\text{def. min, max}) \\ \iff & \forall x \in X: x \in \{\, y \mid a \leq y \leq b \,\} & \\ \iff & \forall x \in X: x \in \gamma([a,b]) & (\text{def. } \gamma) \\ \iff & X \subseteq \gamma([a,b]) & (\text{prop. } \subseteq) \end{array}$$

Arithmétique d'intervalles : addition, soustraction

- $-^\sharp\ [a, b] = [-b, -a]$
- $[a, b]\ +^\sharp\ [c, d] = [a + c, b + d]$
- $[a, b]\ -^\sharp\ [c, d] = [a - d, b - c]$
- $\forall I \in \mathcal{D}^\sharp$: $-^\sharp\ \perp = \perp\ +^\sharp\ I = I\ +^\sharp\ \perp = \cdots = \perp$

  les opérateurs sont stricts : ils retournent $\perp$ si un argument est $\perp$

où $+$ et $-$ sont étendus aux bornes $+\infty$ et $-\infty$ par :
$\forall x \in \mathbb{Z}$: $(+\infty) + x = +\infty,\ (-\infty) + x = -\infty,\ -(+\infty) = (-\infty), \ldots$

Preuve : optimalité de $+^\sharp$

$$
\begin{aligned}
&\alpha(\gamma([a, b]) \mathbin{\overline{+}} \gamma([c, d])) \\
&= \alpha(\{\, x \mid a \leq x \leq b \,\} \mathbin{\overline{+}} \{\, y \mid c \leq y \leq d \,\}) \\
&= \alpha(\{\, x + y \mid a \leq x \leq b \wedge c \leq y \leq d \,\}) \\
&= [\min \{\, x + y \mid a \leq x \leq b \wedge c \leq y \leq d \,\}, \max \{\, x + y \mid a \leq x \leq b \wedge c \leq y \leq d \,\}] \\
&= [a + c, b + d] \\
&= [a, b]\ +^\sharp\ [c, d]
\end{aligned}
$$

Quizz : $+^\sharp$, $-^\sharp$ sont-ils exacts ?

Arithmétique d'intervalles : multiplication

- $[a, b] \times^\sharp [c, d] = [\ \min(a \times c,\ a \times d,\ b \times c,\ b \times d),$
  $\max(a \times c,\ a \times d,\ b \times c,\ b \times d)\ ]$

où $\times$ est étendu aux bornes $+\infty$ et $-\infty$ par la règle des signes :

$c \times (+\infty) = (+\infty)$ si $c > 0$, $(-\infty)$ si $c < 0$

$c \times (-\infty) = (-\infty)$ si $c > 0$, $(+\infty)$ si $c < 0$

et aussi la règle non-standard : $0 \times (+\infty) = 0 \times (-\infty) = 0$

Quizz : $\times^\sharp$ est-il exact ? optimal ?

Arithmétique d'intervalles : division

- $/^\sharp$ par cas :

  $([a,b] \;/^\sharp\; ([c,d] \cap^\sharp [1,+\infty])) \quad \cup^\sharp \quad ([a,b] \;/^\sharp\; ([c,d] \cap^\sharp [-\infty,-1]))$

  où

  $$[a,b] \;/^\sharp\; [c,d] = \begin{cases} [\min(a/c, a/d), \max(b/c, b/d)] & \text{si } 1 \leq c \\ [\min(b/c, b/d), \max(a/c, a/d)] & \text{si } d \leq -1 \end{cases}$$

et $/$ est étendu aux bornes $+\infty$ et $-\infty$ par la règle des signes :

$c/(+\infty) = c/(-\infty) = 0$, en particulier $(+\infty)/(+\infty) = 0$
$(+\infty)/c = (+\infty)$ si $c > 0$, $(-\infty)$ si $c < 0$
$(-\infty)/c = (-\infty)$ si $c > 0$, $(+\infty)$ si $c < 0$

Exemples :

$[-5,5]/^\sharp[0,0] = \perp$

$[5,10]/^\sharp[-1,1] = ([5,10]/^\sharp[1,1]) \cup^\sharp ([5,10]/^\sharp[-1,-1]) = [5,10] \cup^\sharp [-10,-5] = [-10,10]$

Test dans les intervalles : cas particuliers simples

Tests simples : comparaison entre variables et constantes

on note ici : $X^\sharp(V) = [a, b]$ et $X^\sharp(W) = [c, d]$

- $C^\sharp[\![\, V \leq v \,]\!]\, X^\sharp \stackrel{\text{def}}{=} \begin{cases} X^\sharp[V \mapsto [a, \min(b, v)]] & \text{si } a \leq v \\ \dot\bot & \text{si } a > v \end{cases}$

