Cours MOdélisation, Vérification et EXpérimentations
Exercices
Structures partiellement ordonnées, treillis complets, points-fixes (I)
par Dominique Méry
13 mai 2025

☛ *Nous supposons que les opérations suivantes sont définies pour une collection $A_0, \ldots, A_i \ldots i \in \mathbb{N}$ de parties de l'ensemble E :*
- $\cup_{i\in\mathbb{N}} A_i = \{e \in E | \exists i \in \mathbb{N} : e \in A_i\}$
- $\cap_{i\in\mathbb{N}} A_i = \{e \in E | \forall i \in \mathbb{N} : e \in A_i\}$

Ces deux définitions sont correctes et légales. Elles définissent en compréhension deux ensembles.

Exercice 1 *Montrer que les structures suivantes sont des structures partiellement ordonnées inductives :*

Question 1.1 $(\mathbb{P}(E), \subseteq)$ *avec E un ensemble quelconque.*

Question 1.2 $(E^\perp, \sqsubseteq)$ *tel que*

1. $E^\perp = E \cup \perp$ *(et $\perp \notin E$)*
2. $\sqsubseteq \subseteq E \times E$ *: soit $x \in E$ et $y \in E$ $x \sqsubseteq y \Leftrightarrow (x = \perp \vee x = y)$*

Question 1.3 *Soient A et B deux ensembles. Montrer que la structure $(A \nrightarrow B, \sqsubseteq)$ est une structure partiellement ordonnée inductive.*

Exercice 2 *On rappelle qu'une fonction continue au sens de la topologie de Scott est monotone croissante. Indiquer et montrer si les fonctionnelles suivantes sont monotones et/ou continues).*

1. $(F_1(f))(x) \mathrel{\widehat{=}}$ if $(\forall y \in \mathbb{Z}.f(y) = y)$ then $f(x)$ else $\perp$
2. $(F_2(f))(x) \mathrel{\widehat{=}}$ if $x \notin dom(f)$ then 0 else $\perp$
3. $(F_3(f))(x) \mathrel{\widehat{=}}$ if $x = 0$ then 1 else $f(x{+}1)$

$\perp$ *est une expression qui signifie que cést une valeur indéfinie.*

Question 2.1 $(F_1(f))(x) \mathrel{\widehat{=}}$ if $(\forall y \in \mathbb{Z}.f(y) = y)$ then $f(x)$ else $\perp$

Question 2.2 $(F_2(f))(x) \mathrel{\widehat{=}}$ if $x \notin dom(f)$ then 0 else $\perp$

Question 2.3 $(F_3(f))(x) \mathrel{\widehat{=}}$ if $x = 0$ then 1 else $f(x{+}1)$

Exercice 3 *Déterminer les points-fixes des fonctionnelles suivantes et leur plus petit point-fixe, s'ils existent. On travaille dans $\mathbb{Z}$.*

1. $F_1(f)(x) \mathrel{\widehat{=}}$ if $f(x) \equiv 0$ then 1 else 0 *et expliquer si le programme f1 a du sens et ce qu'il calcule :*
2. $F_2(f)(x) \mathrel{\widehat{=}}$ if $f(x) \equiv 0$ then 0 else 1
3. $F_3(f)(x) \mathrel{\widehat{=}}$ if $x = 0$ then 1 else $f(x{+}1)$

Exercice 4 *Soit $E = \mathbb{N} \nrightarrow \mathbb{N}$, soit la fonctionnelle $\tau \in E \nrightarrow E$ définie par :*

$$(\tau(F))(x) \mathrel{\widehat{=}} \text{ if } x = 0 \text{ then } 1 \text{ else } x \cdot F(x{-}1)$$

1. *Calculer $\tau(\varnothing)$, $\tau^2(\varnothing) = \tau(\tau(\varnothing))$, $\tau^3(\varnothing)$. En déduire $\tau^i(\varnothing)$ et le démontrer par récurrence.*
2. *En déduire le plus petit point fixe.*

