TD 3 : Preuves de correction inductives

Matthieu Sozeau (matthieu.sozeau@inria.fr)

October 13, 2014

Ce TD porte sur la preuve de programmes spécifiés en logique du premier ordre. Les notes de cours et corrigés des TDs précédents sont disponibles sur la page du cours:

www.pps.univ-paris-diderot.fr/~sozeau/teaching/MVF-2014.fr.html

Des définition et propriétés utiles sont rappellées en annexe.

Ordres bien fondés

Exercice 1. *Donner une mesure sur les arbres binaires définis par les constructeurs:*

$$\begin{array}{lcl} \mathit{Leaf} & : & \mathit{Tree} \\ \mathit{Node} & : & \mathit{Tree} \quad \mathit{Nat} \quad \mathit{Tree} \; ! \; \mathit{Tree} \end{array}$$

Exercice 2. *Donner un prédicat* NoDup *caractérisant une liste sans duplicats à base de l'opérateur At.*

Solution

$$\begin{array}{rcl} \mathit{NoDup}(\grave{}) & := & 8i\ j\ 2\ \mathit{Nat}; i < j\grave{}j \wedge j < j\grave{}j \wedge : (i = j) \\ & &) \ : (\grave{}[i] = \grave{}[j]) \end{array}$$

Exercice 3. *Montrer* $\mathit{NoDup}(\grave{}) \ , \ \grave{} = [] \_ \ 9a \ \grave{}^{0}; \grave{} = a \ \grave{}^{0} \wedge : \mathit{Is\_in}(a; \grave{}^{0}) \wedge \mathit{NoDup}(\grave{}^{0})$.

Solution Par induction sur $\grave{}$.

- $\mathit{NoDup}([]) \quad 8i\ j\ 2\ \mathit{Nat}; i < 0 \wedge j < 0 \wedge : (i = j) \ ) \ : (\grave{}[i] = \grave{}[j]) \ , \ [] = [] : : :$. La prémisse de la partie gauche est équivalente à $?$, la propriété est donc vraie.
- $\mathit{NoDup}(a \ \grave{}^{0}) \quad 8i\ j\ 2\ \mathit{Nat}; i < j\grave{}^{0} j + 1 \wedge j < j\grave{}^{0} j + 1 \wedge : (i = j) \ )$ $: ((a \ \grave{}^{0})[i] = (a \ \grave{}^{0})[j])$. On a $\mathit{Is\_in}(a; \grave{}) \ , \ 9i < j\grave{}j; \grave{}[i] = a$, la partie

droite est donc équivalente à:

$$
\begin{array}{c}
: (9i < j\grave{}^{\theta}j; \grave{}[i] = a) \wedge NoDup(\grave{}^{\theta}) \\
, \ (8i < j\grave{}^{\theta}j; : (\grave{}^{\theta}[i] = a)) \wedge NoDup(\grave{}^{\theta}) \\
, \ (8i < j\grave{}^{\theta}j; 8j < j\grave{}^{\theta}j; : (i = j) \ ) \ : (\grave{}^{\theta}[i] = a) \wedge : (\grave{}^{\theta}[j] = a)) \wedge NoDup(\grave{}^{\theta}) \\
, \ (8i < j\grave{}^{\theta}j; 8j < j\grave{}^{\theta}j; : (i = j) \ ) \ : (\grave{}^{\theta}[i] = a) \wedge : (\grave{}^{\theta}[j] = a) \wedge : (\grave{}^{\theta}[i] = \grave{}^{\theta}[j])) \\
, \ (8i < j\grave{}^{\theta}j; 8j < j\grave{}^{\theta}j; : (i = j) \ ) \ : ((a \ \grave{}^{\theta})[i+1] = a) \wedge : ((a \ \grave{}^{\theta})[j+1] = a) \\
\wedge : (a \ \grave{}^{\theta})[i+1] = (a \ \grave{}^{\theta})[j+1])) \\
, \ (8i < ja \ \grave{}^{\theta}j; 8j < ja \ \grave{}^{\theta}j; : (i = j) \ ) \\
(i = 0 \wedge : ((a \ \grave{}^{\theta})[j] = (a \ \grave{}^{\theta})[0])) \_ (j = 0 \wedge : ((a \ \grave{}^{\theta})[i] = (a \ \grave{}^{\theta})[0])) \_ \\
(i > 0 \wedge j > 0 \wedge : (a \ \grave{}^{\theta})[j] = (a \ \grave{}^{\theta})[i])) \\
, \ NoDup(a \ \grave{}^{\theta})
\end{array}
$$

Dans la suite, on peut donc utiliser: $NoDup([])$ et $NoDup(a\ \grave{}^{\theta}) \ , \ : Is\_in(a; \grave{}^{\theta}) \wedge NoDup(\grave{}^{\theta})$.

