Sémantique et applications à la vérification
Examen (durée : 2h) — 28 mai 2014

Nous allons étudier les propriétés d'un langage pour machines à piles (similaire au langage RPL rencontré dans les calculatrices HP), et mettre au point une analyse statique simple pour ce langage, afin de s'assurer de la bonne utilisation de la pile.

Un état valide est une *pile* de valeurs entières ; on note l'*état d'erreur* Ω, l'ensemble des piles $\mathbb{P}$, et l'ensemble $\mathbb{P}_\Omega$ des états, correspondants soit à une pile soit à un état d'erreur ($\mathbb{P}_\Omega = \mathbb{P} \uplus \{\Omega\}$) :

p	::=	ϵ	pile vide
	|	$n : p$	où $n \in \mathbb{N}$
s	::=	p	une pile valide (vide ou non)
	|	Ω	état d'erreur

On note $p_0 \cdot p_1$ la concaténation des piles p_0 et p_1 et $\bar{p}$ la longueur de la pile p (nombre d'éléments).

1 Instructions de base et sémantique

On considère pour commencer les *instructions* suivantes :

i	::=	$\langle n \rangle$	ajoute n sur la pile
	|	$\ominus$	opposé de l'élément au sommet de la pile
	|	$\boxplus$	somme des deux éléments au sommet de la pile
	|	$\boxminus$	soustraction des deux éléments au sommet de la pile
	|	$\boxtimes$	produit des deux éléments au sommet de la pile

Par convention, on considérera toujours que le premier élément au sommet de la pile est à profondeur 1. Lorsqu'une instruction qui doit lire des valeurs jusqu'à la profondeur k dans la pile est exécutée à partir d'une pile qui ne contient pas au moins k éléments, celle-ci plante, et produit Ω. Par conséquent, on peut donner, dans le tableau ci-dessous quelques exemples d'opérations :

état précédent	instruction	état suivant
$1 : 9 : 2 : \epsilon$	$\langle 12 \rangle$	$12 : 1 : 9 : 2 : \epsilon$
$1 : 4 : 8 : \epsilon$	$\ominus$	$(-1) : 4 : 8 : \epsilon$
$1 : 4 : 8 : \epsilon$	$\boxminus$	$3 : 8 : \epsilon$
$8 : \epsilon$	$\boxminus$	Ω

Avant d'aller plus loin, nous allons formaliser le comportement de chaque instruction i à l'aide d'une *relation* $[\![i]\!]$ qui relie états précédents et états suivants : $(p, s) \in [\![i]\!]$ si et seulement l'exécution de i à partir de la pile p peut nous conduire dans l'état s (noter que l'état précédent est nécessairement une pile p —pas de calcul possible à partir de Ω—, tandis que l'état suivant peut être soit une pile soit l'état d'erreur Ω —d'où la notation s pour l'état destination). Cette sémantique est proche d'une sémantique dénotationnelle (à base de fonctions), mais l'usage de relations simplifiera les parties suivantes.

Par exemple, $[\![\boxplus]\!]$ est définie par :

$$[\![\boxplus]\!] = \{(n_0 : n_1 : p, (n_0 + n_1) : p) \mid n_0, n_1 \in \mathbb{Z}, p \in \mathbb{P}\} \cup \{(n : \epsilon, \Omega) \mid n \in \mathbb{Z}\} \cup \{(\epsilon, \Omega)\}$$

Cela signifie que les comportements suivants peuvent être observés :

- calcul sans erreur à partir d'une pile comportant au moins deux éléments n_0, n_1, ce qui correspond à l'étape de calcul $n_0 : n_1 : p \to (n_0 + n_1) : p$;
- erreur d'évaluation à partir d'une pile vide ($\epsilon \to \Omega$) ou ne contenant qu'un élément ($(n : \epsilon) \to \Omega$).

Question 1 Sémantique des instructions arithmétiques.

Formaliser la sémantique des autres instructions arithmétiques ($\langle n \rangle, \ominus, \boxminus, \boxtimes$).

Corrigé 1

$$
\begin{array}{rcl}
[\![\langle n \rangle]\!] & = & \{(p, n : p) \mid p \in \mathbb{P}\} \\
[\![\ominus]\!] & = & \{(n_0 : p, (-n_0) : p) \mid n_0 \in \mathbb{Z}, p \in \mathbb{P}\} \cup \{(\epsilon, \Omega)\} \\
[\![\boxplus]\!] & = & \{(n_0 : n_1 : p, (n_0 + n_1) : p) \mid n_0, n_1 \in \mathbb{Z}, p \in \mathbb{P}\} \cup \{(n : \epsilon, \Omega) \mid n \in \mathbb{Z}\} \cup \{(\epsilon, \Omega)\} \\
[\![\boxminus]\!] & = & \{(n_0 : n_1 : p, (n_1 - n_0) : p) \mid n_0, n_1 \in \mathbb{Z}, p \in \mathbb{P}\} \cup \{(n : \epsilon, \Omega) \mid n \in \mathbb{Z}\} \cup \{(\epsilon, \Omega)\} \\
[\![\boxtimes]\!] & = & \{(n_0 : n_1 : p, (n_0 n_1) : p) \mid n_0, n_1 \in \mathbb{Z}, p \in \mathbb{P}\} \cup \{(n : \epsilon, \Omega) \mid n \in \mathbb{Z}\} \cup \{(\epsilon, \Omega)\}
\end{array}
$$

