Sémantique et applications à la vérification
Examen (durée : 2h) — 3 juin 2016

Ce sujet est composé de deux exercices indépendants.

Exercice 1 : correspondances de Galois et opérateurs de clôture supérieure

Dans cet exercice, nous étudions les propriétés des correspondances de Galois en lien avec la notion d'*opérateur de clôture supérieure*. On rappelle qu'un opérateur de clôture supérieure est un opérateur qui est extensif, croissant et idempotent (on appelle aussi un tel opérateur un *uco* —pour *upper closure operator*).

Dans cet exercice, nous supposons que $(C, \sqsubseteq, \bot, \top, \sqcup, \sqcap)$ forme un treillis complet et que $(A^\sharp, \sqsubseteq^\sharp)$ est un ensemble ordonné. Si ρ est un opérateur de clôture supérieure, on note $\mathbf{fp}(\rho)$ l'ensemble de ses points fixes. Enfin, un sous-ensemble non vide M de C est une *famille de Moore* si M est stable par borne inférieure dans C, c'est à dire si :

$$\forall X \subseteq M,\ (\sqcap X) \in M$$

On note $\mathcal{M}_C$ l'ensemble des familles de Moore de C et $\mathcal{U}_C$ l'ensemble des *uco*s sur C.

Question 1 Quelques propriétés simples.

Supposons que $\rho : C \longrightarrow C$ est un opérateur de clôture supérieure sur C.
Que vaut $\rho(\top)$?
Peut on dire quelque chose concernant $\rho(\bot)$?
Montrer que $\mathbf{fp}(\rho)$ est une famille de Moore.

Question 2 Caractérisation d'un uco par l'ensemble de ses points fixes.

Montrer qu'un opérateur de clôture supérieure ρ sur C peut être caractérisé par l'ensemble de ses points fixes $\mathbf{fp}(\rho)$ et décrire comment on peut calculer ρ en connaissant cet ensemble. On appellera **uco** *la fonction qui calcule ρ à partir de l'ensemble de ses points-fixes.*

Question 3 Ordre sur les opérateurs de clôture.

Soient ρ_0, ρ_1 deux opérateurs de clôture supérieure sur C. Montrer que les trois propositions suivantes sont équivalentes :

(i) $\rho_0 \dot{\sqsubseteq} \rho_1$ *(où $\dot{\sqsubseteq}$ décrit l'extension point à point de $\sqsubseteq$);*
(ii) $\rho_1 \circ \rho_0 = \rho_1$
(iii) $\mathbf{fp}(\rho_1) \subseteq \mathbf{fp}(\rho_0)$.

Question 4 Correspondance de Galois.

Montrer que :

$$(\mathcal{U}_C, \dot{\sqsubseteq}) \underset{\mathbf{fp}}{\overset{\mathbf{uco}}{\leftrightarrows}} (\mathcal{M}_C, \supseteq)$$

On étudie maintenant les relations entre opérateurs de clôture supérieure et correspondances de Galois.

Question 5 Des correspondances des Galois vers les opérateurs de clôture supérieure.
Supposons maintenant que $\alpha : C \longrightarrow A^\sharp$, $\gamma : A^\sharp \longrightarrow C$ forment une correspondance de Galois :

$$(C, \sqsubseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (A^\sharp, \sqsubseteq^\sharp)$$

Montrer que $\gamma \circ \alpha$ forme un opérateur de clôture supérieure sur C.

Question 6 Des opérateurs de clôture supérieure vers les correspondances de Galois.
Réciproquement, supposons que $\rho : C \longrightarrow C$ est un opérateur de clôture supérieure. On note F l'ensemble de ses points fixes. Montrer que la fonction $\alpha : C \longrightarrow F$ definie par $\forall x \in C,\ \alpha(x) = \rho(x)$ (cette fonction est bien définie car l'image de ρ est F par définition des opérateurs de clôture) est la fonction d'abstraction d'une correspondance de Galois dont on donnera la fonction de concrétisation. Montrer qu'il s'agit d'une "insertion de Galois", ce qui signifie que la fonction $\alpha \circ \gamma$ est l'identité.