Exercice 4. *Définir une fonction* remove_onedup *qui enlève d'une liste les éléments égaux à un élément donné.*

Solution

$$
\begin{array}{rcl}
remove\_onedup(a; []) & = & [] \\
remove\_onedup(a; b \ \ l) & = & \texttt{if } a = b \texttt{ then } remove\_onedup(a; l) \texttt{ else } b \ \ remove\_onedup(a; l)
\end{array}
$$

Exercice 5. *Montrer que la liste retournée par* remove_onedup *est de taille inférieure ou égale à la taille de la liste en entrée.*

Solution Par induction sur $\grave{}$ on a bien: $8a \ \grave{}; j remove\_onedup(a; \grave{})j \quad j\grave{}j$.

- Cas $\grave{} = []$: par simplification et réflexivité de
- Cas $\grave{} = b \ \ \grave{}^{\theta}$: par simplification et cas sur $a = b$. Les deux cas se montrent en utilisant l'hypothèse d'induction $j remove\_onedup(a; \grave{}^{\theta})j \quad j\grave{}^{\theta}j$ qui implique $1 + remove\_onedup(a; \grave{}^{\theta}) \quad 1 + j\grave{}^{\theta}j$.

Exercice 6. *Donner une spécification pour* remove_onedup *en termes du prédicat* Is_in *et montrez sa correction.*

Solution Spécification:

$$Spec\_onedup(a; \grave{}; \grave{}^{\theta}) := 8x; Is\_in(x; \grave{}^{\theta}) \ , \ (Is\_in(x; \grave{}) \ \wedge \ : (x = a))$$

Proof: Par induction sur $\grave{}$:

- Cas $\grave{} = []$, On doit montrer:

  $$Spec\_onedup(a; []; []) \ , \ 8x; Is\_in(x; []) \ , \ (Is\_in(x; []) \wedge : (x = a))$$

  Les deux applications du prédicats Is_in sont équivalentes à $?$, l'équivalence est triviale.

- Cas $` = b :: `'$. Hypothèse d'induction:

$$Spec\_onedup(a; `'; remove\_onedup(a; `'))$$
$$\forall x; Is\_in(x; remove\_onedup(a; `')) \;,\; (Is\_in(x; `') \wedge : (x = a))$$

On doit montrer:

$$Spec\_onedup(a; b :: `'; remove\_onedup(a; b :: `'))$$
$$\forall x; Is\_in(x; remove\_onedup(a; b :: `')) \;,\; (Is\_in(x; b :: `') \wedge : (x = a))$$

Par cas sur $a = b$:

{ $a = b$: On doit alors montrer:

$$\forall x; Is\_in(x; remove\_onedup(a; `')) \;,\; (Is\_in(x; a :: `') \wedge : (x = a))$$

On peut faire la transitivité avec l'hypothèse d'induction. Il nous reste à montrer:

$$\forall x; (Is\_in(x; `') \wedge : (x = a)) \;,\; (Is\_in(x; a :: `') \wedge : (x = a))$$

Par définition du prédicat Is_in:

$$Is\_in(x; a :: `') \wedge : (x = a) \;,$$
$$(x = a \vee Is\_in(x; `')) \wedge neg(x = a)$$
$$Is\_in(x; `') \wedge neg(x = a)$$

{ $: (a = b)$: On doit montrer:

$$\forall x; Is\_in(x; b :: remove\_onedup(a; `')) \;,\; (Is\_in(x; b :: `') \wedge : (x = a))$$

Encore un fois par définition de Is_in c'est équivalent à:

$$\forall x; (x = b \vee Is\_in(x; remove\_onedup(a; `'))) \;,\; ((x = b \vee Is\_in(x; `')) \wedge : (x = a))$$

Par l'hypothèse d'induction c'est équivalent à:

$$\forall x; (x = b \vee (Is\_in(x; `') \wedge : (x = a))) \;,\; ((x = b \vee Is\_in(x; `')) \wedge : (x = a))$$