- $C^\sharp[\![\, V \leq W \,]\!]\, X^\sharp \stackrel{\text{def}}{=} \begin{cases} X^\sharp[\begin{array}{l} V \mapsto [a, \min(b, d)], \\ W \mapsto [\max(a, c), d] \end{array}] & \text{si } a \leq d \\ \dot\bot & \text{si } a > d \end{cases}$

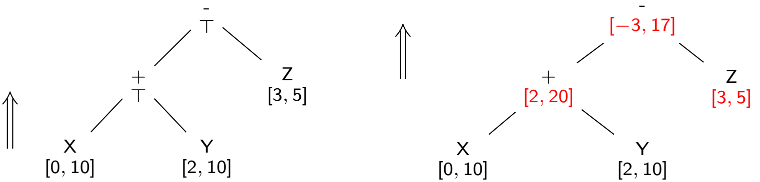
  la borne supérieure de W raffine la borne supérieure de V
  la borne inférieure de V raffine la borne inférieure de W

Test dans les intervalles : cas complexe (étape 1) $\overset{\star}{\star\star}$

Exemple : $C^\sharp[\![\, X + Y - Z \leq 0 \,]\!]\, X^\sharp$

où $X^\sharp = \{\, X \mapsto [0, 10], Y \mapsto [2, 10], Z \mapsto [3, 5] \,\}$

Première étape : annoter l'arbre d'expression avec des intervalles



(1) (2)

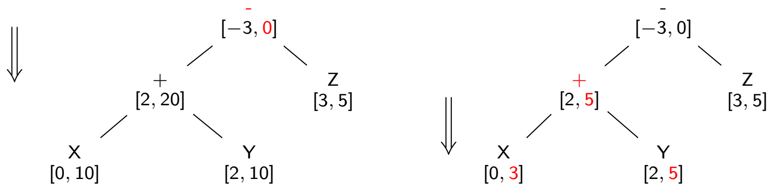
Évaluation *bottom-up* similaire à l'affectation d'intervalles,
en utilisant les opérateurs abstraits $+^\sharp$, $-^\sharp$, etc.
mais on se souvient en plus de tous les résultats intermédiaires.

Test dans les intervalles : cas complexe (étape 2) $\star\star\star$

Example : $C^\sharp[\![\, X + Y - Z \leq 0 \,]\!]\, X^\sharp$

avec $X^\sharp = \{\, X \mapsto [0, 10], Y \mapsto [2, 10], Z \mapsto [3, 5] \,\}$

Deuxième étape : raffinement de l'expression *top-down*.



(3) (4)

- raffine l'intervalle à la racine, sachant que le résultat est négatif ;
- propage les intervalles raffinés vers le bas, vers les feuilles ;
- les intervalles aux feuilles variables fournissent des nouvelles informations, à stocker dans l'environnement abstrait : $\{\, X \mapsto [0, 3], Y \mapsto [2, 5], Z \mapsto [3, 5] \,\}$

Opérateurs arithmétiques et de comparaison en arrière ⁂

Les opérateurs en arrière qui raffinent de manière sûre leurs arguments.

- Opérateur de comparaison, appliqué à la racine de l'expression :

  $\overleftarrow{\leq 0}^\sharp(X^\sharp) \stackrel{\text{def}}{=} X^\sharp \cap^\sharp [-\infty, 0]^\sharp$

- Opérateurs arithmétiques en arrière, appliqués aux nœuds :

  $\overleftarrow{-}^\sharp(X^\sharp, R^\sharp) \stackrel{\text{def}}{=} X^\sharp \cap^\sharp (-^\sharp\, R^\sharp)$

  $\overleftarrow{+}^\sharp(X^\sharp, Y^\sharp, R^\sharp) \stackrel{\text{def}}{=} (X^\sharp \cap^\sharp (R^\sharp -^\sharp Y^\sharp),\ Y^\sharp \cap^\sharp (R^\sharp -^\sharp X^\sharp))$

  $\overleftarrow{-}^\sharp(X^\sharp, Y^\sharp, R^\sharp) \stackrel{\text{def}}{=} (X^\sharp \cap^\sharp (R^\sharp +^\sharp Y^\sharp),\ Y^\sharp \cap^\sharp (X^\sharp -^\sharp R^\sharp))$

  $\overleftarrow{\times}^\sharp(X^\sharp, Y^\sharp, R^\sharp) \stackrel{\text{def}}{=} (X^\sharp \cap^\sharp (R^\sharp /^\sharp Y^\sharp),\ Y^\sharp \cap^\sharp (R^\sharp /^\sharp X^\sharp))$

  $\overleftarrow{/}^\sharp(X^\sharp, Y^\sharp, R^\sharp) \stackrel{\text{def}}{=} (X^\sharp \cap^\sharp (S^\sharp \times^\sharp Y^\sharp),\ Y^\sharp \cap^\sharp ((X^\sharp /^\sharp S^\sharp) \cup^\sharp [0,0]^\sharp))$

  où $S^\sharp = R^\sharp +^\sharp [-1,1]^\sharp$, car / tronque le résultat !

