Rapport du DM de Coq

Adrien BLASSIAU
Corentin JUVIGNY

30 octobre 2019

Avant de commencer les deux exercices, on importe tout d'abord le module `List` comme demandé. On importe aussi `Omega` qui fournit la tactique **omega** qui se révèle être très pratique pour la gestion des expressions arithmétiques. À noter que l'on peut très bien faire sans (en utilisant des lemmes du module `Arith.Gt`, `Arith.Lt`, etc) mais cela permet de réduire de quelques lignes les preuves ...

Les théorèmes admis (donc ceux que nous n'avons pas réussi à démontrer) sont notés en rouge.

1 Exercice 1 : Listes et comptage

Dans la suite on travaille avec un type A quelconque muni d'une égalité décidable. Pour les tests, on utilise nat (ils sont commentés dans le fichier .v).

`Variable` *A* : `Type`.

`Hypothesis` *dec_A* : ∀ (*x y* : *A*), ({*x*=*y*}+{~*x*=*y*}).

Question 1 occ prend en argument un élément x de type A et une liste l de type (list A) et retourne le nombre d'occurrences de x dans l. Elle a la signature suivante :

occ : *A* → *list A* → *nat*

Question 2 On démontre le théorème suivant :

`Theorem` *occ_app* : ∀ (*x* : *A*) *l1 l2*, *occ x* (*app l1 l2*) = (*occ x l1*) + (*occ x l2*).

Question 3 On démontre le théorème suivant :

`Theorem` *occ_filter* : ∀ (*P* : *A* → *bool*) (*a* : *A*) *l*, *occ a* (*filter P l*) = `if` (*P a*) `then` *occ a l* `else` 0.

Question 4 On démontre le théorème suivant :

`Theorem` *occ_map* : ∀ (*y* :*A*) (*z* :*A*), *y*≠*z* → ∃ (*f* : *A* → *A*) (*x* : *A*) (*l* : *list A*), *occ* (*f x*) (*map f l*) ≠ *occ x l*.

Question 5 On démontre le théorème de l'énoncé en deux temps, en utilisant un petit lemme intermédiaire :

`Lemma` *mem_diff* : ∀ (*l* : *list A*) (*x* : *A*) (*y* : *A*), *mem x* (*cons y l*) → *x* ≠ *y* → *mem x l*.

`Theorem` *mem_null_1* : ∀ (*l* : *list A*) (*x* : *A*), *occ x l* = 0 → ~(*mem x l*).

`Theorem` *mem_null_2* : ∀ (*l* : *list A*) (*x* : *A*), ~(*mem x l*) → *occ x l* = 0.

Question 6 On démontre le théorème de l'énoncé en deux temps. Nous ne sommes pas parvenus à démontrer l'un des sens du théorème :

`Theorem` *doublon_1* : ∀ (*l* : *list A*) (*x* : *A*), *nodup l* → *occ x l* ≤ 1.

`Theorem` *doublon_2* : ∀ (*l* : *list A*) (*x* : *A*), *occ x l* ≤ 1 → *nodup l*.

2 Exercice 2 : Implantation des multi-ensembles

Dans la suite on travaille avec un type T quelconque muni d'une égalité décidable. Pour les tests, on utilise nat (ils sont commentés dans le fichier .v).

`Variable` T : `Type`.

`Hypothesis` $T\_eq\_dec : \forall\ (x\ y : T), \{x{=}y\} + \{\tilde{}x{=}y\}$.

2.1 Implantation des multi-ensembles à l'aide de listes d'association

Question 1 On définit ici un multi-ensemble comme une liste de couples (élément, cardinalité), de la manière suivante :

`Definition` *multiset* := *list* ($T \times nat$).

Voici les signatures des différentes fonctions demandées dans l'énoncé :

empty : *multiset*

singleton : $T \to$ *multiset*

add : $T \to nat \to multiset \to multiset$

member : $T \to multiset \to bool$

union : $multiset \to multiset \to multiset$

multiplicity : $T \to multiset \to nat$

removeOne : $T \to multiset \to multiset$

removeAll : $T \to multiset \to multiset$

Question 2

Question 2. a) On définit le prédicat `InMultiset` de la manière suivante :

`Inductive` *InMultiset* (x :T) (l :*multiset*) : `Prop` :=
| *inMultiset_intro* : *member* x l = *true* $\to$ *InMultiset* x l.