C'est vrai par simple raisonnement logique. Les propositions $x = a$ et $x = b$ sont mutuellement exclusives par l'hypothèse $: (a = b)$, i.e.: $x = a \Rightarrow : (x = b)$ et $x = b \Rightarrow : (x = a)$. On utilise le principe logique suivant:
$\forall p\ p'\ q; (p \Rightarrow : p') \wedge (p' \Rightarrow : p) \Rightarrow (p \vee (q \wedge p')) \;,\; ((p \vee q) \wedge p')$

□

Exercice 7. *Définir une fonction* removedup *qui prend une liste en entrée et renvoie cette liste sans duplicats. Justifiez sa terminaison.*

Solution

$$\begin{aligned} \mathit{removedup}(a;[]) &= [] \\ \mathit{removedup}(a;b\ \ l) &= a\ \ \mathit{removedup}(\mathit{remove\_onedup}(a;l)) \end{aligned}$$

Exercice 8. *Donnez la spécification de* removedup *et prouvez sa correction.*

Solution Spécification:

$$\mathit{Spec\_removedup}(`;`^{0}) \ := \ \mathit{NoDup}(l^{0}) \wedge \forall x; \mathit{Is\_in}(x;`) \ , \ \mathit{Is\_in}(x;`^{0})$$

On montre: $\forall l; \mathit{Spec\_removedup}(`; \mathit{removedup}(`))$.

Proof: Par induction bien fondée sur $`$, en utilisant comme mesure la taille de la liste:

- Cas $` = []$, On doit montrer:

  $$\mathit{NoDup}(\mathit{removedup}([])) \wedge \forall x; \mathit{Is\_in}(x;[]) \ , \ \mathit{Is\_in}(x;\mathit{removedup}([]))$$

  Or on a: $\mathit{removedup}([]) = []$, il suffit donc de montrer:

  $$\mathit{NoDup}([]) \wedge \forall x; ? \ , \ ?$$

  En utilisant les lemmes sur NoDup.

- Cas $` = a\ \ `^{0}$. Hypothèse d'induction:

  $$\forall u; |u| < |`| \Rightarrow \mathit{NoDup}(\mathit{removedup}(u)) \wedge \forall x; \mathit{Is\_in}(x;u) \ , \ \mathit{Is\_in}(x;\mathit{removedup}(u))$$

  On doit montrer:

  $$\mathit{NoDup}(\mathit{removedup}(a\ `^{0})) \wedge \forall x; \mathit{Is\_in}(x;a\ `^{0}) \ , \ \mathit{Is\_in}(x;\mathit{removedup}(a\ `^{0}))$$

  Or on a: $\mathit{removedup}(a\ `^{0}) = a\ \ \mathit{removedup}(\mathit{remove\_onedup}(a;`^{0}))$, il suffit donc de montrer:

  $$\mathit{NoDup}(a\ \ \mathit{removedup}(\mathit{remove\_onedup}(a;`^{0}))) \quad (1) \ \wedge$$

  $$\forall x; \mathit{Is\_in}(x;a\ \ `^{0}) \ , \ \mathit{Is\_in}(x;a\ \ \mathit{removedup}(\mathit{remove\_onedup}(a;`^{0}))) \quad (2)$$

  On peut utiliser l'hypothèse d'induction sur $\mathit{remove\_onedup}(a;`^{0})$, puisque la taille de $\mathit{remove\_onedup}(a;`^{0})$ est inférieure ou égale à la taille de $`^{0}$ et donc strictement inférieure à la taille de $`$.

  { Par la définition de NoDup, (1) ,
  : $\mathit{Is\_in}(a;\mathit{removedup}(\mathit{remove\_onedup}(a;`^{0}))) \wedge \mathit{NoDup}(\mathit{removedup}(\mathit{remove\_onedup}(a;`^{0})))$

La deuxième partie de la conjonction découle directement de l'hypothèse d'induction. Pour la première partie, on raisonne par équivalence logique:

$$
\begin{array}{l}
: Is\_in(a; removedup(remove\_onedup(a; `^{\theta})))\\
\quad , \ : Is\_in(a; remove\_onedup(a; `^{\theta}))\\
\quad\quad , \ : (Is\_in(a; `^{\theta}) \ \wedge \ : (a = a))\\
\quad\quad\quad , \ : (Is\_in(a; `^{\theta}) \ \wedge \ ?)\\
\quad\quad\quad\quad , \ : ? \ , \ >
\end{array}
$$