Question 4.1 *Calculer* $\tau(\varnothing)$, $\tau^2(\varnothing) = \tau(\tau(\varnothing))$, $\tau^3(\varnothing)$. *En déduire* $\tau^i(\varnothing)$ *et le démontrer par récurrence.*

Question 4.2 *En déduire le plus petit point fixe.*

Exercice 5 *Soit la fonctionnelle* $\tau \in (\int \nrightarrow \int) \nrightarrow (\int \nrightarrow \int)$ *:*

$$(F(f))(x) \mathrel{\widehat{=}} \text{ if } x > 100 \text{ then } x{-}10 \text{ else } f(f(x{+}11))$$

Question 5.1 *Montrez que* $\mu F \sqsubseteq g$ *avec*

$$g(x) \mathrel{\widehat{=}} \text{ if } x > 100 \text{ then } x{-}10 \text{ else } 91$$

Question 5.2 *Ecrire une fonction C calculant cette fonction et étudier sa correction.*

Exercice 6 *On considère une fonction* $f5$ *définie par le code C suivant :*

```
int f5(int x)
{if (x==0)
    { return(0);}
  else
    { if  (x > 0)
        {return(2-f5(1-x));}
      else
        {/* x <0 */
          return(f5(-x));}
    }
}
```

Question 6.1 *Traduire cette définition en une définition fonctionnelle qui précisera le domaine du problème.*

Question 6.2 *Soient les définitions suivantes où* $\mathcal{F}$ *désigne la fonctionnelle définie dans la question précédente :*

- $\mathcal{F}^{2n} = \{(p, v_p) | 0 \leq p < n \wedge v_p = g(p)\} \cup \{(p, v_p) | 0 > p \geq -(n{-}1) \wedge v_p = g(p)\} \cup \{(n, g(n))\}$
- $\mathcal{F}^{2n+1} = \{(p, v_p) | 0 \leq p < n \wedge v_p = g(p)\} \cup \{(p, v_p) | 0 > p \geq -(n{-}1) \wedge v_p = g(p)\} \cup \{(n, g(n)), (-n, g(-n))\}$

Montrer qu'elles sont correctes en utilisant une récurrence.

Question 6.3 *En déduire que* $\mu\mathcal{F} = \mathcal{F}^0 \cup \mathcal{F}^1 \cup \mathcal{F}^2 \cup \mathcal{F}^3 \cup \ldots \cup \mathcal{F}^{2n} \cup \mathcal{F}^{2n+1} \ldots$

Question 6.4 *Montrer que, pour tout* $p \in \mathbb{N}$, $p \in \mathcal{F}^{2p} \cup \mathcal{F}^{2p+1}$

Question 6.5 *Montrer que* $\mu\mathcal{F}$ *vérifie la propriété* $\mu\mathcal{F} \sqsubseteq g$ *où* $g(x) = \textbf{if } odd(x) \textbf{ then } 2 \textbf{ else } 0 \textbf{ fi}$

Question 6.6 *En déduire que* $\mu\mathcal{F} = g$.

Exercice 7 *Soit la fonction définie comme suit :* $F(f)(x) = \begin{cases} \textbf{if } x = p \textbf{ then } p \\ \textbf{else if } x = q \textbf{ then } q \\ \textbf{else } f(x{+}p{+}q) \\ \textbf{fi} \end{cases}$.

