Produits réduits, domaines disjonctifs

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

Antoine Miné

Année 2016–2017

Cours 11
2 mars 2017

Plan du cours

- combinaison de domaines abstraits :
  - produit réduit ;

  $\Longrightarrow$ construction modulaire d'analyseurs
  par combinaison de briques d'abstraction

- domaines disjonctifs :
  transformateurs de domaines génériques
  permettant de représenter des ensembles non convexes
  - complétion disjonctive
  - partitionnement d'états
  - partitionnement de traces

- conseils d'implantation pour le projet.

Produits réduits

Principe

Théorie :

- l'ensemble des domaines abstraits forme un treillis,
- ordonné par la notion d'abstraction, qui est un ordre partiel, où $(C, \leq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (A, \sqsubseteq)$ indique que C est plus concret que A, i.e., toute propriété de A peut être représentée dans C ;
- il existe alors un plus petit majorant $\sqcup$ de domaines et un plus grand minorant $\sqcap$ de domaines.

Application : produit réduit

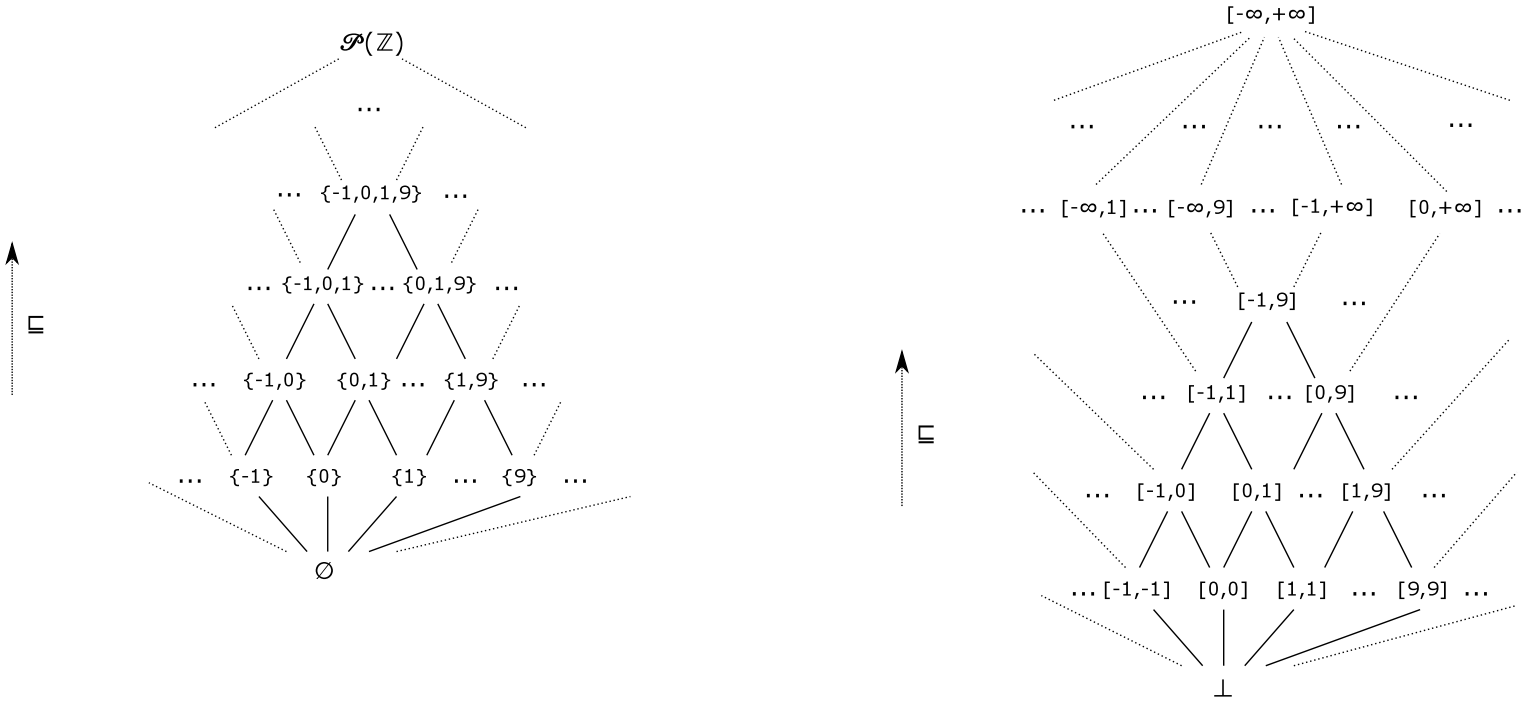
Construction effective du plus grand minorant $A_1 \sqcap A_2$ de deux abstractions,
donc capable de représenter toutes les propriétés représentables dans A_1 ou dans A_2.

Conséquence

Une analyse statique **précise** peut être construite en **combinant** des abstractions élémentaires.

Treillis des abstractions

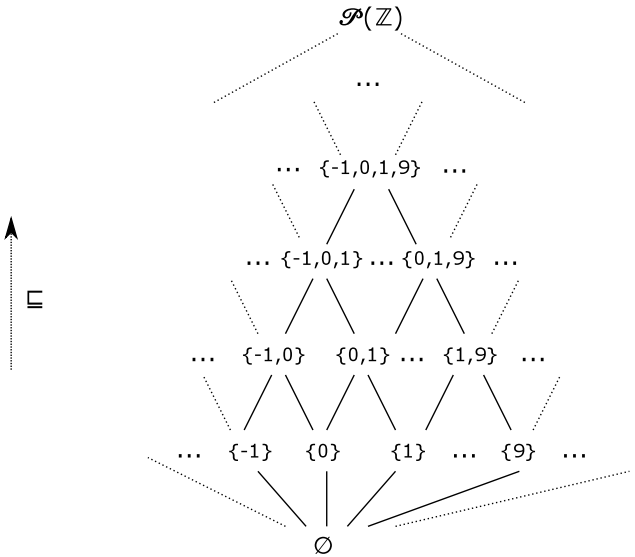
Rappel : abstraction des intervalles

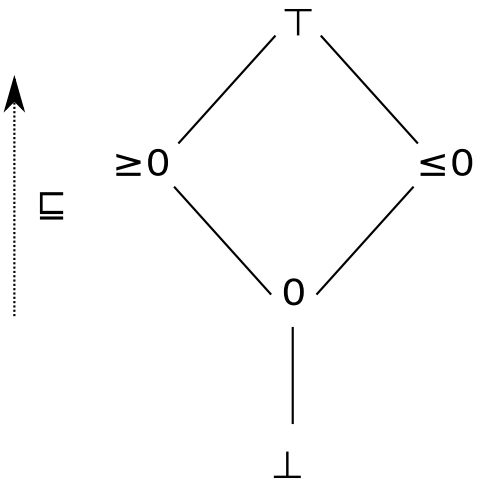


$$\mathcal{P}(\mathbb{Z}) \underset{\alpha_i}{\overset{\gamma_i}{\leftrightarrows}} \{\, [a,b] \mid a \leq b \,\} \cup \{\bot\}$$

- $\alpha_i(S) \overset{\text{def}}{=} [\min S, \max S]$
- $\gamma_i([a,b]) \overset{\text{def}}{=} \{\, x \in \mathbb{Z} \mid a \leq x \leq b \,\}$