On vient donc de voir que l'on peut construire un opérateur de clôture à partir d'une correspondance de Galois et une insertion de Galois à partir d'un opérateur de clôture. Nous allons donc considérer deux exemples :

Question 7 Abstraction des intervalles.
On considère le domaine des intervalles, sur les nombres entiers mathématiques. Donner le domaine concret, le domaine abstrait et les fonctions d'abstraction et de concrétisation. Calculer l'opérateur de clôture supérieure associé. Quelle est l'insertion de Galois qui est obtenue lorsqu'on applique la méthode vue à la question **??** *?*

Question 8 Abstraction non relationnelle.
On considère à présent l'abstraction non relationnelle décrite en cours (abstraction qui décrit un ensemble de fonctions par une fonction vers des ensembles) :

- *A et B sont deux ensembles ;*
- *$C = \mathcal{P}(A \to B)$ et $\sqsubseteq$ est l'inclusion ;*
- *$A^\sharp = A \to \mathcal{P}(B)$ et $\sqsubseteq^\sharp$ est l'extension point à point de l'inclusion.*

Donner les fonctions d'abstraction et de concrétisation et calculer l'opérateur de clôture supérieure associé. Quelle est l'insertion de Galois qui est obtenue lorsqu'on applique la méthode vue à la question **??** *? Comparer avec la question* **??**.

Exercice 2 : analyses de tableaux

Dans cet exercice, nous étudions quelques analyses statiques très simples ayant pour but de calculer une sur-approximation des ensembles de valeurs qui peuvent être stockées dans des cellules de tableaux. On s'intéressera uniquement à des abstractions *non relationnelles*.

Les valeurs de base sont les entiers positifs $\mathbb{N}$. On se donne un ensemble fini $\mathbb{X}_n$ de variables entières (que l'on note ci dessous $\mathtt{x}$ ou $\mathtt{x}_k$, où $k = 0, 1, \ldots$) et un ensemble fini $\mathbb{X}_t$ de variables de type "tableau d'entier" (que l'on note ci-dessous $\mathtt{t}$ ou $\mathtt{t}_k$, où $k = 0, 1, \ldots$). On pose $\mathbb{X} = \mathbb{X}_n \uplus \mathbb{X}_t$. Nous considérons le micro-langage impératif ci-dessous :

P	::=		programmes
	|	ϵ	programme vide
	|	C; P	séquence (commande suivie d'un programme)
	|	**while**($\mathtt{x} < \mathtt{t}$.**len**){P}	itération sur le tableau $\mathtt{t}$
C	::=		commandes
	|	**init**($\mathtt{t}, \mathtt{x}$)	création du tableau $\mathtt{t}$ de longueur $\max(1, \mathtt{x})$ et initialisation à zéro
	|	$\mathtt{x} := [\mathrm{v}_0, \mathrm{v}_1[$	écriture d'une valeur aléatoire v telle que $\mathrm{v}_0 \leq \mathrm{v} < \mathrm{v}_1$ dans la variable $\mathtt{x}$ où $\mathrm{v}_0 \in \mathbb{N}, \mathrm{v}_1 \in \mathbb{N} \uplus \{+\infty\}$
	|	**incr**($\mathtt{x}$)	incrémentation de la valeur stockée dans la variable $\mathtt{x}$
	|	$\mathtt{t}[\mathtt{x}_0] := \mathtt{x}_1$	copie du contenu de la variable $\mathtt{x}$ dans la cellule de $\mathtt{t}$ d'indice $\mathtt{x}$

Les instructions du langage se comportent comme suit :

```
P0 :  x0 = [0, 10 000];
      init(t, x0);
      x1 = 0;
      while(x1 < t.len){
        x2 = [0, 9 999];
        t[x1] = x2;
        incr(x1);
      }
```

```
P1 :  P0;
      x1 = 0;
      while(x1 < t.len){
        x2 = [0, 0];
        t[x1] = x2;
        incr(x1);
      }
```