{ Pour la partie (2) on suppose un x arbitraire et on raisonne aussi par équivalence,

$$
\begin{array}{l}
Is\_in(x; a \ \ removedup(remove\_onedup(a; `^{\theta})))\\
, \ x = a \_ \ Is\_in(x; removedup(remove\_onedup(a; `^{\theta})))\\
, \ x = a \_ \ Is\_in(x; remove\_onedup(a; `^{\theta})) \text{ (par HI)}\\
, \ x = a \_ \ (Is\_in(x; `^{\theta}) \ \wedge \ (: (x = a))) \text{ (par la spec de remove\_onedup)}\\
\quad , \ x = a \_ \ (Is\_in(x; `^{\theta})\\
\quad\quad , \ Is\_in(x; a \ \ `^{\theta})
\end{array}
$$

□

Une fonction de tri

On rappelle la spécification d'une fonction de tri:

$$Spec\_Sort(`; `^{\theta}) = Ordered(`^{\theta}) \ \wedge \ Ms(`) = Ms(`^{\theta})$$

On veut faire la preuve de la fonction de tri suivante:

$$
\begin{array}{rcl}
MinSort([]) & = & [] \\
MinSort(a \ \ `) & = & \texttt{let } (m; `_m) = Extract\_min(a; `) \texttt{ in } m \ \ MinSort(`_m)
\end{array}
$$

La spécification d'$Extract\_min$ est:

$$
\begin{array}{l}
Spec\_Extract\_min(a; `_1; m; `_2) =\\
\quad Is\_in(m; a \ \ `_1) \ \wedge\\
\qquad\quad 8x \ 2 \ ?: Is\_in(x; a \ \ `_1) \ ) \ m \quad x \ \wedge\\
\qquad\quad Ms(a \ \ `_1) = Sg(m) \ ] \ Ms(`_2)
\end{array}
$$

Exercice 9 (Minsort). *Vérifiez la correction du programme de tri $MinSort$ en fonction de la spécification d'$Extract\_min$.*

Solution Par induction bien fondée sur l:

- Cas $\grave{} = []$: $Ordered([]) \wedge Ms([]) = Ms([])$ trivial.
- Cas $\grave{} = a \; \grave{}_0$: Par induction: $\forall u; juj < j\grave{}j \Rightarrow Spec\_Sort(\grave{}_0; MinSort(\grave{}_0))$. On veut montrer $Spec\_Sort(a \; \grave{}_0; MinSort(a \; \grave{}_0))$.

  Par inspection de $MinSort$, on a un appel à $Extract\_min(a; \grave{}_0)$. On raisonne par rapport à la spécification d'$Extract\_min$ et on suppose $l; \grave{}_m$ tels que:

$$Is\_in(m; a \; \grave{}_0) \wedge \tag{1}$$
$$\forall x; Is\_in(x; a \; \grave{}_0) \Rightarrow m \quad x \wedge \tag{2}$$
$$Ms(a \; \grave{}_0) = Sg(m) \uplus Ms(\grave{}_m) \tag{3}$$

  Par induction on peut supposer $Spec\_Sort(\grave{}_m; MinSort(\grave{}_m))$ et on veut montrer $Spec\_Sort(a \; \grave{}_0; m \; MinSort(\grave{}_m))$, soit:

$$Ordered(m \; MinSort(\grave{}_m)) \wedge Ms(m \; MinSort(\grave{}_m)) = Ms(a \; \grave{}_0)$$

  Pour la partie droite de la conjonction on a:

$$\begin{aligned} Ms(m \; MinSort(\grave{}_m)) = Ms(a \; \grave{}_0) \;, & \\ \text{par (3)} \;, \quad & Ms(m \; MinSort(\grave{}_m)) = Sg(m) \uplus Ms(\grave{}_m) \\ \text{(par def de Ms)} \;, \quad & Sg(m) \uplus Ms(MinSort(\grave{}_m)) = Sg(m) \uplus Ms(\grave{}_m) \\ \text{(par HI)} \;, \quad & Sg(m) \uplus Ms(\grave{}_m) = Sg(m) \uplus Ms(\grave{}_m) \;, > \end{aligned}$$

  Pour la partie gauche, on doit montrer: $Ordered(m \; MinSort(\grave{}_m))$
  $\forall i \, j; i < j < jMinSort(\grave{}_m)j{+}1 \Rightarrow (m \; MinSort(\grave{}_m))[i] \quad (m \; MinSort(\grave{}_m)[j])$

  Par induction on a $Ordered(MinSort(\grave{}_m))$. Il suffit donc de montrer pour tout $0 < j < jMinSort(\grave{}_m)j + 1$ que

$$(m \; MinSort(\grave{}_m)[0]) = m \quad (m \; (MinSort(\grave{}_m)))[j]$$

  Or on a $\forall x; Is\_in(x; a \; l^{\emptyset}) \Rightarrow m \quad x$. C'est équivalent à $\forall x; Is\_in(x; m \; MinSort(\grave{}_m)) \Rightarrow m \quad x$ par la partie droite de la conjonction et le lemme 4 sur Is_in. Par définition, $Is\_in((m \; MinSort(\grave{}_m))[j]; m \; MinSort(\grave{}_m))$. La propriété est donc démontrée.