Rappel : abstraction des signes



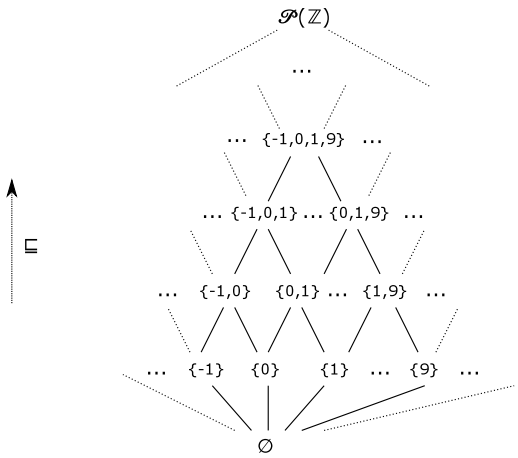


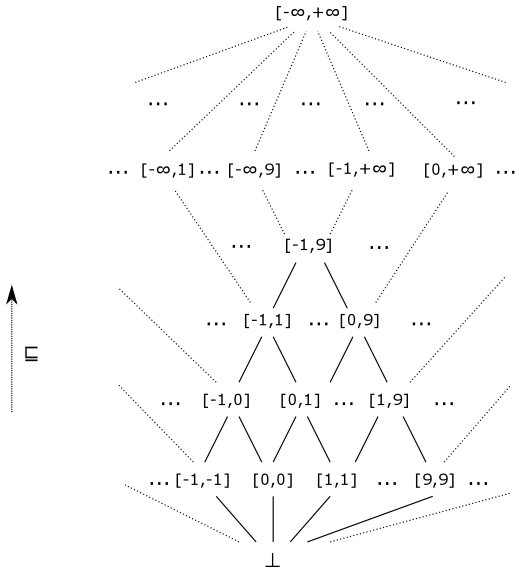
$\mathcal{P}(\mathbb{Z}) \underset{\alpha_s}{\overset{\gamma_s}{\leftrightarrows}} \{\bot, 0, \leq 0, \geq 0, \top\}$

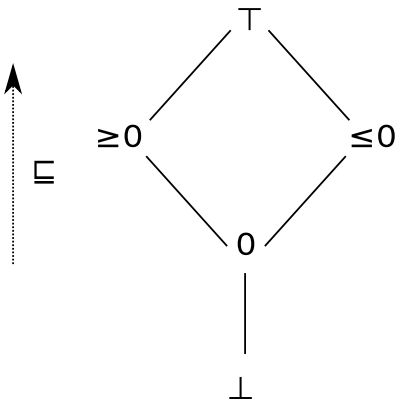
$$\begin{array}{lcl} \gamma_s(\bot) & \overset{\text{def}}{=} & \emptyset \\ \gamma_s(0) & \overset{\text{def}}{=} & \{0\} \\ \gamma_s(\geq 0) & \overset{\text{def}}{=} & \mathbb{N} \\ \gamma_s(\leq 0) & \overset{\text{def}}{=} & -\mathbb{N} \\ \gamma_s(\top) & \overset{\text{def}}{=} & \mathbb{Z} \end{array}$$

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

Composition d'abstractions







$$\mathcal{P}(\mathbb{Z}) \underset{\alpha_i}{\overset{\gamma_i}{\leftrightarrows}} \{\,[a,b] \mid a \leq b\,\} \cup \{\bot\} \underset{\alpha'_s}{\overset{\gamma'_s}{\leftrightarrows}} \{\bot, 0, \leq 0, \geq 0, \top\}$$

où $\gamma'_s(\bot) \overset{\text{def}}{=} \bot$ $\quad \gamma'_s(\top) \overset{\text{def}}{=} [-\infty, +\infty]$

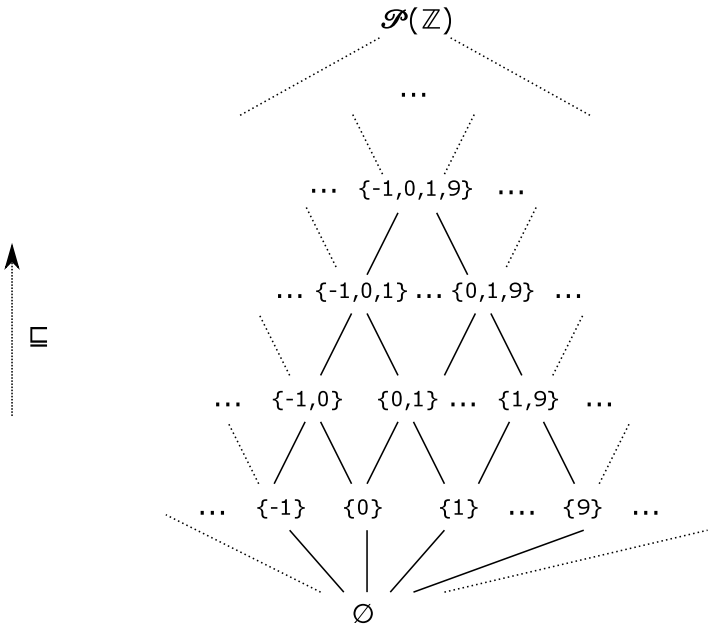
$\gamma'_s(\geq 0) \overset{\text{def}}{=} [0, +\infty]$ $\quad \gamma'_s(\leq 0) \overset{\text{def}}{=} [-\infty, 0]$ $\quad \gamma'_s(0) \overset{\text{def}}{=} [0, 0]$

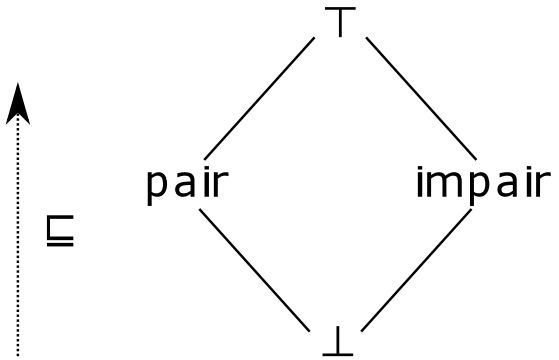
Composition d'abstractions : les correspondances de Galois se composent

Si $(X_1, \sqsubseteq_1) \underset{\alpha_1}{\overset{\gamma_1}{\leftrightarrows}} (X_2, \sqsubseteq_2) \underset{\alpha_2}{\overset{\gamma_2}{\leftrightarrows}} (X_3, \sqsubseteq_3)$, alors $(X_1, \sqsubseteq_1) \underset{\alpha_2 \circ \alpha_1}{\overset{\gamma_1 \circ \gamma_2}{\leftrightarrows}} (X_3, \sqsubseteq_3)$.

preuve : $(\alpha_2 \circ \alpha_1)(c) \sqsubseteq_3 a \iff \alpha_1(c) \sqsubseteq_2 \gamma_2(a) \iff c \sqsubseteq_1 (\gamma_1 \circ \gamma_2)(a)$