Figure 1: Exemples

- la commande **init**(t, x) crée un nouveau tableau de longueur la valeur de x et initialise son contenu à zéro ; on considère que l'ancien tableau disparaît purement et simplement (le langage n'a pas de notion de pointeur) ;
- l'opération d'affectation à une variable concerne une valeur aléatoire, mais on peut simuler une affectation d'une constante à une variable x := v à l'aide de l'affectation d'une valeur aléatoire x := [v, v + 1[(que l'on notera aussi x := [v, v] —plus généralement, on abrégera x := [v_0, v_1[où $v_1 \in \mathbb{N}$ en x := [v_0, $v_1 - 1$]) ;
- dans le cas de l'écriture dans une cellule de tableau, et lorsque l'indice est en dehors des bornes du tableau, on considère qu'il n'y a pas de transition (pas d'état successeur) ; à titre d'exemple, si t est un tableau de longueur 2, et si x_0 vaut 6, la commande t[x_0] := x_1 ne produit pas de successeur (dans la suite, on ne s'intéresse pas aux erreurs de tableaux —même si on pourrait vouloir développer une analyse découvrant cette classe d'erreurs).

Ce langage ne permettrait pas d'écrire des programmes très intéressants (nous n'avons pas inclus d'opération de lecture dans un tableau, d'opérations arithmétiques, ou d'instruction de test), mais il est toutefois suffisant pour étudier quelques abstractions sur les tableaux.

Un état mémoire est une fonction m qui associe chaque variable entière et chaque cellule de tableau à sa valeur :

- si x est une variable de type entier, on note m(x) la valeur de la variable x dans l'état m ;
- si t est une variable de type tableau, on note m(t · **len**) sa longueur et m(t, i) la valeur stockée dans la cellule i.

On note $\mathbb{M}$ l'ensemble des états mémoire. Dans la suite, on utilise une sémantique dénotationnelle, qui décrit les comportements d'un programme à l'aide d'une fonction prenant en argument un ensemble d'états et retournant un ensemble d'états. Nous donnons quelques cas ci-dessous (on note m[y $\not\mapsto$] l'état mémoire obtenu à partir de m en supprimant la définition de la variable y —scalaire ou tableau— si celle-ci existe) :

$$
\begin{aligned}
[\![\epsilon]\!](M) &= M \\
[\![\mathrm{C};\mathrm{P}]\!](M) &= [\![\mathrm{P}]\!] \circ [\![\mathrm{C}]\!](M) \\
[\![\textbf{init}(\mathtt{t},\mathtt{x})]\!](M) &= \{m[(\mathtt{t},0) \mapsto 0, \ldots, (\mathtt{t}, \max(0, m(\mathtt{x}) - 1)) \mapsto 0, \\
&\qquad (\mathtt{t}, m(\mathtt{x})) \not\mapsto, (\mathtt{t}, m(\mathtt{x}) + 1) \not\mapsto, \ldots] \mid m \in M\} \\
[\![\mathtt{x} := [\mathrm{v}_0, \mathrm{v}_1[\,]\!](M) &= \{m[\mathtt{x} \mapsto \mathrm{v}] \mid m \in M \wedge \mathrm{v}_0 \leq \mathrm{v} < \mathrm{v}_1\}
\end{aligned}
$$

On donne deux exemples dans la figure **??** : P_0 créé un tableau de longeur aléatoire et écrit dans chaque cellule une valeur aléatoire entre 0 et 9 999 ; P_1 exécute les mêmes instructions que dans P_0, puis ré-initialise le tableau à 0.

Question 9 Compléter la sémantique.

Définir :

- *la sémantique de la commande* **incr**(x) *;*
- *la sémantique de la commande* t[x_0] := x_1 *(on rappelle que les états pour lesquels aurait lieu un accés en dehors des bornes sont ignorés) ;*
- *la sémantique de la boucle* **while**(x < t.**len**){P} *; on la donnera sous la forme d'un plus petit point fixe dont on justifiera l'existence.*

Question 10 Une première idée d'abstraction.

Une solution serait de définir un domaine abstrait permettant de décrire une sur-approximation des valeurs dans chaque cellule en utilisant un intervalle par cellule. Ainsi, l'ensemble d'états $\{(\!|(\mathtt{t},0) \mapsto 8, (\mathtt{t},1) \mapsto 9|\!), (\!|(\mathtt{t},0) \mapsto 6, (\mathtt{t},1) \mapsto 15|\!)\}$ *pourrait être décrit par* $(\!|(\mathtt{t},0) \mapsto [6,8], (\mathtt{t},1) \mapsto [9,15]|\!)$.