Question 2. b) On définit le prédicat `wf` de la manière suivante :

`Inductive` `wf` (l : *multiset*) : `Prop` :=
| *wf_intro* : ($\forall$ x, (*InMultiset* x (*removeAll* x l) $\to$ *False*) $\wedge$ (*member* x l = *true* $\to$ (*multiplicity* x l) > 0)) $\to$ `wf` l.

Question 2. c) On démontre maintenant la bonne formation de nos différentes fonctions. 2 lemmes intermédiaires ont été créés.
Nous ne sommes pas parvenus à démontrer cette propriété pour 4 fonctions :

`Theorem` *empty_wf* : `wf` *empty*.

`Theorem` *singleton_wf* : $\forall$ (*x* :*T*), `wf` (*singleton x*).

`Lemma` *plus_n_0* : $\forall$ *n*, *n* + *O* = *n*.

`Lemma` *wf_plus_n* : $\forall$ *t n0 n l*, `wf` ((*t*, *n0*) : : *l*) $\rightarrow$ `wf` ((*t*, *n0*+*n*) : : *l*).

`Theorem` *add_wf* : $\forall$ (*x* :*T*) (*n* :*nat*) (*l* : *multiset*), `wf` *l* $\rightarrow$ `wf` (*add x n l*).

`Theorem` *union_wf* : $\forall$ (*l* : *multiset*) (*l'* : *multiset*), `wf` *l* $\rightarrow$ `wf` *l'* $\rightarrow$ `wf` (*union l l'*).

`Theorem` *removeOne_wf* : $\forall$ (*x* : *T*) (*l* : *multiset*), `wf` *l* $\rightarrow$ `wf` (*removeOne x l*).

`Theorem` *removeAll_wf* : $\forall$ (*x* : *T*) (*l* : *multiset*), `wf` *l* $\rightarrow$ `wf` (*removeAll x l*).

Question 3 On démontre maintenant les différents théorèmes indiqués dans l'énoncé. 1 lemme intermédiaire a été créé.
Nous ne sommes pas parvenus à démontrer les deux derniers théorèmes :

`Theorem` *proof_1_1* : $\forall$ (*x* : *T*), $\neg$*InMultiset x empty*.

`Theorem` *proof_2_1* : $\forall$ *x y* , *InMultiset y* (*singleton x*) $\leftrightarrow$ *x* = *y*.

`Theorem` *proof_3_1* :$\forall$ *x*, *multiplicity x* (*singleton x*) = 1.

`Theorem` *proof_4_1* : $\forall$ *x s*, `wf` *s* $\rightarrow$ (*member x s* = *true* $\leftrightarrow$ *InMultiset x s*).

`Theorem` *proof_5_1* : $\forall$ *x n s*, *n* > 0 $\rightarrow$ *InMultiset x* (*add x n s*).

`Theorem` *proof_6_1* : $\forall$ *x n y s*, *x* $\neq$ *y* $\rightarrow$ (*InMultiset y* (*add x n s*) $\leftrightarrow$ *InMultiset y s*).

`Lemma` *proof_7_1_1* : $\forall$ *x s*, *multiplicity x s* $\neq$ 0 $\rightarrow$ *InMultiset x s*.

`Theorem` *proof_7_1_2* : $\forall$ *x s*, `wf` *s* $\rightarrow$ (*multiplicity x s* = 0 $\leftrightarrow$ $\neg$*InMultiset x s*).

`Theorem` *proof_8_1* : $\forall$ *x n s*, *multiplicity x* (*add x n s*) = *n* + (*multiplicity x s*).

`Theorem` *proof_9_1* : $\forall$ *x n y s*, *x* $\neq$ *y* $\rightarrow$ `wf` *s* $\rightarrow$*multiplicity y* (*add x n s*) = *multiplicity y s*.

`Theorem` *proof_10_1* : $\forall$ *s t x*, `wf` *s* $\rightarrow$ `wf` *t* ->(*InMultiset x* (*union s t*) $\leftrightarrow$ *InMultiset x s* $\vee$ *InMultiset x t*).

Question 4 On démontre maintenant quelques théorèmes supplémentaires sur `removeOne` et `removeAll`.

`Theorem` *proof_11_1* : $\forall$ *x*, *multiplicity x* (*removeOne x* (*singleton x*)) = 0.

`Theorem` *proof_12_1* : $\forall$ *x*, *multiplicity x* (*removeAll x* (*singleton x*)) = 0.

`Theorem` *proof_13_1* : $\forall$ *x l n*, *multiplicity x l* = *n* $\rightarrow$ *n* > 1 $\rightarrow$ *multiplicity x* (*removeOne x l*) = *n*-1.

