TD 2 : Logique et spécifications

Matthieu Sozeau (matthieu.sozeau@inria.fr)

October 7, 2014

Ce TD porte sur l'utilisation de la logique du premier ordre pour la spéficiation de programmes. Les notes de cours sont disponibles à l'adresse:
www.pps.univ-paris-diderot.fr/~sozeau/teaching/MVF-2014-lecture2.pdf

Le corrigé du TD1 est disponible à l'adresse:
www.pps.univ-paris-diderot.fr/~sozeau/teaching/MVF-2014-td1-corr.pdf

Le fichier Coq contenant la même correction formalisée et commentée est à l'adresse:
www.pps.univ-paris-diderot.fr/~sozeau/teaching/MVF-2014-td1-corr.v

Formules logiques et preuves

Exercice 1 (Variables libres et liées)**.** *Dans les formules suivantes sur les entiers naturels, déterminez les variables libres et liées. Quelles formules sont closes ?*

1. $8x, x = y$
2. $8x\ n, x \quad x + n$
3. $9x, x = s(y) \wedge y = s(x)$
4. $9y, y < x$
5. $9y, x < y$
6. $8x, 9y, x < y$
7. $8x\ y, x \quad y \_ y \quad x$

Solution

1. x est liée, y est libre, formule non close.
2. Formule close
3. y est libre, formule non close.
4. x est libre, non close.

5. x est libre, non close.
6. close, x et y liées.
7. close, x et y liées.

Théorie des entiers

On se donne la théorie des entiers de Peano avec les axiomes suivants:

- $\forall x\ y, s(x) = s(y) \Rightarrow x = y$
- $\forall x, 0 \neq s(x)$
- $\forall x\ n, x \neq s^n(x)$.
- Associativité, commutativité de l'addition et la multiplication.
- 0 est neutre pour l'addition, abosrbant pour la multiplication.
- 1 est neutre pour la multiplication.
- $\forall x, x \quad x$
- $\forall x, x \quad s(x)$
- $\forall x, x \quad y \Rightarrow y \quad z \Rightarrow x \quad z$
- $: \exists x, x < 0$
- Irréflexivité de $<$: $: \exists x, x < x$
- Transitivité de $<$.

Exercice 2 (Validité et satisfiabilité)**.** *Pour les formules 1, 2, 3, et 5 de l'exercice précédent, démontrez qu'elles sont valides ou invalides (que leur négation est valide) en utilisant la théorie des entiers. Pour les formules non closes, on supposera qu'elles sont quantifiées universellement sur leurs variables libres.*

Solution

1. Invalide. Supposons y arbitraire, alors on montre $: \forall x, x = y$. Supposons $\forall x\ y, x = y$. On applique l'hypothèse sur 0 et 1 pour obtenir $0 = 1$, et on obtient une contradiction par l'axiome 2 ($0 \neq 1$).
2. Par induction sur n.
   - Cas $n = 0$: $x \quad x + 0$ par réflexivité.
   - Cas $n = s(n')$. Par induction $x \quad x+n'$ On montre $x \quad x+s(n') \Leftrightarrow x \quad s(x+n')$. Par transitivité avec $x+n'$ on doit montrer $x \quad x+n'$ (par l'hypothèse d'induction) et $x + n' <= s(x + n')$ par axiome.

3. Invalide. Supposons y arbitraire et montrons $: (\exists x, x = s(y) \wedge y = s(x)$. On suppose un x arbitraire tel que $H : x = s(y) \wedge y = s(x)$. Par substitution d'une égalité dans l'autre on obtient $x = s(s(x))$, contradiction.

4. Invalide. Supposons $\forall x, \exists y, y < x$. On applique l'hypothèse à 0 pour obtenir: $\exists y, y < 0$, contradiction.

5. Valide. Supposons un x, soit $y := s(x)$, on a bien $x < s(x)$.

6. idem.

Spécifications

Exercice 3. *Donner une spécification de l'action de la fonction* `rev` (l) *sur chaque élement de la liste* l*.*

Solution $SpecRev(\ell, \ell') ::= |\ell| = |\ell'| \wedge \forall i \in Nat, i < |\ell| \Rightarrow \ell[i] = \ell'[|\ell| - s(i)]$

Exercice 4. *Donner une définition* récursive *du prédicat* In *qui spécifie quand un élément appartient à une liste à partir du prédicat* $l[i] = e$*.*

Solution $In(x, \ell) ::= \exists i, (i < |\ell|) \wedge \ell[i] = x$

Exercice 5. *Donner une spécification d'une fonction calculant le plus grand entier d'une liste d'entiers.*

Solution $\forall l \in List[Nat], \forall m, SpecMax(\ell, m) \Leftrightarrow \ldots In(m, \ell) \wedge \forall x, In(x, \ell) \Rightarrow \ldots$
$x \quad m$

On rappelle que le produit cartésien de domaines $A \times B$ pour deux domaines A et B a pour valeurs des pairs (a, b) où $a \in A$ et $b \in B$.