Un nouveau domaine simple : le domaine des parités



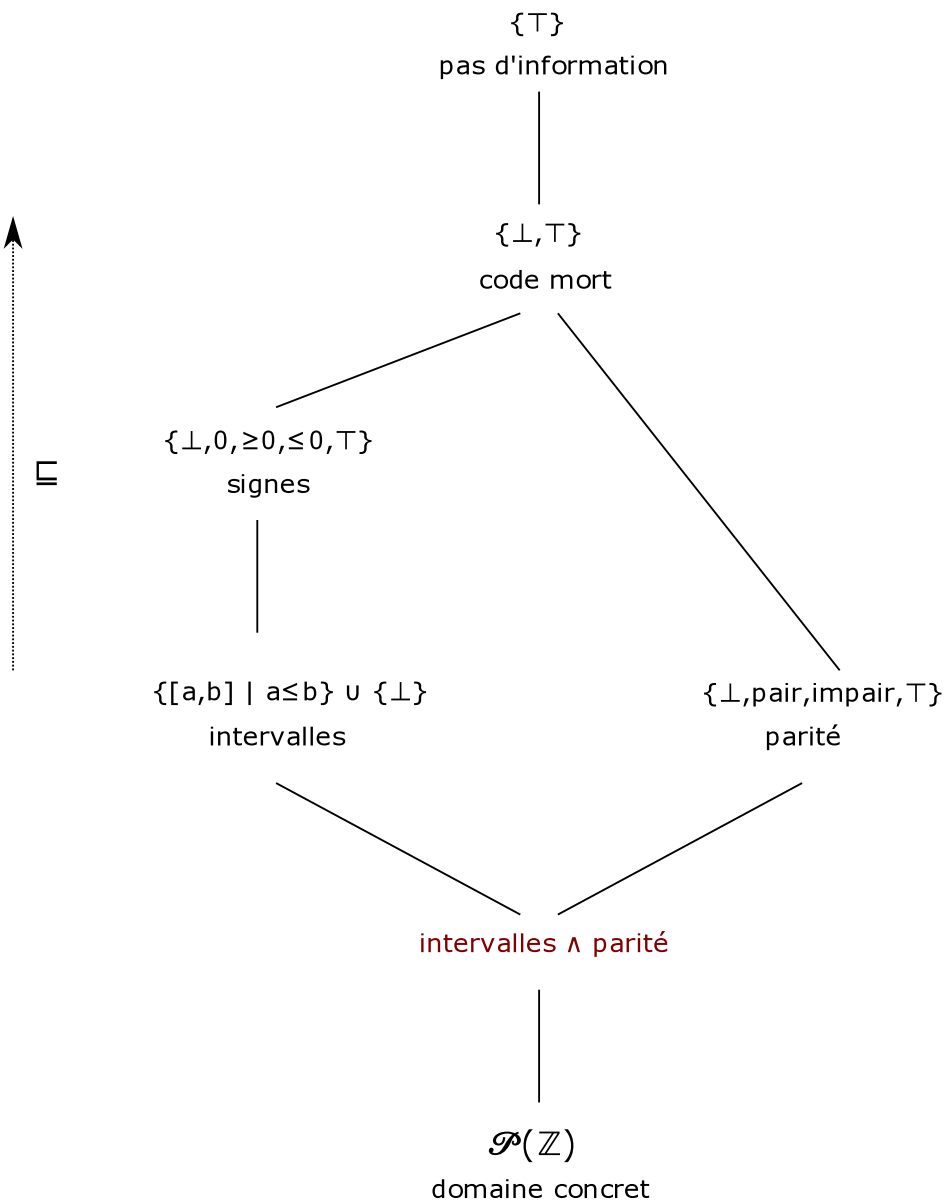


$$\mathcal{P}(\mathbb{Z}) \underset{\alpha_p}{\overset{\gamma_p}{\leftrightarrows}} \{\bot, \top, \mathit{pair}, \mathit{impair}\}$$

$$\begin{aligned}
\gamma_p(\bot) &\overset{\text{def}}{=} \emptyset \\
\gamma_p(\mathit{pair}) &\overset{\text{def}}{=} 2\mathbb{Z} \\
\gamma_p(\mathit{impair}) &\overset{\text{def}}{=} 2\mathbb{Z}+1 \\
\gamma_p(\top) &\overset{\text{def}}{=} \mathbb{Z}
\end{aligned}$$

$$\alpha_p(S) \overset{\text{def}}{=} \begin{cases} \bot & \text{si } S = \emptyset \\ \mathit{pair} & \text{sinon, si } S \subseteq 2\mathbb{Z} \\ \mathit{impair} & \text{sinon, si } S \subseteq 2\mathbb{Z}+1 \\ \top & \text{sinon} \end{cases}$$

Treillis des abstractions de $\mathcal{P}(\mathbb{Z})$



- les signes sont plus abstraits que les intervalles ;
  toutes propriété de signe est exprimable comme un intervalle
- les parités et les intervalles sont incomparables ;
  aucune propriété commune, excepté $\bot$ et $\top$
- $\mathcal{P}(\mathbb{Z})$ est le domaine le plus concret ;
- $\{\top\}$ est le domaine le plus abstrait ;
- intervalles $\wedge$ parité est le domaine le plus abstrait à la fois plus précis que les intervalles et plus précis que les parités.

Construction du produit réduit

Produit simple de domaines

Structure :

Étant donnés deux domaines $(\mathcal{D}_1^\sharp, \sqsubseteq_1)$ et $(\mathcal{D}_2^\sharp, \sqsubseteq_2)$,
nous représentons dans $\mathcal{D}^\sharp$ les paires d'éléments abstraits ;
elles correspondent à des conjonctions de propriétés.

- $\mathcal{D}_{1\times 2}^\sharp \stackrel{\text{def}}{=} \mathcal{D}_1^\sharp \times \mathcal{D}_2^\sharp$
- $\gamma_{1\times 2}(X_1^\sharp, X_2^\sharp) \stackrel{\text{def}}{=} \gamma_1(X_1^\sharp) \cap \gamma_2(X_2^\sharp)$
- $\alpha_{1\times 2}(S) \stackrel{\text{def}}{=} (\alpha_1(S), \alpha_2(S))$
- $(X_1^\sharp, X_2^\sharp) \sqsubseteq_{1\times 2} (Y_1^\sharp, Y_2^\sharp) \stackrel{\text{def}}{\iff} X_1^\sharp \sqsubseteq_1 Y_1^\sharp \quad \text{and} \quad X_2^\sharp \sqsubseteq_2 Y_2^\sharp$

Opérations abstraites dans $\mathcal{D}^\sharp$:

Effectuées en parallèle sur chaque élément, dans son domaine respectif :

- $(X_1^\sharp, X_2^\sharp) \cup_{1\times 2}^\sharp (Y_1^\sharp, Y_2^\sharp) \stackrel{\text{def}}{=} (X_1^\sharp \cup_1^\sharp Y_1^\sharp,\ X_2^\sharp \cup_2^\sharp Y_2^\sharp),$;
- $(X_1^\sharp, X_2^\sharp) \triangledown_{1\times 2} (Y_1^\sharp, Y_2^\sharp) \stackrel{\text{def}}{=} (X_1^\sharp \triangledown_1 Y_1^\sharp,\ X_2^\sharp \triangledown_2 Y_2^\sharp)$;
- $S^\sharp[\![s]\!]_{1\times 2}(X_1^\sharp, X_2^\sharp) \stackrel{\text{def}}{=} (S^\sharp[\![s]\!]_1(X_1^\sharp),\ S^\sharp[\![s]\!]_2(X_2^\sharp))$.

Produit simple d'analyses : limitation

$V \leftarrow 1$;
while $V \leq 10$ **do** $V \leftarrow V + 2$ **done**;
● **if** $V \geq 12$ **then** ● $V \leftarrow 0$ ●;

Analyse dans le domaine des intervalles, des parités, et leur produit :

	intervalles	parités	produit : intervalles×parités
●	$V \in [11, 12]$	V impair	$V \in [11, 12] \wedge V$ impair
●	$V = 12$	V impair	$V = 12 \wedge V$ impair
●	$V = 0$	V pair	$V = 0 \wedge V$ pair

L'analyse est identique à deux analyses séparées :

- en ●, elle trouve $V = 12 \wedge V$ impair qui est insatisfiable, i.e., $\emptyset$;
- en ●, l'application indépendente de $V \leftarrow 0$
  sur les intervalles et les parités redonne $V = 0$.
  $\gamma_{1\times 2}(S^\sharp[\![V \leftarrow 0]\!](X^\sharp, Y^\sharp)) \neq \emptyset$ alors que $\gamma_{1\times 2}(X^\sharp, Y^\sharp) = \emptyset$!

$\Longrightarrow$ perte de précision importante.