Il est parfois utile d'effectuer des opérations diverses sur la pile, pour modifier l'ordre des éléments qui y sont stockés, en dupliquer ou en supprimer... De même que RPL, notre language fournit des instructions spécifiques pour ce faire :

i	::=	...	toutes les instructions vues précédemment
	|	**depth**	profondeur de la pile
	|	**dup**	duplication de l'élément au sommet de la pile
	|	**drop**	suppression de l'élément au sommet de la pile
	|	**pick**	lit (et retire) la valeur n au sommet de la pile puis recopie au sommet de la pile l'élément à profondeur n
	|	**roll**	lit (et retire) la valeur n au sommet de la pile puis déplace au sommet de la pile l'élément à profondeur n (cela revient à faire une sorte de rotation)
	|	**swap**	permute les deux éléments au sommet de la pile

On peut illustrer ces instructions à l'aide des exemples suivants :

état précédent	instruction	état suivant
$1 : 4 : 8 : \epsilon$	**depth**	$3 : 1 : 4 : 8 : \epsilon$
$1 : 4 : 8 : \epsilon$	**dup**	$1 : 1 : 4 : 8 : \epsilon$
$1 : 4 : 8 : \epsilon$	**drop**	$4 : 8 : \epsilon$
ϵ	**drop**	Ω
$3 : 2 : 4 : 6 : 8$	**pick**	$6 : 2 : 4 : 6 : 8$
$3 : 2 : 4 : 6 : 8$	**roll**	$6 : 2 : 4 : 8$
$3 : 2 : 4 : 6 : 8$	**swap**	$2 : 3 : 4 : 6 : 8$

On peut étendre la sémantique définie plus haut à ces instructions. Ainsi, dans le cas de l'instruction **dup**, on a :

$$[\![\textbf{dup}]\!] = \{(n : p, n : n : p) \mid n \in \mathbb{Z}, p \in \mathbb{P}\} \cup \{(\epsilon, \Omega)\}$$

Question 2 Sémantique des instructions de manipulation de la pile.

*Formaliser la sémantique des autres instructions de la pile (***depth**, **drop**, **pick**, **roll**, **swap***).*

Corrigé 2

$$\begin{aligned}
[\![\mathbf{depth}]\!] &= \{(p, \bar{p} : p) \mid p \in \mathbb{P}\} \\
[\![\mathbf{drop}]\!] &= \{(n_0 : p, p) \mid n_0 \in \mathbb{Z}, p \in \mathbb{P}\} \cup \{(\epsilon, \Omega)\} \\
[\![\mathbf{dup}]\!] &= \{(n : p, n : n : p) \mid n_0, n_1 \in \mathbb{Z}, p \in \mathbb{P}\} \cup \{(\epsilon, \Omega)\} \\
[\![\mathbf{pick}]\!] &= \{(n : n_1 : \ldots : n_k : p, n_k : n_1 : \ldots : n_k : p) \mid n, n_1, \ldots, n_k \in \mathbb{Z}, p \in \mathbb{P}\} \\
&\quad \cup \{(n : p, \Omega) \mid n \in \mathbb{Z} \wedge \bar{p} < n \vee n < 0\} \cup \{(\epsilon, \Omega)\} \\
[\![\mathbf{roll}]\!] &= \{(n : n_1 : \ldots : n_k : p, n_k : n_1 : \ldots : n_{k-1} : p) \mid n, n_1, \ldots, n_k \in \mathbb{Z}, p \in \mathbb{P}\} \\
&\quad \cup \{(n : p, \Omega) \mid n \in \mathbb{Z} \wedge \bar{p} < n \vee n < 0\} \cup \{(\epsilon, \Omega)\} \\
[\![\mathbf{swap}]\!] &= \{(n_0 : n_1 : p, n_1 : n_0 : p) \mid n_0, n_1 \in \mathbb{Z}, p \in \mathbb{P}\} \cup \{(n : \epsilon, \Omega) \mid n \in \mathbb{Z}\} \cup \{(\epsilon, \Omega)\}
\end{aligned}$$

2 Fonctions (ou suite d'instructions)

Une *fonction* f est décrite par une suite d'instructions :

$$\begin{array}{rcll} f & ::= & \cdot & \text{suite vide} \\ & | & i\ f & \end{array}$$

Question 3 Sémantique d'une suite d'instructions.