Exercice 10 (Extractmin). *Donnez une implémentation pour $Extract\_min$ et vérifiez sa correction.*

Solution Définition:

$$\begin{aligned} Extract\_min &: ? \quad List[\,] \; ! \; ? \quad List[?] \\ Extract\_min(a; []) &= (a; []) \\ Extract\_min(a; (b \; \grave{})) &= \texttt{if } a < b \texttt{ then let } (m; \grave{}_m) = Extract\_min(a; \grave{}) \texttt{ in } (m; b \; \grave{}_m) \\ &\quad \texttt{else let } (m; \grave{}_m) = Extract\_min(b; \grave{}) \texttt{ in } (m; a \; \grave{}_m) \end{aligned}$$

Proof: Par induction sur la liste $\grave{}$.

- Cas $\grave{} = []$. $Is\_in(a; a \quad []) \wedge a \quad \quad a \wedge Ms(a \quad []) = Sg(a) \;] \; Ms([])$. Par définition.
- Cas $\grave{} = b \quad \grave{}^{\theta}$. Par cas sur $a < b$:

  { $a < b$: On obtient récursivement $(m; \grave{}_m)$ avec par induction:

  $$Is\_in(m; a \quad \grave{}^{\theta}) \wedge \tag{4}$$
  $$8x; Is\_in(x; a \quad \grave{}^{\theta}) \;) \; m \quad x \wedge \tag{5}$$
  $$Ms(a \quad \grave{}^{\theta}) = Sg(m) \;] \; Ms(\grave{}_m) \tag{6}$$

  On veut montrer la propriété pour $(m; b \quad \grave{}_m)$.

  1. Pour montrer $Is\_in(m; a \quad b \quad \grave{}^{\theta})$ il nous suffit de $Is\_in(m; a \quad \grave{}^{\theta})$.
  2. Par transitivité, on a bien $m \quad x$ pour tout x tel que $Is\_in(x; a \quad (b \quad l^{\theta}))$ puisque $a < b$ et $m \quad x$ pour tout x tel que $Is\_in(x; a \quad l^{\theta})$.
  3. Pour l'égalité de multiensembles on montre:

  $$\begin{aligned} Ms(a \quad (b \quad \grave{}^{\theta})) &= Sg(a) \;] \; Sg(b) \;] \; Ms(\grave{}^{\theta}) \\ &= Sg(b) \;] \; Sg(a) \;] \; Ms(\grave{}^{\theta}) \\ &= Sg(b) \;] \; Sg(m) \;] \; Ms(\grave{}_m) \\ &= Sg(m) \;] \; Sg(b) \;] \; Ms(\grave{}_m) \\ &= Sg(m) \;] \; Ms(b \quad \grave{}_m) \end{aligned}$$

  { $a \quad b$: On obtient récursivement $(m; \grave{}_m)$ avec par induction:

  $$Is\_in(m; b \quad \grave{}^{\theta}) \wedge \tag{7}$$
  $$8x; Is\_in(x; b \quad \grave{}^{\theta}) \;) \; m \quad x \wedge \tag{8}$$
  $$Ms(b \quad \grave{}^{\theta}) = Sg(m) \;] \; Ms(\grave{}_m) \tag{9}$$

  On veut montrer la propriété pour $(m; a \quad \grave{}_m)$.