Produit totalement réduit

Principe : propager des informations entre les domaines

Étant donnés les correspondances de Galois (α_1, γ_1) et (α_2, γ_2) sur $\mathcal{D}_1^\sharp$ et $\mathcal{D}_2^\sharp$ nous définissons l'opérateur de réduction ρ comme :

$$\rho : \mathcal{D}_{1\times 2}^\sharp \to \mathcal{D}_{1\times 2}^\sharp$$
$$\rho(X_1^\sharp, X_2^\sharp) \stackrel{\text{def}}{=} (\alpha_1(\gamma_1(X_1^\sharp) \cap \gamma_2(X_2^\sharp)),\ \alpha_2(\gamma_1(X_1^\sharp) \cap \gamma_2(X_2^\sharp)))$$

i.e., la meilleur représentation de $\gamma_{1\times 2}(X_1^\sharp, X_2^\sharp)$ dans chacun des domaines.

Application :

Utilisation de ρ pour affiner le résultat après chaque opération abstraite :

- $(X_1^\sharp, X_2^\sharp) \cup_{1\times 2}^\sharp (Y_1^\sharp, Y_2^\sharp) \stackrel{\text{def}}{=} \rho(X_1^\sharp \cup_1^\sharp Y_1^\sharp,\ X_2^\sharp \cup_2^\sharp Y_2^\sharp)$,
- $S^\sharp[\![s]\!]_{1\times 2}(X_1^\sharp, X_2^\sharp) \stackrel{\text{def}}{=} \rho(S^\sharp[\![s]\!]_1(X_1^\sharp),\ S^\sharp[\![s]\!]_2(X_2^\sharp))$.

Attention :

ne pas appliquer ρ après ∇ ; cela pourrait empêcher la convergence des itérés dans $\mathcal{D}_1^\sharp \times \mathcal{D}_2^\sharp$!
intuition : ∇ accélère la croissance, tandis que ρ la limite.

Exemple d'analyse avec réduction

$V \leftarrow 1$;
while $V \leq 10$ **do** $V \leftarrow V + 2$ **done**;
• **if** $V \geq 12$ **then** $V \leftarrow 0$

Réduction ρ entre les intervalles et les parités $\rho([a, b], p)$:

D'abord raffiner les bornes de l'intervalle $[a, b]$ en utilisant la parité p, puis utiliser les bornes pour raffiner la parité :

- soit $a' = a + 1$ si $a \notin \gamma_p(p)$, $a' = a$ sinon ;
- soit $b' = b - 1$ si $b \notin \gamma_p(p)$, $b' = b$ sinon ;
- si $a' > b'$, retourner $(\bot, \bot)$;
- si $a' = b'$, retourner $([a', b'], \alpha_p(a))$;
- sinon, retourner $([a', b'], p)$.

Exemple :

En •, $\rho([11, 12], \text{impair}) = ([11, 11], \text{impair})$
$\Longrightarrow$ la branche **then** n'est pas atteignable.

Limitations de la réduction totale, réduction partielle

La réduction ρ optimale est bien définie mathématiquement, mais :

- ρ suppose l'existence de correspondances de Galois ;
- la formule ne donne pas d'algorithme effectif pour implanter ρ.

  cas similaire à l'abstraction optimale d'un opérateur : $F^\sharp \stackrel{\text{def}}{=} \alpha \circ F \circ \gamma$

Réduction partielle :

Définition pragmatique, quand la réduction optimale n'est pas utilisable :

- $\rho(X_1^\sharp, X_2^\sharp) = (Y_1^\sharp, Y_2^\sharp)$ est une réduction partielle si :
- $Y_1^\sharp \sqsubseteq_1 X_1^\sharp$ et $Y_2^\sharp \sqsubseteq_2 X_2^\sharp$; (raffinement)
- $\gamma_{1\times 2}(Y_1^\sharp, Y_2^\sharp) = \gamma_{1\times 2}(X_1^\sharp, X_2^\sharp)$. (sûreté)

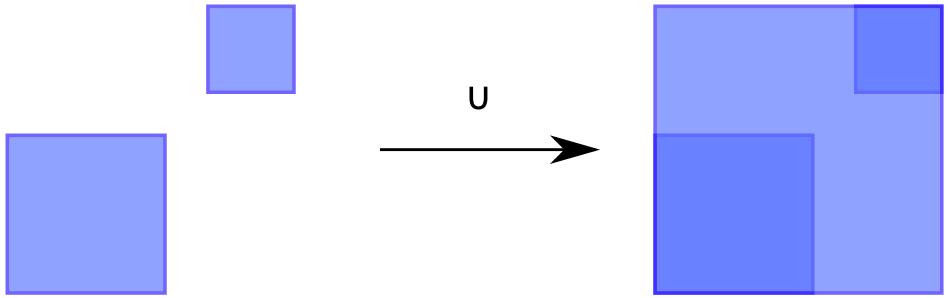
Exemple générique :

$$\rho(X_1^\sharp, X_2^\sharp) \stackrel{\text{def}}{=} \begin{cases} (\bot_1, \bot_2) & \text{si } X_1^\sharp = \bot_1 \text{ ou } X_2^\sharp = \bot_2 \\ (X_1^\sharp, X_2^\sharp) & \text{sinon} \end{cases}$$

En pratique, un analyseur comporte de nombreux domaines abstraits (expressivité) avec des réductions très limitées entre domaines (efficacité).

Domaines disjonctifs

Perte de précision : abstraction de l'union



Perte de précision :

Les domaines sont rarement clos par union :

- pour les constantes et la parité , $a \cup^\sharp b = \top$ si $a \neq b$;
- les intervalles ne représentent que des ensembles convexes ;

$\Longrightarrow$ perte de précision à chaque application de $\cup^\sharp$.

Particulièrement gênant car $\cup^\sharp$ est utilisé fréquemment :

- $C^\sharp[\![e_1 \vee e_2]\!]$
  ou logique dans un test
- $S^\sharp[\![\textbf{if } c \textbf{ then } s_1 \textbf{ else } s_2]\!]$
  fusion des branches **then** et **else**
- $S^\sharp[\![\textbf{while } c \textbf{ do } s]\!]$
  fusion des itérés pour inférer un invariant de boucle

Motivation

Remarque : la plus part des domaines représentent des **ensembles convexes**

conjonctions de contraintes

$\Longrightarrow \cup^\sharp$ cause des pertes de précision importantes !

Utilités des invariants non convexes

$X \leftarrow$ **rand**$(10, 20)$;
$Y \leftarrow$ **rand**$(0, 1)$;
if $Y > 0$ **then** $X \leftarrow -X$;
• $Z \leftarrow 100/X$

Sémantique concrète :

En •, $X \in [-20, -10] \cup [10, 20]$
$\Longrightarrow$ pas de division par zéro

Analyse abstraite :

Toute analyse convexe intervalles, polyèdres trouvera $X \in [-20, 20]$
dans les intervalles : $[-20, -10] \cup^\sharp [10, 20] = [-20, 20]$
$\Longrightarrow$ division par zéro possible (fausse alarme)

Domaines disjonctifs

Domaines disjonctifs :

constructions génériques pour étendre tout domaine abstrait numérique en un domaine représentant certaines disjonctions de manière exacte.

Principe :

utiliser plusieurs éléments abstraits à chaque point de programme au lieu d'un seul.