Définir la sémantique des suites d'instructions par induction sur la syntaxe. On pourra utiliser l'opérateur de composition des relations ci-dessous et on justifiera ce choix :

$$\begin{array}{l} \forall R_0, R_1 \in \mathcal{P}(\mathbb{P} \times \mathbb{P}_\Omega), \\ \quad R_0 \circledcirc R_1 \quad ::= \quad \{(p_0, s_2) \in \mathbb{P} \times \mathbb{P}_\Omega \mid \exists p_1 \in \mathbb{P}, (p_0, p_1) \in R_0 \wedge (p_1, s_2) \in R_1\} \\ \qquad\qquad\qquad\quad \cup\ (R_0 \cap \{(p, \Omega) \mid p \in \mathbb{P}\}) \end{array}$$

(noter que, conformément à la convention adoptée pour notre sémantique sous forme de relation, l'état initial est toujours une pile —notée p, p_0, p_1—, et l'état final est un état pouvant être soit une pile soit l'état d'erreur —noté s_2)

Corrigé 3 *Une suite vide ne modifie pas l'état de la pile :* $[\![\cdot]\!] = \{(p, p) \mid p \in \mathbb{P}\}$.
Une suite $i\ f$ commence par exécuter i ; dans le cas où on ne rencontre pas alors un état d'erreur à ce stade, on exécute ensuite le reste de la suite d'instructions ; si i renvoie un état d'erreur, l'exécution de la suite complète renvoie un état d'erreur. Par conséquent :

$$\begin{aligned}
[\![i\ f]\!] &= \{(p_0, s_2) \in \mathbb{P} \times \mathbb{P}_\Omega \mid \exists p_1 \in \mathbb{P}, (p_0, p_1) \in [\![i]\!] \wedge (p_1, s_2) \in [\![f]\!]\} \\
&\quad \cup\ ([\![i]\!] \cap \{(p, \Omega) \mid p \in \mathbb{P}\}) \\
&= [\![i]\!] \circledcirc [\![f]\!]
\end{aligned}$$

Question 4 Exemples.

Donner la sémantique des suites d'instructions suivantes (on pourra traiter les cas d'erreur séparément et donner les étapes du calcul dans le cas où la pile initiale contient suffisament d'éléments) :

- $f_0 ::= 8\ \boxplus$
- $f_1 ::= 3\ \boxtimes\ \mathbf{swap}\ 2\ \boxtimes\ 5$
- $f_2 ::= \mathbf{dup}\ 3\ \mathbf{pick}\ \boxminus\ \mathbf{swap}\ 3\ \mathbf{pick}\ \boxplus\ \boxtimes\ \mathbf{swap}\ \mathbf{drop}$

Corrigé 4 *Pour f_0 : la première instruction ajoute 8 sur la pile, la seconde dépile les deux premières valeurs (donc, 8, ainsi que la valeur qui était au sommet de la pile) et les ajoute ; par conséquent f_0 plante si et seulement si la pile était vide ; sinon, f_0 fait passer d'un état $n : p$ à un état $(n + 8) : p$. On obtient ainsi :*

- $[\![f_0]\!] = \{(n : p, (n+8) : p) \mid p \in \mathbb{P}\} \cup \{(\epsilon, \Omega)\}$
- $[\![f_1]\!] = \{(n_0 : n_1 : p, 5 : 2n_1 : 3n_0 : p) \mid n_0, n_1 \in \mathbb{Z}, p \in \mathbb{P}\} \cup \{(n : \epsilon, \Omega) \mid n \in \mathbb{Z}\} \cup \{(\epsilon, \Omega)\}$
- $[\![f_2]\!] = \{(n_0 : n_1 : p, (n_0 - n_1)(n_0 + n_1) : p) \mid n_0, n_1 \in \mathbb{Z}, p \in \mathbb{P}\} \cup \{(n : \epsilon, \Omega) \mid n \in \mathbb{Z}\} \cup \{(\epsilon, \Omega)\}$

Question 5 Quelques propriétés de la sémantique.

Montrer qu'on a les propriétés suivantes pour toute suite d'instructions f (on commencera par justifier ces propriétés dans le cas des instructions de base) :

- $\forall p_0, p_1, p' \in \mathbb{P},\ (p_0, p_1) \in [\![f]\!] \Longrightarrow (p_0 \cdot p', p_1 \cdot p') \in [\![f]\!]$
- $\forall p_0, p' \in \mathbb{P},\ (p_0 \cdot p', \Omega) \in [\![f]\!] \Longrightarrow (p_0, \Omega) \in [\![f]\!]$

On remarquera que ces propriétés sont vérifiées dans les exemples précédents.

Corrigé 5 *On procéde par induction sur la longueur des suites d'instructions. La sémantique de chaque instruction de base (instruction arithmétique ou instruction de pile) satisfait ces deux propriétés. De même pour la liste vide. Pour traiter le cas de la récurrence, il suffit de remarquer que $\circledcirc$ préserve chacune des deux propriétés.*

3 Abstraction de la taille de la pile

Nous avons pû constater que l'exécution d'une suite d'instructions peut se terminer par une erreur, lorsqu'il n'y a pas assez d'éléments sur la pile et nous allons donc proposer une analyse statique qui va permettre de déterminer dans quels cas on peut garantir qu'une suite d'instructions pourra s'exécuter sans planter. Nous nous intéressons tout d'abord à l'abstraction que nous allons utiliser.

Chaque instruction et chaque suite d'instructions sera abstraite par une paire (D, E) où :

- $D \subseteq \mathcal{P}(\mathbb{Z})$ décrit un sur-ensemble des variations de la taille de la pile ;
- $E \subseteq \mathcal{P}(\mathbb{N})$ représente un sur-ensemble des tailles des piles qui provoquent une erreur.