Exercice 6. *Donner une spécification Spec_Split d'une fonction qui décompose une liste en une paire de deux listes à un indice donné (utilisez la spéfication Spec_Append).*

Solution $\forall \ell : List[\ldots], \forall n \in Nat, \forall p \in (List[\ldots] \times List[\ldots])$,
Spec_Split$(\ell, n, p) := \exists \ell 1, \ell 2, p = (\ell 1, \ell 2) \wedge |\ell 1| = n \wedge$ Spec_Append$(\ell 1, \ell 2, \ell)$

Multiensembles

On rappelle que les multiensembles sont modélisés par des fonctions: $Multiset[\star] = \star \to \mathbf{N}$.

- $\emptyset = \lambda x \in \star. 0$
- $Sg(a) = \lambda x \in \star.$ `if` $x = a$ `then` 1 `else` 0

- $M_1 \uplus M_2 = \lambda x \in \star.\ M_1(x) + M_2(x)$

Pour montrer l'égalité de deux multiensembles, il faut donc montrer que deux fonctions sont égales. Pour cela il suffit de montrer l'égalité point-à-point, c'est à dire sur chaque élément du domaine:
$\forall f\ g : \star \to \mathbf{N}, f = g \iff \forall x \in \star, f\ x = g\ x$

Exercice 7. *Énoncez et démontrez la propriété de neutralité du multiensemble vide pour l'union de multiensembles.*

Solution On veut montrer: $\forall M, \emptyset \uplus M = M$. Supposons $M : Ms[\star]$, on doit montrer $\emptyset \uplus M = M \iff (\lambda x, \emptyset(x) + M(x)) = M$

C'est équivalent à: $\forall x \in \star, (\lambda x, \emptyset(x) + M(x))(x) = M(x)$. On suppose $x \in \star$ et on doit montrer $\emptyset(x) + M(x) = M(x) \iff (\lambda x.0)(x) + M(x) = M(x) \iff 0 + M(x) = M(x)$ par la neutralité du 0 pour l'addition.

Exercice 8. *Montrez la commutativité et l'associativité de l'union de multiensembles.*

On suppose les propriétés de la concaténation et de l'inverse d'une liste prouvées au TD1.

Exercice 9. *Montrer les propriétés:*

- $Ms(l1@l2) = Ms(l1) \uplus Ms(l2)$
- $Ms(l1@l2) = Ms(l2@l1)$
- $Ms(\texttt{rev}\ l) = Ms(l)$

Solution

1. Par induction sur $l1$:
   - Cas $l1 = []$: trivial par les propriétés de la concaténation et $Ms([])$.
   - Cas $l1 = a :: l1'$: Hypothèse d'induction: $Ms(l1'@l2) = Ms(l1') \uplus Ms(l2)$ On doit montrer $Ms(a :: l1'@l2) = Ms(a :: l1') \uplus Ms(l2)$.

     $$\begin{aligned} Ms((a :: l1')@l2) &= Ms(a :: (l1'@l2)) \\ &= Sg(a) \uplus Ms(l1'@l2) \\ &\quad \text{(Hyp. Induction)} \\ &= Sg(a) \uplus Ms(l1') \uplus Ms(l2) \end{aligned}$$

     Et d'autre part:

     $$Ms(a :: l1') \uplus Ms(l2) = (Sg(a) \uplus Ms(l1')) \uplus Ms(l2)$$

2. En utilisant le lemme ci-dessus et la commutativité de l'union.

3. Par induction sur l.

   - Cas $l = []$, trivial par simplification.
   - Cas $l = a :: l'$: Hyp. Ind.: $Ms(\texttt{rev}\ l') = Ms(l)$. On simplifie: $Sg(a) \uplus Ms(l') = Ms(\texttt{rev}\ l' @ [a])$ Par les lemmes 1 et 2: $Sg(a) \uplus Ms(l') = Ms([a]) \uplus Ms(\texttt{rev}\ l')$. Par simplification de Ms et l'hypothèse d'induction.