Étant donnés des arguments $X^\sharp$, $Y^\sharp$ originaux, et un résultat $R^\sharp$ raffiné,
retourne $X^{\sharp\prime}$ et $Y^{\sharp\prime}$ qui raffinent $X^\sharp$ et $Y^\sharp$,
tout en couvrant le résultat raffiné $R^\sharp$ (sûreté).

Opérateurs en arrière dans les intervalles ⁂

Application des constructions générique d'opérateurs en arrière au cas des intervalles :

- $\overleftarrow{\leq 0}^\sharp([a,b]) \stackrel{\text{def}}{=} \begin{cases} [a, \min(b,0)] & \text{si } a \geq 0 \\ \perp & \text{sinon} \end{cases}$

- $\overleftarrow{-}^\sharp([a,b],[r,s]) \stackrel{\text{def}}{=} [a,b] \cap^\sharp [-s,-r]$

- $\overleftarrow{+}^\sharp([a,b],[c,d],[r,s]) \stackrel{\text{def}}{=} ([a,b] \cap^\sharp [r-d, s-c],$
  $[c,d] \cap^\sharp [r-b, s-a])$

- ...

Implantation de la sémantique abstraite en OCaml

Rappel : itérateur générique et domaine concret

L'itérateur est générique
et paramétré par un domaine d'interprétation :

- foncteur `Interprete` dans `interpreter/interpreter.ml`
- paramétré par un domaine obéissant à la signature `DOMAIN` dans `domains/domain.ml`.
  abstraction de $\mathcal{P}(\mathbb{V} \to \mathbb{Z})$

Dans le cours précédent, nous avons vu l'interprète concret :

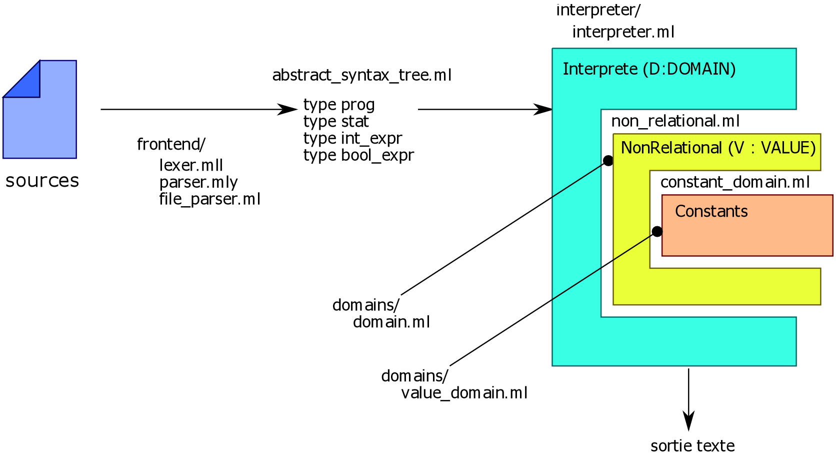
- module `Concrete` dans `domains/concrete_domain.ml` qui implante `DOMAIN`.

Analyse non-relationnelle

- pas de changement à l'itérateur
  pour l'instant. . .
- ajout d'une signature de domaines pour les valeurs, fournie
  - `VALUE_DOMAIN` dans `domains/value_domain.ml` ;
  - modélise des ensembles d'entiers $\mathcal{P}(\mathbb{Z})$.
- exemple de domaines de valeurs :
  - `Constants` dans `domains/constant_domain.ml`
    domaine des constantes (implantation partielle fournie, à compléter)
  - domaine des intervalles : non fourni, à faire intégralement !
- un foncteur non-relationnel générique, fourni :
  - `NonRelational` dans `domains/non_relational` ;
  - prend un domaine de valeurs `VALUE` en argument ;
  - fournit un domaine d'environnements `DOMAIN`
    avec la méthode vue en cours.

L'ajout d'une analyse non-relationnelle nécessite uniquement l'ajout d'un nouveau domaine de valeurs, de signature `VALUE_DOMAIN` :
$\Longrightarrow$ plus simple que d'ajouter un nouveau module de signature `DOMAIN` !