La composante D joue un rôle crucial dans l'analyse : en effet, si on considère la suite d'instruction $i_0; i_1$, et si l'on sait par exemple que i_1 plante si elle ne trouve pas au moins 2 éléments sur la pile, il faut savoir de combien la taille de celle-ci varie lors de l'exécution de i_0 pour savoir combien d'éléments doit on trouver sur la pile au début de l'exécution du programme pour être sûr qu'aucune erreur n'interviendra. Autrement dit, si nous n'avions pas de composante D, nous ne pourrions pas élaborer une analyse statique calculant avec précision la composante E.

Formellement, nous définissons :

- $\mathbb{A} = \mathcal{P}(\mathbb{Z}) \times \mathcal{P}(\mathbb{N})$
- $\sqsubseteq$, inclusion point à point sur $\mathbb{A}$: $(D_0, E_0) \sqsubseteq (D_1, E_1) \iff D_0 \subseteq D_1 \wedge E_0 \subseteq E_1$
- $\alpha : (\mathbb{P} \times \mathbb{P}_\Omega) \to \mathbb{A}$, telle que $\alpha(R) = (\{\bar{p_1} - \bar{p_0} \mid (p_0, p_1) \in R \cap \mathbb{P} \times \mathbb{P}\}, \{\bar{p} \mid (p, \Omega) \in R\})$

Question 6 Domaine abstrait.

Définir l'opérateur de borne supérieure $\sqcup$ sur $\mathbb{A}$.

Corrigé 6 *L'opérateur de borne supérieure $\sqcup$ est défini par :* $(D_0, E_0) \sqcup (D_1, E_1) = (D_0 \cup D_1, E_0 \cup E_1)$.

Question 7 Correspondance de Galois.

Exhiber une fonction de concrétisation $\gamma : \mathbb{A} \to \mathcal{P}(\mathbb{P} \times \mathbb{P}_\Omega)$ de manière à former une correspondance de Galois :

$$(\mathcal{P}(\mathbb{P} \times \mathbb{P}_\Omega), \subseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (\mathbb{A}, \sqsubseteq)$$

En déduire que α définit une fonction d'abstraction.

Corrigé 7 *Cherchons une fonction de concrétisation γ telle que $\forall R \subseteq \mathbb{P} \times \mathbb{P}_\Omega, (D,E) \in \mathbb{A},\ \alpha(R) \sqsubseteq (D,E) \iff R \subseteq \gamma(D,E)$:*

$$\begin{array}{ll} & \alpha(R) \sqsubseteq (D,E) \\ \iff & (\{\bar{p_1} - \bar{p_0} \mid (p_0,p_1) \in R \cap \mathbb{P} \times \mathbb{P}\}, \{\bar{p} \mid (p,\Omega) \in R\}) \sqsubseteq (D,E) \\ \iff & \left\{ \begin{array}{ll} & \{\bar{p_1} - \bar{p_0} \mid (p_0,p_1) \in R \cap \mathbb{P} \times \mathbb{P}\} \subseteq D \\ \wedge & \{\bar{p} \mid (p,\Omega) \in R\} \subseteq E \end{array} \right. \\ \iff & R \subseteq \{(p_1,p_0) \in \mathbb{P}^2 \mid \bar{p_1} - \bar{p_0} \in D\} \cup \{(p,\Omega) \mid \bar{p} \in E\} \end{array}$$

Nous avons donc une correspondance de Galois, en utilisant la fonction de concrétisation ci-dessous :

$$\begin{array}{rccl} \gamma : & \mathbb{A} & \longrightarrow & \mathcal{P}(\mathbb{P} \times \mathbb{P}_\Omega) \\ & (D,E) & \longmapsto & \{(p_1,p_0) \in \mathbb{P}^2 \mid \bar{p_1} - \bar{p_0} \in D\} \cup \{(p,\Omega) \mid \bar{p} \in E\} \end{array}$$

Question 8 Exemples d'abstractions.

Pour chacune des suites d'instructions f_0, f_1, f_2 que nous avons introduites à la question 4, donner la meilleure abstraction de sa sémantique, en lui appliquant la fonction α (ce n'est pas le résultat d'une analyse statique —qui sera abordée dans la partie suivante— ; il s'agit simplement de voir quelles informations nous pouvons espérer énoncer sur ces exemples).

Corrigé 8 *Il suffit de calculer $\alpha([\![f]\!]$ pour chaque f :*

$$\begin{array}{rcl} \alpha([\![f_0]\!]) & = & (\{-1\},\{0\}) \\ \alpha([\![f_1]\!]) & = & (\{1\},\{0,1\}) \\ \alpha([\![f_2]\!]) & = & (\{-1\},\{0,1\}) \end{array}$$

4 Analyse statique

Nous allons maintenant définir une analyse statique permettant de calculer une approximation du comportement d'une suite d'instructions sans l'exécuter ni calculer à la main sa sémantique.

Question 9 Sémantique abstraite des instructions.