Spécification du tri

Exercice 10. *Définir une fonction calculant les occurences d'un élément dans une liste.*

Solution

$$\begin{array}{rcl} occs(x, []) & = & 0 \\ occs(x, a :: l) & = & \texttt{if}\ x = a\ \texttt{then}\ s(occs(x, l))\ \texttt{else}\ occs(x, l) \end{array}$$

Exercice 11. *Définir un prédicat $perm(l, l')$ spécifiant quand l' est une permutation de l en utilisant la fonction précédente.*

Solution $perm(l, l') :: \Pi x : \star, occs(x, l) = occs(x, l')$

Exercice 12. *Donner une spéfication alternative d'une fonction de tri sur les listes basée sur la fonction précédente.*

Solution $Spec\_Sort(\ell, \ell') =$

$$\begin{array}{c} \Pi i, j, 2\ Nat.\ (i < j < |\ell|) \Rightarrow \ell'[i] \leq \ell'[j] \\ \wedge \\ perm(\ell, \ell') \end{array}$$

Prédicats inductifs

On rappelle la définition du prédicat inductif `even`:

$$\begin{array}{rcl} \texttt{even} & : & Nat \to prop \\ even0 & : & \texttt{even}\ 0 \\ evenS & : & \Pi n, \texttt{even}\ n \Rightarrow \texttt{even}\ s(s(n)) \end{array}$$

Exercice 13. *Donner une preuve de* $\texttt{even}\ 4$.

Exercice 14. *Montrer que* $\Pi n, \texttt{even}\ n \Rightarrow \Sigma k, 2 \quad k = n$.

Solution Par induction sur `even` n:

- Cas even0: $n = 0$. On donne $k = 0$ et la proriétée $2 \quad 0 = 0$ est vérifiée.
- Cas evenS: $n = s(s(n'))$, hypothèse d'induction: $\exists k', 2 \quad k' = n'$. On doit montrer $\exists k, 2 \quad k = s(s(n'))$. Avec $k = s(k')$, on montre $2 \quad s(k') = s(k') + s(k') = s(s(k' + k')) = s(s(2 \quad k')) = s(s(n'))$.

Exercice 15. *Donner une définition* inductive *du prédicat In qui spécifie quand un élément appartient à une liste.*

Solution $In : List[\star] \to \star \to prop$

$$\begin{array}{rcl} \text{Inone} & : & In\ x\ [x] \\ \text{Inrec} & : & In\ x\ l \to In\ x\ (a :: l) \end{array}$$

Préfixes

On définit la relation inductive suivante qui caractérise les préfixes d'une liste.
prefix : $List[\star] \to List[\star] \to prop$

$$\begin{array}{rcl} prefixnil & : & \forall l, \text{prefix}\ []\ l \\ prefixskip & : & \forall x\ l\ l', \text{prefix}\ l\ l' \to \text{prefix}\ (x :: l)\ (x :: l') \end{array}$$

Une version non inductive peut-être donnée par:

$$\text{prefix}'(l, l') ::= \exists l'', l@l'' = l'$$

Exercice 16. *Énoncez et démontrez l'équivalence des deux définitions.*

Relation d'ordre et inversion

La définition inductive d'ordre stricte est:

$$\begin{array}{rcl} < & : & Nat \to Nat \to prop \\ lt0 & : & \forall x, 0 < s(x) \\ ltS & : & \forall x\ y, x < y \to s(x) < s(y) \end{array}$$

On rappelle que la définition de $<$ implique que $\neg \exists x, x < 0$. **Proof:** Supposons $x : Nat$ et une hypothèse $H : x < 0$. Par cas sur l'hypothèse H: aucun constructeur de la relation n'a pour conclusion $\_ < 0$, il n'y a donc aucun cas à considérer et la conclusion ($?$) est triviale. □

Exercice 17. *Par une analyse similaire, montrer que la relation inductive $<$ a la propriété: $\forall x\ y, s(x) < s(y) \to x < y$.*

Exercice 18. *Montrer que la relation inductive $<$ du cours est transitive par induction sur l'hypothèse $x < y$: $\forall x\ y, x < y \to \forall z, y < z \to x < z$. Notez bien l'utilisation du quantificateur imbriqué et soyez précis sur son utilisation dans la preuve.*

Familiarisation avec Coq

Rejouez la correction du TD1 avec coqide en salle de TP. Essayez de refaire certaines preuves.