On suppose que p *et* q *sont deux constantes non nulles entières positives distinctes et que* F *est une fonction partielle (*$\in (\mathbb{Z} \nrightarrow \mathbb{Z}) \longrightarrow (\mathbb{Z} \nrightarrow \mathbb{Z})$*). On se place dans le cadre de la topologie de Scott sur léspace* $(\mathbb{Z} \nrightarrow \mathbb{Z}) \longrightarrow (\mathbb{Z} \nrightarrow \mathbb{Z}), \sqsubseteq)$ *où* $f \sqsubseteq g$ *signifie que* f *est moins définie que* g *(ou* $graph(f) \subseteq graph(g)$*).*

Question 7.1 *Expliquez clairement pourquoi l'équation $F(f) = f$ admet un plus petit point-fixe.*

Question 7.2 *Ecrire une fonction C calculant le plus petit point-fixe μF de F.*

Question 7.3
1. *Calculer $\mu F(p)$, $\mu F(q)$, $\mu F(-p)$, $\mu F(-q)$.*
2. *Calculer pour k entier naturel, $\mu F(-(k+1)\cdot p - k\cdot q)$ et $\mu F(-(k+1)\cdot q - k\cdot p)$*

Question 7.4 *Donnez une expression simplifiée de la fonction μF. Pour cela, on pourra utiliser la caractérisation de μF par le théorème du point-fixe pour les fonctions continues au sens de Scott.*

Exercice 8 *(invariant inductif)*
On rappelle les définitions suivantes. Un modèle relationnel $\mathcal{MS}$ pour un système $\mathcal{S}$ est une structure

$$(Th(s,c), x, \textsc{Vals}, \textsc{Init}(x), \{r_0, \ldots, r_n\})$$

où
- $Th(s,c)$ *est une théorie définissant les ensembles, les constantes et les propriétés statiques de ces éléments.*
- x *est une liste de variables flexibles.*
- VALS *est un ensemble de valeurs possibles pour x.*
- $\{r_0, \ldots, r_n\}$ *est un ensemble fini de relations reliant les valeurs avant x et les valeurs après x'.*
- $\textsc{Init}(x)$ *définit lénsemble des valeurs initiales de x.*

On note $\textsc{Next} \stackrel{def}{=} r_0 \vee \ldots \vee r_n$. *Une propriété S est une propriété de sûreté pour le système $\mathcal{S}$, si* $\forall y, x \in \textsc{Vals}.Init(y) \wedge \textsc{Next}^\star(y,x) \Rightarrow x \in S$. *On définit la fonction suivante F sur $\mathcal{P}(\textsc{Vals})$ à valeurs dans $\mathcal{P}(\textsc{Vals})$: $F(X) = Init \cup Next[X]$ où $Next[X]$ est lénsemble des états accessibles à partir de X par $Next$. On rappelle aussi que x peut être une variable ou une liste de variables ;* VALS *est donc un ensemble de valeurs ou de tuples de valeures correspondant à x.*

Question 8.1 *Montrer que $(\mathcal{P}(\textsc{Vals}), \subseteq, \varnothing, \cup, \cap)$ est un treillis complet.*

Question 8.2 *Montrer que F est croissante monotone.*

Question 8.3 *Montrer que F admet un plus petit point-fixe noté μF.*

Question 8.4 *Montrer que μF est un invariant inductif de F et que cést le plus petit.*

Question 8.5 *Montrer que, pour toute propriété de sûreté S, $\mu F \subseteq S$.*

Question 8.6 *On suppose que* VALS *est finie. Montrer qu'il existe un algorithme pour vérifier qu'une propriété S est une propriété de sûreté pour un système donné défini comme ci-dessus.*

Exercice 9 Question 9.1 *Soit le petit programme suivant annoté mais incomplet.*

```
 /*@ requires z == exp1 ;
  ensures \result == exp2;
*/

int q2(int x,int y, int z){

  /*@ assert B(x,y,z)  ; */
  x = y+1;
  /*@  assert  A(x,y,z); */
```

```
  y = x + z;
  /*@  assert  y == 3 + x ; */
  return y;
  }
```

En utilisant l'opérateur wp, proposer des assertions pour $A(x,y,z)$ et $B(x,y,z)$ et des valeurs pour les expressions expr1 et expr2, afin que le contrat soit correct.