Toutefois, cette idée ne marche pas. *Expliquer pourquoi on ne peut s'appuyer sur une telle abstraction pour décrire les états que peuvent produire nos programmes.*

On va donc définir dans les questions suivantes une abstraction très différente. L'idée fondamentale est d'utiliser *une* variable abstraite pour décrire le contenu de *toutes* les cellules du tableaux. On dit que cette technique utilise des variables "résumant" chaque tableau.

Question 11 Domaine abstrait avec variables résumées.

On pose $\mathbb{M}_s = \{\Phi : \mathbb{X} \to \mathcal{P}(\mathbb{N}) \mid (\forall \mathtt{u},\ \Phi(\mathtt{u}) = \emptyset) \vee (\forall \mathtt{u},\ \Phi(\mathtt{u}) \neq \emptyset)\}$. *Définir les fonctions d'abstraction* α_s *et de concrétisation* γ_s *correspondantes, et telles que :*

$$(\mathcal{P}(\mathbb{M}), \subseteq) \underset{\alpha_s}{\overset{\gamma_s}{\leftrightarrows}} (\mathbb{M}_s, \dot{\subseteq})$$

Indice : *on s'inspirera de l'abstraction non relationnelle, et on décrira toutes les cellules d'un tableau* `t` *à l'aide d'une seule variable abstraite que l'on appelera aussi* `t` *(c'est pour cela que l'on prend un domaine abstrait à base de fonctions définies sur* $\mathbb{X}_n \uplus \mathbb{X}_t$*).*

Question 12 Exemples.

On considère le programme P_0 *défini sur la figure* **??**. *Décrire* $[\![\mathrm{P}_0]\!](\mathbb{M})$ *et* $\alpha_s([\![\mathrm{P}_0]\!](\mathbb{M}))$ *(autrement dit, on commence l'exécution du programme à partir de n'importe quel état). Faire de même pour* P_1.

Question 13 Sémantique avec variables résumées.

On souhaite définir $[\![\mathrm{P}]\!]_s$ *(resp.* $[\![\mathrm{C}]\!]_s$*) telles que, pour toute fonction* $\Phi \in \mathbb{M}_s$, *on ait* $[\![\mathrm{P}]\!] \circ \gamma_s(\Phi) = \gamma_s \circ [\![\mathrm{P}]\!]_s(\Phi)$ *(resp.,* $[\![\mathrm{C}]\!] \circ \gamma_s(M) = \gamma_s \circ [\![\mathrm{C}]\!]_s(\Phi)$*). Fournir une définition par induction sur la syntaxe de ces sémantiques, en apportant les justifications nécessaires.*

La sémantique avec résumés ne fournit toutefois pas une analyse statique, puisqu'elle manipule des ensembles de valeurs qui peuvent être de taille non bornée.

Question 14 Interprétation abstraite avec résumés et intervalles.

Proposer une abstraction à base d'intervalles, et fondée sur l'abstraction non relationnelle proposée à la question **??**. *On définira le domaine abstrait* $\mathbb{M}^\sharp$, *et les fonctions d'abstraction* α *et de concrétisation* γ *associées. Définir une fonction d'analyse* $[\![\mathrm{P}]\!]^\sharp$ *(resp.,* $[\![\mathrm{C}]\!]^\sharp$*) qui sur-approxime l'effet du programme* P *(resp., de la commande* C*), qui soit telle que, pour tout élément* $M^\sharp$ *de* $\mathbb{M}^\sharp$, *on ait* $[\![\mathrm{P}]\!]_s(\gamma(M^\sharp)) \subseteq \gamma([\![\mathrm{P}]\!]^\sharp(M^\sharp))$ *(resp.,* $[\![\mathrm{C}]\!]_s(\gamma(M^\sharp)) \subseteq \gamma([\![\mathrm{C}]\!]^\sharp(M^\sharp))$*). Les définitions seront données par induction sur la syntaxe.*

Question 15 Limite du domaine avec résumés.

Décrire les résultats et les étapes de l'analyse de P_0. *Expliquer en quoi cette analyse est imprécise. Donner les résultats de l'analyse de* P_1. *Commenter ces résultats.*
Que pourrait on faire pour éviter de telles imprécisions ?
Indice : *on pourra s'appuyer sur une preuve "à la main" des résultats de la question* **??**, *et déduire de cette preuve les propriétés que l'analyse statique aurait besoin de calculer pour arriver à un résultat plus précis.*