Exemples de construction :

- complétion par l'ensemble des parties
  "soupe" non structurée d'éléments abstraits
- partitionnement d'état
  fixer un partitionnement de $\mathcal{E}$, utiliser un élément abstrait par élément de partition
- analyses sensibles aux chemins d'exécution *(path-sensitive)*
  partitionnement vis à vis de l'historique des executions

chaque construction a ses forces et ses faiblesses ;
souvent, elles sont combinées dans une analyse

Complétion par l'ensemble des parties

Domaine de l'ensemble des parties

Étant donné : $(\mathcal{E}^\sharp, \sqsubseteq, \gamma, \cup^\sharp, \cap^\sharp, \triangledown, \mathsf{S}^\sharp[\![\, stat \,]\!])$

un domaine abstrait $\mathcal{E}^\sharp$
ordonné par $\sqsubseteq$
avec une concrétisation $\gamma : \mathcal{E}^\sharp \to \mathcal{P}(\mathcal{E})$
des abstractions sûres $\cup^\sharp$, $\cap^\sharp$, $\mathsf{S}^\sharp[\![\, stat \,]\!]$ de $\cup$, $\cap$, $\mathsf{S}[\![\, stat \,]\!]$
et un élargissement $\triangledown$

Le domaine des parties est : $(\hat{\mathcal{E}}^\sharp, \hat{\sqsubseteq}, \hat{\gamma}, \hat{\cup}^\sharp, \hat{\cap}^\sharp, \hat{\triangledown}, \hat{\mathsf{S}}^\sharp[\![\, stat \,]\!])$

- $\hat{\mathcal{E}}^\sharp \stackrel{\text{def}}{=} \mathcal{P}_{finis}(\mathcal{E}^\sharp)$ — ensembles finis d'éléments abstraits
- $\hat{\gamma}(A^\sharp) \stackrel{\text{def}}{=} \cup \{ \gamma(X^\sharp) \mid X^\sharp \in A^\sharp \}$ — union ensembliste des concrétisations

Exemple : avec le domaine des intervalles $\mathcal{E}^\sharp$

$\hat{\gamma}(\{[-10,-5],[2,4],[0,0],[2,3]\}) = [-10,-5] \cup \{0\} \cup [2,4]$

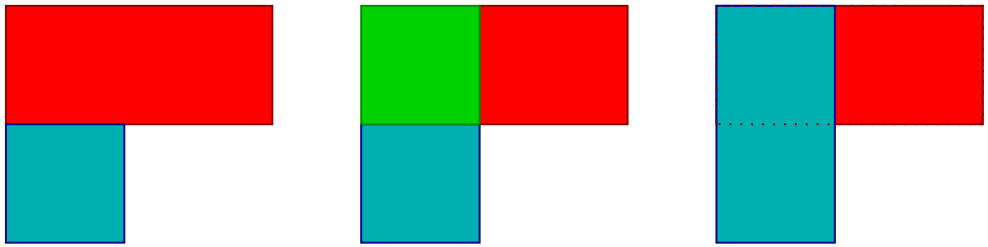
Ordre

Problème : comment comparer deux éléments de $\hat{\mathcal{E}}^\sharp$?

- $\hat{\gamma}$ n'est généralement pas injective
  il n'y a pas de représentation canonique de $\hat{\gamma}(A^\sharp)$
- tester $\hat{\gamma}(A^\sharp) = \hat{\gamma}(B^\sharp)$ ou $\hat{\gamma}(A^\sharp) \subseteq \hat{\gamma}(B^\sharp)$ est généralement très difficile !

$\Longrightarrow$ nous utilisons une solution approchée.

Exemple : ensemble des parties sur le domaine des intervalles



$A^\sharp$ $B^\sharp$ $C^\sharp$

$A^\sharp = \{\{0\} \times \{0\}, [0,1] \times \{1\}\}$

$B^\sharp = \{\{0\} \times \{0\}, \{0\} \times \{1\}, \{1\} \times \{1\}\}$

$C^\sharp = \{\{0\} \times [0,1], [0,1] \times \{1\}\}$

$\hat{\gamma}(A^\sharp) = \hat{\gamma}(B^\sharp) = \hat{\gamma}(C^\sharp)$

$B^\sharp$ est plus coûteux à représenter : il nécessite trois éléments abstraits au lieu de deux ;
$C^\sharp$ est un recouvrement et non une partition (rouge $\cap$ bleu $= \{0\} \times \{1\} \neq \emptyset$)

Ordre (suite)

Solution : abstraction sûre de $\subseteq$

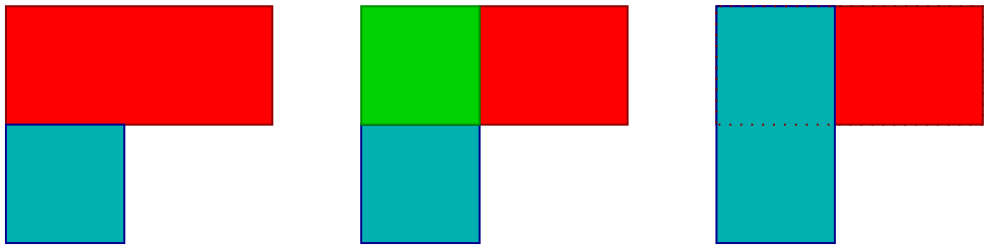
$A^\sharp \mathbin{\hat{\sqsubseteq}} B^\sharp \overset{\text{def}}{\iff} \forall X^\sharp \in A^\sharp : \exists Y^\sharp \in B^\sharp : X^\sharp \sqsubseteq Y^\sharp$

tout élément de $A^\sharp$ est inclus dans un élément de $B^\sharp$
relation de Hoare

- $\hat{\sqsubseteq}$ est un ordre partiel, en supposant que $\sqsubseteq$ en soit un
- $\hat{\sqsubseteq}$ est une approximation de $\subseteq$, si $\sqsubseteq$ en est aussi une
  $A^\sharp \mathbin{\hat{\sqsubseteq}} B^\sharp \implies \hat{\gamma}(A^\sharp) \subseteq \hat{\gamma}(B^\sharp)$ mais la réciproque n'est pas forcément vraie !
- tester $\hat{\sqsubseteq}$ se réduit à tester $\sqsubseteq$ un nombre fini de fois

$\implies$ très efficace !

Exemple : ensemble des parties du domaine des intervalles



$A^\sharp$ $B^\sharp$ $C^\sharp$

$\hat{\gamma}(A^\sharp) = \hat{\gamma}(B^\sharp) = \hat{\gamma}(C^\sharp)$

$B^\sharp \mathbin{\hat{\sqsubseteq}} A^\sharp \mathbin{\hat{\sqsubseteq}} C^\sharp$

Opérations abstraites

Opérations abstraites :

- $\hat{\mathrm{S}}^\sharp[\![\, stat \,]\!]\, A^\sharp \stackrel{\text{def}}{=} \{\, \mathrm{S}^\sharp[\![\, stat \,]\!]\, X^\sharp \mid X^\sharp \in A^\sharp \,\}$

  appliquer $\mathrm{S}^\sharp[\![\, stat \,]\!]$ indépendemment sur chaque élément de $\mathcal{E}^\sharp$

- $A^\sharp \mathbin{\hat{\cup}^\sharp} B^\sharp \stackrel{\text{def}}{=} A^\sharp \cup B^\sharp$

  garde intacts les éléments abstraits de chaque argument
  sans appliquer $\cup^\sharp$, donc sans perte de précision
  $\hat{\cup}^\sharp$ est exact !