`Theorem` *proof_14_1* : $\forall$ *x l*, `wf` *l* $\rightarrow$ ˜(*InMultiset x* (*removeAll x l*)).

`Theorem` *proof_15_1* : $\forall$ *x l*, *multiplicity x l* > 1 $\rightarrow$ *InMultiset x* (*removeOne x l*).

2.2 Implantation Fonctionnelle des multi-ensembles

Question 1 On définit ici un multi-ensemble comme une fonction qui encode les multiplicités, de la manière suivante :

`Definition` $multiset\_2 := T \to nat$.

Voici les signatures des différentes fonctions demandées dans l'énoncé :

$empty\_2 : multiset\_2$

$singleton\_2 : T \to multiset\_2$

$add\_2 : T \to nat \to multiset\_2 \to multiset\_2$

$member\_2 : T \to multiset\_2 \to bool$

$union\_2 : multiset\_2 \to multiset\_2 \to multiset\_2$

$multiplicity\_2 : T \to multiset\_2 \to nat$

$removeOne\_2 : T \to multiset\_2 \to multiset\_2$

$removeAll\_2 : T \to multiset\_2 \to multiset\_2$

Question 2 On définit le prédicat `InMultiset_2` de la manière suivante :

`Inductive` $InMultiset\_2\ (x : T)\ (l : multiset\_2)$: `Prop` :=
| $inMultiset\_2\_intro : member\_2\ x\ l = true \to InMultiset\_2\ x\ l$.

Question 3 On démontre maintenant les différents théorèmes indiqués dans l'énoncé ainsi que ceux sur les fonctions `removeOne` et `removeAll`. 1 lemme intermédiaire a été créé.

`Theorem` $proof\_1\_2 : \forall\ (x : T), \neg InMultiset\_2\ x\ empty\_2$.

`Theorem` $proof\_2\_2 : \forall\ x\ y\ , InMultiset\_2\ y\ (singleton\_2\ x) \leftrightarrow x = y$.

`Theorem` $proof\_3\_2 : \forall\ x, multiplicity\_2\ x\ (singleton\_2\ x) = 1$.

`Theorem` $proof\_4\_2 : \forall\ x\ s, member\_2\ x\ s = true \leftrightarrow InMultiset\_2\ x\ s$.

`Theorem` $proof\_5\_2 : \forall\ x\ n\ s, n > 0 \to InMultiset\_2\ x\ (add\_2\ x\ n\ s)$.

`Theorem` $proof\_6\_2 : \forall\ x\ n\ y\ s, x \neq y \to (InMultiset\_2\ y\ (add\_2\ x\ n\ s) \leftrightarrow InMultiset\_2\ y\ s)$.

`Lemma` $proof\_7\_2\_1 : \forall\ x\ s, s\ x \neq 0 \to InMultiset\_2\ x\ s$.

`Theorem` $proof\_7\_2\_2 : \forall\ x\ s, multiplicity\_2\ x\ s = 0 \leftrightarrow \neg InMultiset\_2\ x\ s$.

`Theorem` $proof\_8\_2 : \forall\ x\ n\ s, multiplicity\_2\ x\ (add\_2\ x\ n\ s) = n + (multiplicity\_2\ x\ s)$.

`Theorem` $proof\_9\_2 : \forall\ x\ n\ y\ s, x \neq y \to multiplicity\_2\ y\ (add\_2\ x\ n\ s) = multiplicity\_2\ y\ s$.

`Theorem` $proof\_10\_2 : \forall\ s\ t\ x, (InMultiset\_2\ x\ (union\_2\ s\ t) \leftrightarrow InMultiset\_2\ x\ s \lor InMultiset\_2\ x\ t)$.

`Theorem` $proof\_11\_2 : \forall\ x, multiplicity\_2\ x\ (removeOne\_2\ x\ (singleton\_2\ x)) = 0$.

`Theorem` $proof\_12\_2 : \forall\ x, multiplicity\_2\ x\ (removeAll\_2\ x\ (singleton\_2\ x)) = 0$.

`Theorem` $proof\_13\_2 : \forall\ x\ l\ n, multiplicity\_2\ x\ l = n \to n > 1 \to multiplicity\_2\ x\ (removeOne\_2\ x\ l) = n\text{-}1$.

Theorem $proof\_14\_2 : \forall\ x\ l,\ \tilde{}(InMultiset\_2\ x\ (removeAll\_2\ x\ l)).$

Theorem $proof\_15\_2 : \forall\ x\ l,\ multiplicity\_2\ x\ l > 1 \rightarrow InMultiset\_2\ x\ (removeOne\_2\ x\ l).$