Schéma de fonctionnement de l'interprète non-relationnel



Domaine de valeurs : signature

domains/value_domain.ml

```
module type VALUE_DOMAIN = sig
    type t                    (* P(Z) *)

    val top: t                (* ⊤ *)
    val bottom: t             (* ⊥) *)
    val const: Z.t -> t       (* {c} *)
    val rand: Z.t -> Z.t -> t (* [a, b] *)

    val join: t -> t -> t     (* ∪ *)
    val meet: t -> t -> t     (* ∩ *)
    val subset: t -> t -> bool (* ⊆ *)
    val is_bottom: t -> bool  (* = ∅ ? *)

    val print: Format.formatter -> t -> unit

    val unary: t -> int_unary_op -> t                   (* − *)
    val binary: t -> t -> int_binary_op -> t           (* +, −, ×, / *)
    val compare: t -> t -> compare_op -> (t * t)  (* =, ≠, <, >, ≤, ≥ *)

    (* pour les tests complexes, optionnel *)
    val bwd_unary: t -> int_unary_op -> t -> t
    val bwd_binary: t -> t -> int_binary_op -> t -> (t * t)

    (* pour les boucles, voir prochain cours *)
    val widen: t -> t -> t
end
```

Domaine des constantes : représentation et ordre

domains/constant_domain.ml

```
module Constants = struct
    type t = Cst of Z.t  (* c *)
           | BOT         (* ⊥ *)
           | TOP         (* ⊤ *)

    let top      = TOP
    let bottom   = BOT
    let const c  = Cst c
    let rand x y = if x=y then Cst x else if x<y then TOP else BOT

    let subset a b = match a,b with
    | BOT,_ | _,TOP -> true
    | Cst x, Cst y -> x=y
    | _ -> false

    let join a b = match a,b with
    | BOT,x | x,BOT -> x
    | Cst x, Cst y when x=y -> a
    | _ -> TOP

    let meet a b = match a,b with
    | TOP,x | x,TOP -> x
    | Cst x, Cst y when x=y -> a
    | _ -> BOT

    ...
end: VALUE_DOMAIN
```

Domaine des constantes : opérateurs

domains/constant_domain.ml

```
(* applique f à une constante, ⊥ → ⊥, ⊤ → ⊤ *)
let lift1 f x = match x with
  | BOT -> BOT
  | TOP -> TOP
  | Cst a -> Cst (f a)

(* idem pour une fonction binaire*)
let lift2 f x y = ...

(* ces opérations sont-elles optimales ? *)
let neg = lift1 Z.neg
let add = lift2 Z.add
let sub = lift2 Z.sub
let mul = lift2 Z.mul
let div a b = if b = Cst Z.zero then BOT else lift2 Z.div a b

(* à faire *)
let eq a b = a,b
...

(* dispatch *)
let unary x op = match op with ...
let binary x y op = match op with ...
let compare x y op = match op with ...

(* tests complexes, à lire chez soi *)
let bwd_unary = ...
```

Domaine non-relationnel générique : représentation

domains/non_relational.ml

```
module NonRelational(V:VALUE_DOMAIN) =
struct

  module VarMap = Mapext.Make(String)
  type env = V.t VarMap.t  (* 𝕍 → 𝒟♯ *)

  type t = Val of env | BOT (* ℰ♯ ≝ (𝕍 → 𝒟♯) ∪ {⊥} *)

  (* arbre d'expression annoté *)
  type atree =
    | A_unary  of int_unary_op * atree * V.t
    | A_binary of int_binary_op * atree * V.t * atree * V.t
    | A_var    of var * V.t
    | A_cst    of V.t

  (* évaluation d'expression : arbre annoté et valeur abstraite *)
  let rec eval (m:env) (e:int_expr) : atree * V.t = match e with ...

  ...

end: DOMAIN
```

Domaine non-relationnel générique : environnement

domains/non_relational.ml

```
let init ()   = Val VarMap.empty

let bottom () = BOT

let add_var a var = match a with
| BOT -> BOT
| Val m -> Val (VarMap.add var (V.const Z.zero) m)

let del_var a var = match a with
| BOT -> BOT
| Val m -> Val (VarMap.remove var m)

(* assignment *)
let assign a var e = match a with
| BOT -> BOT
| Val m ->
    let _,v = eval m e in
    if V.is_bottom v then BOT
    else Val (VarMap.add var v m)
```

Domaine non-relationnel générique : ordre

domains/non_relational.ml

```
let subset a b = match a,b with
| BOT,_ -> true
| _,BOT -> false
| Val m, Val n -> VarMap.for_all2z (fun _ x y -> V.subset x y) m n

let join a b = match a,b with
| BOT,x | x,BOT -> x
| Val m, Val n -> Val (VarMap.map2z (fun _ x y -> V.join x y) m n)

exception Empty

let meet a b = match a,b with
| BOT,x | x,BOT -> x
| Val m, Val n ->
    try Val
        (VarMap.map2z
           (fun _ x y ->
             let r = V.meet x y in
             if V.is_bottom r then raise Empty;
             r
           ) m n)
    with Empty -> BOT
```