- $A^\sharp \mathbin{\hat{\cap}^\sharp} B^\sharp \stackrel{\text{def}}{=} \{\, X^\sharp \cap^\sharp Y^\sharp \mid X^\sharp \in A^\sharp,\ Y^\sharp \in B^\sharp \,\}$

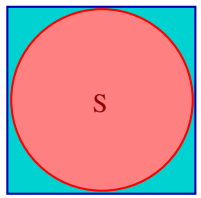
  $\hat{\cap}^\sharp$ est toujours exact si $\cap^\sharp$ l'est
  (car $\cap$ distribue $\cup$ dans le concret)

Correspondance de Galois :

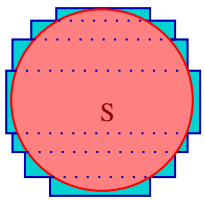
en général, il n'y a pas de fonction d'abstraction $\hat{\alpha}$ correspondant à $\hat{\gamma}$

Exemple : ensemble des parties $\hat{\mathcal{E}}^\sharp$ sur le domaine des intervalles $\mathcal{E}^\sharp$

étant donné le disque $S \stackrel{\text{def}}{=} \{\, (x, y) \mid x^2 + y^2 \leq 1 \,\}$
$\alpha(S) = [-1, 1] \times [-1, 1]$ (abstraction optimale dans les intervalles)
mais il n'y pas de meilleur abstraction dans $\hat{\mathcal{E}}^\sharp$



$\alpha(S)$



pas de $\hat{\alpha}(S)$

Exemple d'analyse

Exemple

$X \leftarrow \textbf{rand}(10, 20)$;
$Y \leftarrow \textbf{rand}(0, 1)$;
• **if** $Y > 0$ **then** $X \leftarrow -X$;
• $Z \leftarrow 100/X$

Analyse :

- $\mathcal{E}^\sharp$ est le domaine des intervalles
- en •, l'élément abstrait est $\{(X \in [10, 20], Y \in [0, 1])\}$
- lors du test $\hat{S}^\sharp[\![\textbf{if } Y > 0 \textbf{ then } X \leftarrow -X]\!]$:
  - la branche **then** donne $\{(X \in [-20, -10], Y \in [1, 1])\}$
  - la branche **else** donne $\{(X \in [10, 20], Y \in [0, 0])\}$
  - l'union en • donne :
    $$X^\sharp \overset{\text{def}}{=} \{(X \in [-20, -10], Y \in [1, 1])\} \hat{\cup} \{(X \in [10, 20], Y \in [0, 0])\}$$
    $$= \{(X \in [-20, -10], Y \in [1, 1]), (X \in [10, 20], Y \in [0, 0])\}$$
- ni $X \in [10, 20]$, ni $X \in [-20, -10]$ ne contient 0
  $\Longrightarrow \hat{S}^\sharp[\![Z \leftarrow 100/X]\!] X^\sharp$ n'a pas de division par 0

Élargissement

Problème : le itérés $(A_n^\sharp)_{n\in\mathbb{N}}$ dans les boucles ne convergent pas forcément

- le nombre d'éléments dans $A_n^\sharp$ peut croître de manière arbitraire ;
- même si $|A_n^\sharp|$ est stable, les éléments de $A_n^\sharp$ ne convergent pas forcément ;
  dans le cas où $\mathcal{E}^\sharp$ a des séquences infinies strictement croissantes

$\Longrightarrow$ un élargissement $\triangledown$ est nécessaire.

La construction d'élargissements pour les domaines d'ensembles des parties est un problème difficile.

Exemple naïf d'élargissement :

fusionner les éléments abstraits à partir d'un nombre fixé d'itérations

$$A_{n+1}^\sharp \overset{\text{def}}{=} \begin{cases} A_n^\sharp \mathbin{\hat{\cup}^\sharp} B_{n+1}^\sharp & \text{si } n < N \\ \{(\cup^\sharp A_n^\sharp) \triangledown (\cup^\sharp B_{n+1}^\sharp)\} & \text{sinon} \end{cases}$$

Note : d'autres élargissements plus sophistiqués existent.

Partitionnement d'états

Partitionnement d'états

Principe :

- partitionner *a priori* $\mathcal{E}$ en un nombre fini d'ensembles
- abstraire les états contenus dans chaque partie de manière indépendante par un élément abstrait de $\mathcal{E}^\sharp$

Domaine abstrait :

Étant donné un domaine abstrait $\mathcal{E}^\sharp$,
nous choisissons une partition de $\mathcal{E}$ représentable dans $\mathcal{E}^\sharp$;
i.e., un ensemble $P^\sharp \subseteq \mathcal{E}^\sharp$ tel que :

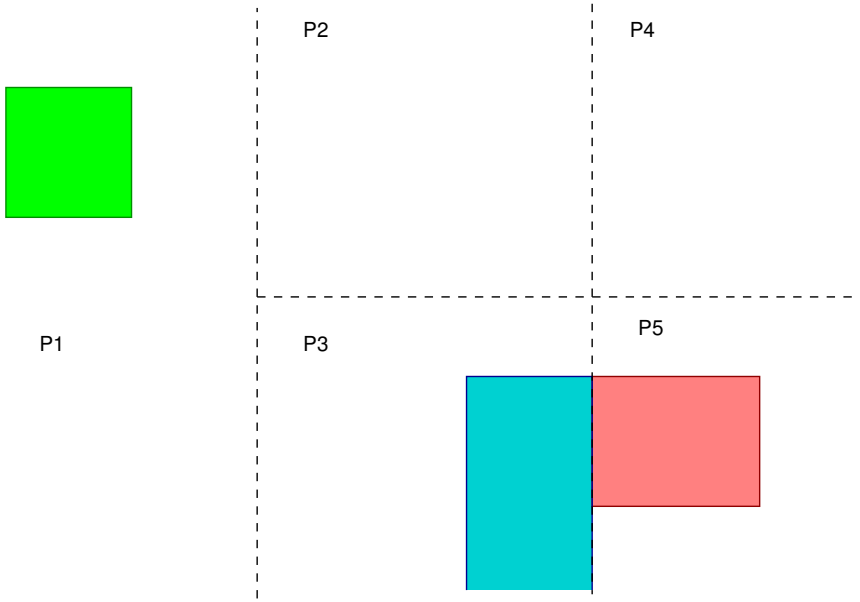
- $P^\sharp$ est fini
- $\cup \{ \gamma(X^\sharp) \mid X^\sharp \in P^\sharp \} = \mathcal{E}$

  Note : il peut s'agir en réalité d'un recouvrement de $\mathcal{E}$ et non d'une partition ;
  nous autorisons : $X^\sharp \neq Y^\sharp \in P^\sharp$ avec $\gamma(X^\sharp) \cap \gamma(Y^\sharp) \neq \emptyset$
  cela permet de choisir $\mathcal{E}^\sharp$ de manière arbitraire

Le domaine partitionné est alors : $\tilde{\mathcal{E}}^\sharp \stackrel{\text{def}}{=} P^\sharp \to \mathcal{E}^\sharp$

représentable en mémoire car $P^\sharp$ est fini

Structure d'ordre



Exemple : ici, $\mathcal{E}^\sharp$ est le domaine des intervalles

$P^\sharp = \{P_1, P_2, P_3, P_4, P_5\}$ où
$P_1 = [-\infty, 0] \times [-\infty, +\infty]$
$P_2 = [0, 10] \times [0, +\infty]$
$P_3 = [0, 10] \times [-\infty, 0]$
$P_4 = [10, +\infty] \times [0, +\infty]$
$P_5 = [10, +\infty] \times [-\infty, 0]$

$X^\sharp = [P_1 \mapsto [-6, -5] \times [5, 6],$
$P_2 \mapsto \bot,$
$P_3 \mapsto [9, 10] \times [-\infty, -1],$
$P_4 \mapsto \bot,$
$P_5 \mapsto [10, 12] \times [-3, -1]]$

- $\tilde{\mathcal{E}}^\sharp \stackrel{\text{def}}{=} P^\sharp \to \mathcal{E}^\sharp$
  à chaque élément de la partition est associé un élément abstrait
- $\tilde{\gamma}(A^\sharp) \stackrel{\text{def}}{=} \cup \{ \gamma(A^\sharp(X^\sharp)) \cap \gamma(X^\sharp) \mid X^\sharp \in P^\sharp \}$
  chaque $A^\sharp(X^\sharp)$ représente un ensemble d'environnements inclus dans $\gamma(X^\sharp)$
- $A^\sharp \stackrel{\tilde{}}{\sqsubseteq} B^\sharp \iff^{\text{def}} \forall X^\sharp \in P^\sharp : A^\sharp(X^\sharp) \sqsubseteq B^\sharp(X^\sharp)$
  ordre point à point
- $\tilde{\alpha}(S) \stackrel{\text{def}}{=} \lambda X^\sharp \in P^\sharp . \alpha(S \cap \gamma(X^\sharp))$
  nous abstrayons de manière indépendante dans chaque $\gamma(A^\sharp)$, $A^\sharp \in P^\sharp$
  nous avons une correspondance de Galois pour $\tilde{\mathcal{E}}^\sharp$, si il en existe une pour $\mathcal{E}^\sharp$