Question 9.2 *Soit le petit programme suivant annoté mais incomplet.*

```
/*@ requires  A(x,y,z)  ;
  ensures \result == 6 ;
*/

int q2(int x,int y, int z){

  /* @ assert B(x,y,z)  ; */
   x = y+1;
  /* @  assert  C(x,y,z); */
   y = x + z;
  /* @  assert  D(x,y,z); */
   return y;
   }
```

En utilisant l'opérateur wp, proposer des assertions pour $A(x,y,z)$, $B(x,y,z)$, $C(x,y,z)$ et $D(x,y,z)$, afin que le contrat soit correct.

Question 9.3 *Soit le petit programme suivant annoté mais incomplet.*

```
/*@ requires A;
ensures \result == 0;
  assigns \nothing;
*/

int f(int c) {
     //@ assert 49 == 49 && 2*c ==  2*c &&  49+2*c+1 ==  (c+1)*(c+1);
  int x = 49;
     //@ assert x == 49 && 2*c ==  2*c &&  x+2*c+1 ==  (c+1)*(c+1);
  int z = 2*c;
    //@ assert x == 49 && z ==  2*c &&  x+z+1 ==  (c+1)*(c+1);
  int y = (2*c+1)*(2*c+1);
    // @ assert B;
    //@ assert x == 49 && z ==  2*c &&  x+z+1 ==  (c+1)*(c+1);
    y= x+z+1;
    //@ assert x == 49 && z ==  2*c && y ==  (c+1)*(c+1);
    return(0);
}
```

En utilisant l'opérateur wp, proposer des assertions pour A et B, afin que le contrat soit correct.

Question 9.4 *Soit le petit programme suivant annoté mais incomplet.*

```
/*@ requires  A(x,y,z)  ;
  ensures \result == 12 ;
*/

int q2(int x,int y, int z){

  /* @ assert B(x,y,z)  ; */
```

```
  x = y+z;
  /* @  assert  C(x,y,z); */
  y = x + 1;
  /* @  assert  D(x,y,z); */
  return y;
  }
```

En utilisant l'opérateur wp, proposer des assertions pour $A(x,y,z)$, $B(x,y,z)$, $C(x,y,z)$ et $D(x,y,z)$, afin que le contrat soit correct.

Question 9.5 *Soit le petit programme suivant annoté mais incomplet.*

```
33/*@ requires A;
ensures \result == 0;
  assigns \nothing;
*/

int f(int c) {
    //@ assert (2*c == 14 ==> 49 == 49 && 2*c ==  2*c && 49+2*c +1 ==  (c+1)*(c+1))
&& (2*c != 14 ==> 49 == 49 && 2*c ==  2*c && 49+1==  50);
if
  int x = 49;
  int z = 2*c;

      //@ assert (z == 14 ==> x == 49 && z ==  2*c && x+z+1 ==  (c+1)*(c+1));
//@ assert (z != 14 ==> x == 49 && z ==  2*c && x+1==  50);

      //@ assert (z == 14 ==> x == 49 && z ==  2*c && x+z+1 ==  (c+1)*(c+1))
&& (z != 14 ==> x == 49 && z ==  2*c && x+1==  50);
if  z == 14
            {
    //@ assert x == 49 && z ==  2*c && x+z+1 ==  (c+1)*(c+1);
  int y = (2*c+1)*(2*c+1);
    // @ assert B
;    //@ assert x == 49 && z ==  2*c && x+z+1 ==  (c+1)*(c+1);
    y= x+z+1;
    //@ assert x == 49 && z ==  2*c && y ==  (c+1)*(c+1);

            }
  else
    {
        // @ assert C;
    //@ assert x == 49 && z ==  2*c && x+1==  50;
    y= x+1;
    //@ assert x == 49 && z ==  2*c && y ==  50;

    }
      //@ assert (x == 49 && z ==  2*c && y ==  (c+1)*(c+1)) || (x == 49 && z ==
2*c && y ==  50);

    return(0);
}
```

En utilisant l'opérateur wp, proposer des assertions pour A et B, afin que le contrat soit correct.