Donner la meilleure abstraction de la sémantique de chacune des instructions, que nous avons définies aux questions 1 et 2 (on pourra noter $[\![i]\!]^\sharp$ la sémantique abstraite de l'instruction i).
À titre d'exemple, et pour l'instruction $\boxplus$ dont nous avions donné la sémantique, la meilleure abstraction est $(\{-1\},\{0,1\})$.

Corrigé 9

$$\begin{array}{rcl} [\![\langle n \rangle]\!]^\sharp & = & (\{1\},\emptyset) \\ [\![\ominus]\!]^\sharp & = & (\{0\},\{0\}) \\ [\![\boxplus]\!]^\sharp & = & (\{-1\},\{0,1\}) \\ [\![\boxminus]\!]^\sharp & = & (\{-1\},\{0,1\}) \\ [\![\boxtimes]\!]^\sharp & = & (\{-1\},\{0,1\}) \\ [\![\textbf{depth}]\!]^\sharp & = & (\{1\},\emptyset) \end{array} \qquad \begin{array}{rcl} [\![\textbf{drop}]\!]^\sharp & = & (\{-1\},\{0\}) \\ [\![\textbf{dup}]\!]^\sharp & = & (\{1\},\{0\}) \\ [\![\textbf{pick}]\!]^\sharp & = & (\{0\},\mathbb{N}) \\ [\![\textbf{roll}]\!]^\sharp & = & (\{-1\},\mathbb{N}) \\ [\![\textbf{swap}]\!]^\sharp & = & (\{0\},\{0,1\}) \end{array}$$

Question 10 Approximation de la composition.

Proposer un opérateur abstrait $\circledcirc^\sharp$ qui calcule une sur-approximation de $\circledcirc$, c'est à dire tel que :

$$\forall R_0, R_1 \subseteq \mathbb{P} \times \mathbb{P}_\Omega,\ \alpha(R_0 \circledcirc R_1) \sqsubseteq \alpha(R_0) \circledcirc^\sharp \alpha(R_1)$$

On veillera à définir le meilleur tel opérateur.
Justifier l'utilité de la première composante des éléments abstraits, décrivant les variations de tailles de la pile.

Corrigé 10 *Supposons que $\alpha(R_i) = (D_i, E_i)$. Alors :*

$$\begin{array}{rcl}
\alpha(R_0 \circledcirc R_1) & = & (\{\bar{p_1} - \bar{p_0} \mid (p_0, p_1) \in R_0 \circledcirc R_1 \cap \mathbb{P} \times \mathbb{P}\}, \{\bar{p} \mid (p, \Omega) \in R_0 \circledcirc R_1\}) \\
& & \quad \textit{by definition of } \alpha \\
& = & (\{\bar{p_1} - \bar{p_0} \mid (p_0, p_1) \in \{(p_0, s_2) \in \mathbb{P} \times \mathbb{P}_\Omega \mid \exists p_1 \in \mathbb{P}, (p_0, p_1) \in R_0 \wedge (p_1, s_2) \in R_1\} \\
& & \qquad \cup (R_0 \cap \{(p, \Omega) \mid p \in \mathbb{P}\}) \cap \mathbb{P} \times \mathbb{P}\}, \\
& & \quad \{\bar{p} \mid (p, \Omega) \in \{(p_0, s_2) \in \mathbb{P} \times \mathbb{P}_\Omega \mid \exists p_1 \in \mathbb{P}, (p_0, p_1) \in R_0 \wedge (p_1, s_2) \in R_1\} \cup \\
& & \quad (R_0 \cap \{(p, \Omega) \mid p \in \mathbb{P}\})\}) \\
& & \quad \textit{by definition of } \circledcirc \\
& = & (\{\bar{p_2} - \bar{p_0} \mid \exists p_1 \in \mathbb{P}, (p_0, p_1) \in R_0 \wedge (p_1, p_2) \in R_1\}, \\
& & \quad \{\bar{p_0} \mid \exists p_1 \in \mathbb{P}, (p_0, p_1) \in R_0 \wedge (p_1, \Omega) \in R_1\} \cup \{\bar{p_0} \mid (p_0, \Omega) \in R_0\}) \\
& \subseteq & (\{d_0 + d_1 \mid d_0 \in D_0, d_1 \in D_1\}, \{e_1 - d_0 \mid e_1 \in E_1, d_0 \in D_0, e_1 \geq d_0\} \cup E_0\}
\end{array}$$

D'où l'opérateur abstrait :

$$\begin{array}{rccl}
\circledcirc^\sharp : & \mathbb{A}^2 & \longrightarrow & \mathbb{A} \\
& ((D_0, E_0), (D_1, E_1)) & \longmapsto & (\{d_0 + d_1 \mid d_0 \in D_0, d_1 \in D_1\}, \{e_1 - d_0 \mid e_1 \in E_1, d_0 \in D_0, e_1 \geq d_0\} \cup E_0\}
\end{array}$$

Cet opérateur garantit bien que $\alpha(R_0 \circledcirc R_1) \sqsubseteq \alpha(R_0) \circledcirc^\sharp \alpha(R_1)$.

Avant d'obtenir une analyse statique, nous devons encore résoudre un autre problème : notre domaine abstrait utilise des ensembles d'entiers qui ne sont pas tous représentables, et nous devons donc nous restreindre à certains ensembles représentables et sur lesquels il est possible de calculer l'approximation de la composition.