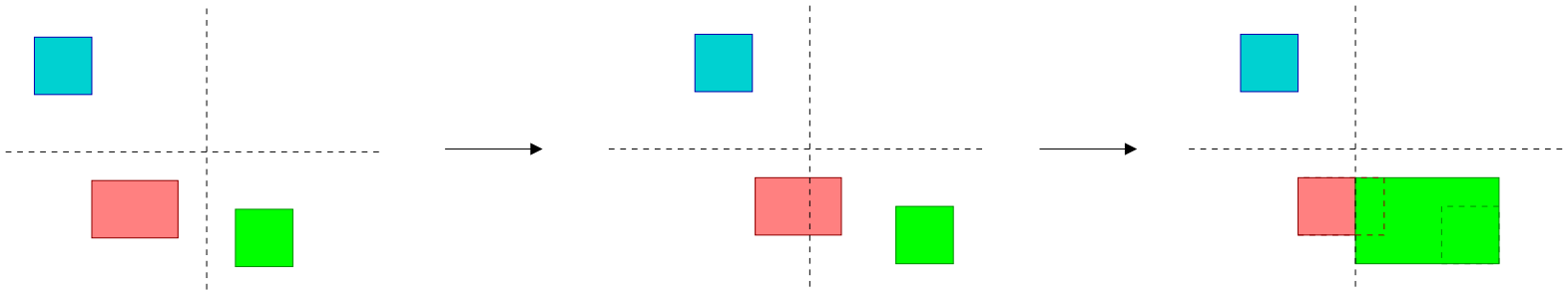
Opérateurs abstraits

Opérateurs abstraits : extension point à point de $\mathcal{E}^\sharp$ en $P^\sharp \to \mathcal{E}^\sharp$

- $A \mathbin{\tilde{\cup}^\sharp} B \stackrel{\text{def}}{=} \lambda X^\sharp \in P^\sharp . A(X^\sharp) \cup^\sharp B(X^\sharp)$
- $A \mathbin{\tilde{\cap}^\sharp} B \stackrel{\text{def}}{=} \lambda X^\sharp \in P^\sharp . A(X^\sharp) \cap^\sharp B(X^\sharp)$
- $A \mathbin{\tilde{\triangledown}} B \stackrel{\text{def}}{=} \lambda X^\sharp \in P^\sharp . A(X^\sharp) \triangledown B(X^\sharp)$
- $\tilde{C}^\sharp [\![e_1 \bowtie e_2]\!] A^\sharp \stackrel{\text{def}}{=} \lambda X^\sharp \in P^\sharp . C^\sharp [\![e_1 \bowtie e_2]\!] A^\sharp(X^\sharp)$
- $\tilde{S}^\sharp [\![V \leftarrow e]\!] A^\sharp$ est plus complexe

  une image $S^\sharp [\![V \leftarrow e]\!] A^\sharp(X^\sharp)$ peut échapper de sa partition originale $X^\sharp$;
  nous devons couper le résultat abstrait selon les bords des partitions,
  puis fusionner tous les morceaux qui tombent dans la même partition

  Exemple : $X \leftarrow X + 2$



$$\tilde{S}^\sharp [\![V \leftarrow e]\!] A^\sharp \stackrel{\text{def}}{=} \lambda X^\sharp . \cup^\sharp \{ X^\sharp \cap^\sharp S^\sharp [\![V \leftarrow e]\!] A(Y^\sharp) \mid Y^\sharp \in P^\sharp \}$$

Exemple d'analyse

Exemple

$X \leftarrow \textbf{rand}(10, 20);$
$Y \leftarrow \textbf{rand}(0, 1);$
if $Y > 0$ **then** $X \leftarrow -X;$
• $Z \leftarrow 100/X$

Analyse :

- $\mathcal{E}^\sharp$ est le domaine des intervalles
- nous partitionnons par rapport au signe de X
  $P^\sharp \stackrel{\text{def}}{=} \{ X^+, X^- \}$ où
  $X^+ \stackrel{\text{def}}{=} [0, +\infty] \times \mathbb{Z} \times \mathbb{Z}$ et $X^- \stackrel{\text{def}}{=} [-\infty, 0] \times \mathbb{Z} \times \mathbb{Z}$
- en •, nous trouvons :
  $X^+ \mapsto [X \in [10, 20], Y \mapsto [0, 0], Z \mapsto [0, 0]]$
  $X^- \mapsto [X \in [-20, -10], Y \mapsto [1, 1], Z \mapsto [0, 0]]$

  $\Longrightarrow$ pas de division par zéro

Les analyses par ensemble des parties et par partitionnement d'états donnent des résultats similaires, mais les domaines ont des structures bien différentes : $\mathcal{P}(\mathcal{E}^\sharp)$ et $P^\sharp \to \mathcal{E}^\sharp$.

Partitionnement de traces

Sensibilité au chemin d'exécution

Principe : partitionner vis à vis de l'histoire des calculs

- garder séparés les éléments abstraits issus de chemins d'exécution différents

  e.g., branches différentes prises, nombre d'itérations de boucle différents, etc.

- éviter de fusionner avec $\cup^\sharp$ des éléments :
  - à la fin d'un **if** $\cdots$ **then** $\cdots$ **else**
  - à la fin d'une itération de boucle

  car l'histoire des calculs est différente

Intuition : vision par transformation de programme

$X \leftarrow$ **rand**$(-50, 50)$;
if $X \geq 0$ **then**
  $Y \leftarrow X + 10$
else
  $Y \leftarrow X - 10$;
assert $Y \neq 0$

$\longrightarrow$

$X \leftarrow$ **rand**$(-50, 50)$;
if $X \geq 0$ **then**
  $Y \leftarrow X + 10$;
  assert $Y \neq 0$
else
  $Y \leftarrow X - 10$;
  assert $Y \neq 0$

l'**assert** n'est plus testé après la fusion des deux branches, mais dans le contexte de chaque branche
$\Longrightarrow$ amélioration de la précision

le domaine des intervalles échoue à prouver l'assertion à gauche, mais réussit à droite

Domaine abstrait

Formalisation comme domaine abstrait : restreint au cas des tests **if** $\cdots$ **then** $\cdots$ **else**

- $\mathcal{L}$ dénote un ensemble de points syntaxiques du programme correspondants aux instructions ℓ : **if** $\cdots$ **then** $\cdots$ **else**, où $\ell \in \mathcal{L}$
- l'histoire abstraite est un élément de $\mathbb{H} \stackrel{\text{def}}{=} \mathcal{L} \to \{\text{true}, \text{false}, \bot\}$

  $H \in \mathbb{H}$ se souvient, pour chaque **if** $\cdots$ **then** $\cdots$ **else**
  si nous avons exécuté la branche **then** ou **else**
  lors de la dernière exécution du test
  - $H(\ell) = \text{true}$: nous avons pris la branche **then**
  - $H(\ell) = \text{false}$: nous avons pris la branche **else**
  - $H(\ell) = \bot$: nous n'avons jamais exécuté ce test