  1. Pour montrer $Is\_in(m; a \quad b \quad \grave{}^{\theta})$ il nous suffit de $Is\_in(m; a \quad \grave{}^{\theta})$.
  2. Par transitivité, on a bien $m \quad x$ pour tout x tel que $Is\_in(x; a \quad (b \quad \grave{}^{\theta}))$ puisque $b \quad a$ et $m \quad x$ pour tout x tel que $Is\_in(x; b \quad \grave{}^{\theta})$.
  3. Pour l'égalité de multiensembles on montre:

  $$\begin{aligned} Ms(a \quad (b \quad \grave{}^{\theta})) &= Sg(a) \;] \; Sg(b) \;] \; Ms(\grave{}^{\theta}) \\ &= Sg(a) \;] \; Sg(m) \;] \; Ms(\grave{}_m) \\ &= Sg(m) \;] \; Sg(a) \;] \; Ms(\grave{}_m) \\ &= Sg(m) \;] \; Ms(a \quad \grave{}_m) \end{aligned}$$

□

QuickSort – Devoir à rendre le 21 octobre

Pour le prochain cours, rédigez une preuve de la correction de la fonction *quicksort* (preuve qu'elle respecte bien la spécification d'une fonction de tri):

$$\begin{array}{rcl} qsort([]) & = & [] \\ qsort(a\ \grave{}) & = & \texttt{let}\ (\grave{}_1;\grave{}_2) = \texttt{split}\ (a;\grave{})\ \texttt{in} \\ & & \quad qsort(\grave{}_1)@(a\ \ qsort(\grave{}_2)) \end{array}$$

Vous aurez besoin de donner un type, une définition et une preuve de correction de la fonction *split* qui prend un élément x, une liste l et renvoie deux listes contenant respectivement les éléments inférieur ou égaux à x et les éléments supérieurs à x.

Annexe: définitions

Ordres

La relation sur les entiers est réflexive et transitive. La relation $<$ sur les entiers irréflexive et transitive. Pour ces deux relations, on suppose les lemmes usuels en relation avec l'addition et la multiplication.

Prédicat $Ordered : List[\ ]\ !\ prop$:

$$Ordered(\grave{}) = \forall i; j; 2\ Nat: (i < j < j\grave{}\ )\ \ \grave{}[i]\quad \grave{}[j])$$

Multiensembles

- $; =\ \ x\ 2\ ?: 0$
- $Sg(a) =\ \ x\ 2\ ?:\ \texttt{if}\ x = a\ \texttt{then}\ 1\ \texttt{else}\ 0$
- $M_1\ ]\ M_2 =\ \ x\ 2\ ?: M_1(x) + M_2(x)$

Propriétés:

- Neutral element: $;\ ]\ M = M\ ]\ ; = M$
- Commutativity: $M_1\ ]\ M_2 = M_2\ ]\ M_1$
- Associativity: $M_1\ ]\ (M_2\ ]\ M_3) = (M_1\ ]\ M_2)\ ]\ M_3$

Fonction Ms:

$$\begin{array}{rcl} Ms & : & List[\ ]\ !\ Multiset[\ ] \\ Ms([]) & = & ; \\ Ms(a\ \grave{}) & = & Sg(a)\ ]\ Ms(\grave{}) \end{array}$$

Propriétés:

- $Ms(`_1@`_2) = Ms(`_2@`_1) = Ms(`_1) \rbag Ms(`_2)$
- $Ms(Rev(`)) = Ms(`)$

Fonction Is_in:

$$\begin{array}{rcl} Is\_in & : & ? \quad List[?] \quad ! \quad Bool \\ Is\_in(a;`) & = & Ms(`)(a) > 0 \end{array}$$

Propriétés:

- $: (Is\_in(x;[]))$
- $8x\ l; Is\_in(x; a \quad `) \ , \ (x = a \_ Is\_in(x;`))$
- $Is\_in(a;`) \ , \ 9i < j`j; `[i] = a$
- $8l\ l'; Ms(l) = Ms(l') \ ) \ 8a; (Is\_in(a;l) \ , \ Is\_in(a;l'))$
- $8i < j`j; Is\_in(`[i];`)$

Equivalences logiques

- $\wedge$ est associatif, commutatif, $>$ est son élément neutre et $?$ absorbant (par exemple $p \wedge ? = ?$).
- $\_$ est associatif, commutatif, $?$ est son élément neutre et $>$ absorbant (par exemple $p \_ > = >$).
- $a \ ) \ b$ est équivalent à $: a \_ b$.
- $a \ , \ b$ est équivalent à $(a \ ) \ b) \wedge (b \ ) \ a)$.
- $: p$ est équivalent à $p \ ) \ ?$.
- $: > \ , \ ?$ et $: ? \ , \ >$.
- On a les lois de De Morgan:
  - $\{ \ : (a \wedge b) \ , \ : a \_ : b.$
  - $\{ \ : (a \_ b) \ , \ : a \wedge : b.$
  - $\{ \ : (9x; p) \ , \ 8x; : p.$
  - $\{ \ : (8x; p) \ , \ 9x; : p.$