Question 11 Application de l'abstraction des intervalles.

Définir un nouveau domaine abstrait $\mathbb{A}_\mathcal{I}$, dont les éléments sont des paires d'intervalles, et qui abstrait chaque composante de $\mathbb{A}$ à l'aide d'un intervalle. On écrira la correspondance de Galois qui définit cette nouvelle abstraction.

Corrigé 11 *Il suffit de poser $\mathbb{A}_\mathcal{I} = \mathcal{I}^2$ où $\mathcal{I}$ représente l'ensemble des intervalles. L'opérateur de comparaison $\sqsubseteq_\mathcal{I}$ est l'ordre produit. De plus :*

$$\begin{array}{rccl}
\alpha_i : & R & \longmapsto & ([\min \Delta, \max \Delta], [\min \Sigma, \max \Sigma]) \\
& & & \quad \textit{où } \Delta = \{\bar{p_1} - \bar{p_0} \mid (p_0, p_1) \in R \cap \mathbb{P} \times \mathbb{P}\} \textit{ et } \Sigma = \{\bar{p} \mid (p, \Omega) \in R\} \\
\gamma_i : & (D, E) & \longmapsto & \{(p_1, p_0) \in \mathbb{P}^2 \mid \bar{p_1} - \bar{p_0} \in D\} \cup \{(p, \Omega) \mid \bar{p} \in E\}
\end{array}$$

Question 12 Analyse statique et application.

Déduire de ce qui précède la définition d'une analyse statique qui calcule une sur-approximation du comportement de toute suite d'instructions. On notera $[\![i]\!]^\sharp_\mathcal{I}$ le résultat de l'application de cette analyse à l'instruction i (et de même pour une suite d'instructions f). Donner une preuve de sa correction. Appliquer cette analyse aux exemples de la question 4.

Corrigé 12 *Il suffit de définir la sémantique abstraite des instructions puis des suites d'instructions :*

$$\begin{array}{rcl}
\mathcal{A}[\langle n \rangle]_\mathcal{I} & = & ([1,1], \bot) \\
\mathcal{A}[\ominus]_\mathcal{I} & = & ([0,0],[0,0]) \\
\mathcal{A}[\boxplus]_\mathcal{I} & = & ([-1,-1],[0,1]) \\
\mathcal{A}[\boxminus]_\mathcal{I} & = & ([-1,-1],[0,1]) \\
\mathcal{A}[\boxtimes]_\mathcal{I} & = & ([-1,-1],[0,1]) \\
\mathcal{A}[\mathbf{depth}]_\mathcal{I} & = & ([1,1], \bot)
\end{array}
\qquad
\begin{array}{rcl}
\mathcal{A}[\mathbf{drop}]_\mathcal{I} & = & ([-1,-1],[0,0]) \\
\mathcal{A}[\mathbf{dup}]_\mathcal{I} & = & ([1,1],[0,1]) \\
\mathcal{A}[\mathbf{pick}]_\mathcal{I} & = & ([1,1],[0,+\infty[) \\
\mathcal{A}[\mathbf{roll}]_\mathcal{I} & = & ([1,1],[0,+\infty[) \\
\mathcal{A}[\mathbf{swap}]_\mathcal{I} & = & ([0,0],[0,1])
\end{array}$$

Ensuite, l'opérateur $\circledcirc^\sharp_{\mathcal{I}}$ est défini en utilisant les opérations classiques sur les intervalles :

$$\circledcirc^\sharp_{\mathcal{I}} : \quad ((D_0, E_0), (D_1, E_1)) \quad \longmapsto \quad (D_0 + D_1, (E_1 - D_0) \sqcap [0, +\infty[\sqcup E_0)$$

Puis finalement :

$$\begin{aligned} \mathcal{A}[\cdot]_{\mathcal{I}} &= ([0,0], \bot) \\ \mathcal{A}[i\ f]_{\mathcal{I}} &= \mathcal{A}[i]_{\mathcal{I}} \circledcirc^\sharp_{\mathcal{I}} \mathcal{A}[f]_{\mathcal{I}} \end{aligned}$$

La correction de l'analyse se prouve par induction sur la syntaxe, et découle de la correction de la sémantique abstraite de l'analyse de chaque instruction ainsi que de $\circledcirc^\sharp_{\mathcal{I}}$.

On a donc :

- $\mathcal{A}[f_0]_{\mathcal{I}} = ([0,0], [0,0])$
- $\mathcal{A}[f_1]_{\mathcal{I}} = ([0,0], [0,1])$
- $\mathcal{A}[f_2]_{\mathcal{I}} = ([-1,-1], [0, +\infty[)$

Question 13 Amélioration de l'analyse.

L'analyse statique que nous avons ainsi obtenue donne-t'elle des résultats satifaisants en termes de précision ? Comparer les résultats obtenus à la question 12 à ceux que nous avions obtenus à la question 8. Si vous remarquez une perte de précision, suggérer une amélioration possible.