  Notes :
  - $\mathbb{H}$ se souvient de l'exécution de plusieurs tests successifs ℓ_1 : **if** $\cdots$ **then** $\cdots$ **else**; ℓ_2 : **if** $\cdots$ **then** $\cdots$ **else**
  - si le test apparaît dans une boucle, $\mathbb{H}$ se souvient uniquement du dernier tour de boucle **while** $\cdots$ **do** ℓ : **if** $\cdots$ **then** $\cdots$ **else**
  - $\mathbb{H}$ peut être étendu à des histoires de calcul plus longues par $\mathbb{H} = (\mathcal{L} \to \{\text{true}, \text{false}, \bot\})^*$
  - $\mathbb{H}$ peut être étendu pour distinguer les itérations de boucle par $\mathbb{H} = \mathcal{L} \to \mathbb{N}$
- $\check{\mathcal{E}}^\sharp \stackrel{\text{def}}{=} \mathbb{H} \to \mathcal{E}^\sharp$

  un élément abstrait différent est associé à chaque histoire abstraite différente

Opérateurs abstraits

- $\check{\mathcal{E}}^\sharp \stackrel{\text{def}}{=} \mathbb{H} \to \mathcal{E}^\sharp$
- $\check{\gamma}(A^\sharp) = \cup \{ \gamma(A^\sharp(H)) \mid H \in \mathbb{H} \}$
- $\check{\sqsubseteq}$, $\check{\cup}^\sharp$, $\check{\cap}^\sharp$, $\check{\nabla}$ sont définis point à point
- $\check{S}^\sharp[\![V \leftarrow e]\!]$ et $\check{C}^\sharp[\![e_1 \bowtie e_2]\!]$ sont également point à point
- $\check{S}^\sharp[\![\ell : \textbf{if } c \textbf{ then } s_1 \textbf{ else } s_2]\!] A^\sharp$ est plus complexe
  - nous commençons par fusionner toutes les partitions distinguant ℓ
    $C^\sharp = \lambda H.A^\sharp(H[\ell \mapsto \text{true}]) \cup^\sharp A^\sharp(H[\ell \mapsto \text{false}]) \cup^\sharp A^\sharp(H[\ell \mapsto \bot])$
    en effet, nous ne pouvons nous souvenir que de la dernière exécution de ℓ ; il faut oublier l'exécution précédente de ℓ pour garder l'exécution courante
  - nous calculons la branche **then**, avec $H(\ell) = \text{true}$
    $T'^\sharp = \check{S}^\sharp[\![s_1]\!](\check{C}^\sharp[\![c]\!] T^\sharp)$ où
    $T^\sharp = \lambda H.C^\sharp(H)$ si $H(\ell) = \text{true}$, $\bot$ sinon
  - nous calculons la branche **else**, avec $H(\ell) = \text{false}$
    $F'^\sharp = \check{S}^\sharp[\![s_2]\!](\check{C}^\sharp[\![\neg c]\!] F^\sharp)$ où
    $F^\sharp = \lambda H.C^\sharp(H)$ si $H(\ell) = \text{false}$, $\bot$ sinon
  - nous fusionnons les deux branches : $T'^\sharp \check{\cup}^\sharp F'^\sharp$
    l'union dans $\mathcal{E}^\sharp$ est en réalité exacte
    car $\forall H \in \mathbb{H}$: , soit $T'^\sharp(H) = \bot$, soit $F'^\sharp(H) = \bot$

Exemple d'analyse

Exemple

$$
\begin{array}{rl}
& X \leftarrow \textbf{rand}(10, 20); \\
& Y \leftarrow \textbf{rand}(0, 1); \\
\ell : & \textbf{if } Y > 0 \textbf{ then } X \leftarrow -X; \\
& Z \leftarrow 100/X
\end{array}
$$

Analyse :

- $\mathcal{E}^\sharp$ est le domaine des intervalles
- avant le test, l'élément abstrait est $[\ell \mapsto \bot] \mapsto (X \in [10, 20], Y \in [0, 1])$
- après le test, nous avons :
  $[\ell \mapsto \text{true}] \mapsto (X \in [-20, -10], Y \in [1, 1])$,
  $[\ell \mapsto \text{false}] \mapsto (X \in [10, 20], Y \in [0, 0])]$
- ni $X \in [10, 20]$, ni $X \in [-20, -10]$ ne provoque de division par zéro
  $\Longrightarrow \hat{S}^\sharp[\![Z \leftarrow 100/X]\!] X^\sharp$ n'a pas de division par zéro

Les analyses par partitionnement d'états et par partitionnement de traces donnent des résultats similaires, mais le critère de partitionnement est différent, et ce ne sont donc pas les même opérations qui mettent à jour les partitions : $X \leftarrow e$ pour le partitionnement d'état, et **if then else** pour le partitionnement de traces.

Un exemple plus complexe

Interpolation affine par morceaux

$X \leftarrow \textbf{rand}(TX[0], TX[N])$;
$I \leftarrow 0$;
while $I < N \ \wedge X > TX[I+1]$ **do**
  $I \leftarrow I + 1$;
done;
$Y \leftarrow TY[I] + (X - TX[I]) \times TS[I]$

Sémantique concrète :

interpolation dans une table, par cas selon la valeur de X

- recherche de l'indice I dans la table d'interpolation : $TX[I] \leq X \leq TX[I+1]$
- calcul affine avec pour valeur de base $TY[I]$ quand $X = TX[I]$, et pour pente $TS[I]$

Analyse : dans le domaine des intervalles

- sans partitionnement : $Y \in [\min\ TY, \max\ TY] + (X - [\min\ TX, \max\ TX]) \times [\min\ TS, \max\ TS]$
- avec partitionnement vis à vis du nombre d'itérations de boucle : $Y \in \cup_{I \in [0,N]}\ TY[I] + ([0, TX[I+1] - TX[I]) \times TS[I]$
  plus précis car nous gardons la relation entre les indices dans la table

Conseils d'implantation OCaml pour le projet

Produit réduit

But :

- définir un mécanisme de réduction générique pour tout `VALUE_DOMAIN`,
- appliquer à la réduction entre intervalles et parités.

Note : nous raisonnons dans les domaines de valeurs $\mathcal{P}(\mathbb{Z})$,
avant l'extension aux domaines d'environnements non-relationnels dans $\mathcal{P}(\mathcal{E})$!

Étapes :

- programmer le module `Parity` de signature `VALUE_DOMAIN` ;
  (facile !)
- définir la signature des réductions `REDUCTION`, contenant :
  - deux domaines de signature `VALUE_DOMAIN` ;
  - un opérateur de réduction entre les domaines ;
- définir un foncteur général de produit réduit :
  `ReducedProduct` : `REDUCTION` $\rightarrow$ `VALUE_DOMAIN` ;
- définir un module `ParityIntervalReducion` de signature `REDUCTION`
  contenant la réduction entre parités et intervalles ;
- tout brancher ensemble pour obtenir l'analyse demandée.

Produit réduit : signature des réductions

domains/value_reduction.ml

```
module type VALUE_REDUCTION =
sig

  module A : VALUE_DOMAIN
  module B : VALUE_DOMAIN

  type t = A.t * B.t

  val reduce: t -> t

end
```

Produit réduit : instance de réduction

domains/parity_interval_reduction.ml

```
module ParityIntervalsReduction =
(struct

  module A = Parity
  module B = Intervals

  type t = A.t * B.t

  let reduce ((a,b):t) : t =
    ...

end : VALUE_REDUCTION)
```

Produit réduit : produit réduit générique

domains/value_reduced_product.ml

```
module ReducedProduct(R : VALUE_REDUCTION) =
(struct

  module A = R.A
  module B = R.B

  type t = R.t (* A.t * B.t *)

  let top = ...

  let unary ((a,b):t) (op:int_unary_op) : t =
    R.reduce ...

  ...

end : VALUE_DOMAIN)
```