Corrigé 13 *On remarque qu'on obtient les résultats les plus précis possibles pour f_0 et f_1.*
Par contre, pour f_2, on a obtenu un résultat très imprécis, en raison de l'analyse très imprécise de **pick**. *Cela est dû au fait que cette instruction pourrait en général aller chercher un élément à une profondeur arbitrairement grande, mais en pratique ce n'est pas un problème, car on trouve la plupart du temps la valeur exacte de la profondeur juste avant (c'est le cas pour f_2). Il est possible de remédier à cette perte de précision en traitant plus précisément la suite $\langle n \rangle$* **pick** *(il faudrait faire la même chose pour* **roll** *:*

$$\alpha_i \langle n \rangle \ \mathbf{pick} = ([0,0], [0, n-1])$$

Si on applique cela à f_2, on obtient le résultat optimal !

5 Ajout d'une structure conditionnelle

Le langage que nous avons considéré jusqu'à présent n'est pas très expressif, et nous allons lui ajouter des structures conditionnelles dans cette section, puis des boucles dans la section suivante.

On étend l'ensemble des instructions en ajoutant au langage une instruction $\mathbf{if}[f_t \mid f_f]$ qui dépile la valeur n au sommet de la pile, puis exécute la suite d'instructions f_t si $n > 0$ et la suite d'instructions f_f sinon :

i	::=	$\ldots$	toutes les instructions vues précédemment
	$\mid$	$\mathbf{if}[f_t \mid f_f]$	structure conditionnelle

Ainsi :

état précédent	instruction	état suivant
$7:9:2:\epsilon$	$\mathbf{if}[\boxplus \mid \boxtimes]$	$11:\epsilon$
$0:4:8:\epsilon$	$\mathbf{if}[\boxplus \mid \ominus \boxtimes]$	$(-32):\epsilon$
$(-12):2:\epsilon$	$\mathbf{if}[\ominus \mid \boxtimes]$	Ω

Question 14 Sémantique concrête.

Définir la sémantique concrête de l'instruction $\mathbf{if}[f_t \mid f_f]$. On utilisera les relations suivantes, ainsi que les opérateurs $\circledcirc$ et $\cup$ pour exprimer cette sémantique de manière concise :

$$\mathbf{filter}_+ = \{(n:p, p) \mid n \in \mathbb{Z}^*_+, p \in \mathbb{P}\} \qquad \mathbf{filter}_- = \{(n:p, p) \mid n \in \mathbb{Z}_-, p \in \mathbb{P}\}$$

Corrigé 14 *Une exécution de la condition peut se voir comme un test, puis l'exécution de la branche correspondante. On obtient donc :*

$$[\![\mathbf{if}[f_t \mid f_f]]\!] = [\![f_t]\!] \circledcirc \mathbf{filter}_+ \cup [\![f_f]\!] \circledcirc \mathbf{filter}_-$$

Question 15 Analyse statique.

Déduire de la question 14 une définition de $[\![\mathbf{if}[f_t \mid f_f]]\!]^\sharp$ *et de* $[\![\mathbf{if}[f_t \mid f_f]]\!]^\sharp_{\mathcal{I}}$*, de manière à étendre l'analyse statique définie à la question 12.*

Corrigé 15 *On rappelle que* $\alpha(x \cup y) = \alpha(x) \sqcup \alpha(y)$*. Donc :*

$$\begin{array}{rcl}
\alpha_i([\![\mathbf{if}[f_t \mid f_f]]\!]) & = & \alpha_i([\![f_t]\!] \circledcirc \mathbf{filter}_+ \cup [\![f_f]\!] \circledcirc \mathbf{filter}_-) \\
& = & \alpha_i([\![f_t]\!] \circledcirc \mathbf{filter}_+) \sqcup_{\mathcal{I}} \alpha([\![f_f]\!] \circledcirc \mathbf{filter}_-) \\
& \sqsubseteq_{\mathcal{I}} & \alpha_i([\![f_t]\!]) \circledcirc^\sharp_{\mathcal{I}} \alpha_i(\mathbf{filter}_+) \sqcup_{\mathcal{I}} \alpha_i([\![f_t]\!]) \circledcirc^\sharp_{\mathcal{I}} \alpha_i(\mathbf{filter}_-) \\
& \sqsubseteq_{\mathcal{I}} & [\![f_t]\!]^\sharp_{\mathcal{I}} \circledcirc^\sharp_{\mathcal{I}}([-1,-1],[0,0]) \sqcup_{\mathcal{I}} [\![f_f]\!]^\sharp_{\mathcal{I}} \circledcirc^\sharp_{\mathcal{I}}([-1,-1],[0,0])
\end{array}$$

On en déduit l'extension de l'analyse statique :

$$\mathcal{A}[\mathbf{if}[f_t \mid f_f]]_{\mathcal{I}} = \mathcal{A}[f_t]_{\mathcal{I}} \circledcirc^\sharp_{\mathcal{I}}([-1,-1],[0,0]) \sqcup_{\mathcal{I}} \mathcal{A}[f_f]_{\mathcal{I}} \circledcirc^\sharp_{\mathcal{I}}([-1,-1],[0,0])$$

Question 16 Exemples.

Définir une suite d'instructions qui calcule la valeur absolue de l'élément au sommet de la pile (et plante si celle-ci est vide). Donner le résultat de l'analyse statique de cette suite d'instructions.
Donner également les résultats des analyses statiques des suite d'instructions qui figurent dans le tableau ci-dessus.

Corrigé 16 *La valeur absolue peut être calculée par la fonction* $f_3 = \mathbf{dup}\ \mathbf{if}[\mid \ominus]$*. L'analyse statique de cette fonction donne* $([0,0],[0,0])$ *(autrement dit la pile ne varie pas de taille, et la fonction plante si et seulement si la pile est vide initialement).*

On a par ailleurs :

$$\begin{array}{rcl}
\mathcal{A}[\mathbf{if}[\boxplus \mid \boxtimes]]_{\mathcal{I}} & = & ([-2,-2],[0,2]) \\
\mathcal{A}[\mathbf{if}[\boxplus \mid \ominus \boxtimes]]_{\mathcal{I}} & = & ([-2,-2],[0,2]) \\
\mathcal{A}[\mathbf{if}[\ominus \mid \boxtimes]]_{\mathcal{I}} & = & ([-2,-1],[0,2])
\end{array}$$

6 Ajout d'une structure de boucle

Nous étendons à présent notre langage d'une structure de boucle :

$$\begin{array}{rcll}
i & ::= & \ldots & \text{toutes les instructions vues précédemment} \\
& \mid & \mathbf{while}[f_l] & \text{boucle}
\end{array}$$

Cette instruction dépile la valeur au sommet de la pile, et ne fait rien d'autre si cette valeur est négative ou nulle ; dans le cas où la valeur qui était stockée au sommet de la pile était positive, elle exécute la suite d'instructions f_l, puis recommence.

Question 17 Sémantique concrête.

Définir la sémantique concrête de l'instruction $\mathbf{while}[f_l]$*. On pourra s'inspirer de la sémantique de la structure conditionnelle (question 14) afin de l'exprimer par une formule rendant l'analyse statique aisée.*

Corrigé 17 *La définition est très similaire :*

$$[\![\mathbf{while}[f_l]]\!] = \mathcal{X} \circledcirc \mathbf{filter}_-$$

où

$$\mathcal{X} = \mathit{lfp}_{([0,0],\bot)} \lambda \mathcal{Y} \cdot (\mathcal{Y} \circledcirc \mathbf{filter}_+ \circledcirc [\![f_l]\!])$$

Question 18 Analyse statique des boucles.

Déduire de la question précédente une définition de $[\![\mathbf{while}[f_l]]\!]^\sharp$*, ainsi qu'une approximation calculable* $[\![\mathbf{while}[f_l]]\!]^\sharp_{\mathcal{I}}$ *de la sémantique concrête.*
On effectuera des itérations croissantes avec élargissement (donner l'opérateur d'élargisssement).

Corrigé 18 *On applique les méthodes classiques d'approximation de points fixes avec élargissement (en utilisant l'opérateur d'élargissement* $\triangledown_{\mathcal{I}}$ *qui applique l'élargissement des intervalles à chaque composante de* $\mathbb{A}_{\mathcal{I}}$*), et on obtient :*

$$\mathcal{A}[\mathbf{while}[f_l]]_{\mathcal{I}} = (\lim X^\sharp_n)$$

où :

$$\begin{array}{rcl} X^\sharp_0 & = & ([0,0], \bot) \\ X^\sharp_{n+1} & = & X^\sharp_n \triangledown_{\mathcal{I}} (X^\sharp_n \circledcirc^\sharp_{\mathcal{I}} ([-1,-1],[0,0]) \circledcirc^\sharp_{\mathcal{I}} \mathcal{A}[f_l]_{\mathcal{I}}) \end{array}$$

Question 19 Exemple.

Écrire une suite d'instructions f *qui implémente le calcul suivant :*

$$[\![f]\!] = \{(n : p, \sum_{k=0}^{n} k^2) : p) \mid n \in \mathbb{N}, p \in \mathbb{P}\} \cup \{(n : p, (-1) : p) \mid n \in \mathbb{Z}_-, p \in \mathbb{P}\} \cup \{(\epsilon, \Omega)\}$$

Donner le résultat de l'analyse statique de cette fonction (on décrira les principales étapes de l'analyse, en particulier concernant les itérations sur la boucle).

Corrigé 19 *On considère la fonction*

$$f_4 = \mathbf{dup} \ \ominus \ \mathbf{if}[-1 \mid 0 \ \mathbf{swap\ dup\ while}[\langle 2 \rangle \ \mathbf{roll} \ \boxplus \ \mathbf{swap\ dup}] \ \mathbf{drop}]$$

Question 20 Vers une analyse plus précise.

Écrire une suite d'instruction f *qui lit un entier* n *sur la pile, et calcule l'addition des* n *éléments suivants. Donner les résultats de notre analyse statique sur* f*. Comment améliorer la précision ?*

Corrigé 20 *Pour analyser un tel exemple, il est indispensable de calculer des relations entre les valeurs sur la pile et la taille de celle-ci. Cela nous oriente vers un domaine abstrait nettement plus complexe que celui que nous avons considéré jusqu'à présent.*