Interactive Proof Tools
Assignment 5

Hans de Nivelle, Patrick Maier

INFORMATIK

http://www.mpi-sb.mpg.de/~nivelle/teaching/intprooftools2003/main.html

Exercise 5.1 Recall the formalization of multisets from exercise 4.1. A multiset s is *finite* iff the set of its members is finite. This is formalized by the PVS theory `finite_multiset_defs` below. That theory also defines a cardinality function `size` for finite multisets by summing up the multiplicities of the members.

1. Typecheck the theory `finite_multiset_defs`, i. e., prove the TCCs.
2. Prove that every submultiset of a finite multiset is finite.
3. Prove that the operators `empty` and `singleton` yield finite multisets.
4. Prove that the operators `union`, `add` and `remove` take finite multisets to finite multisets.
5. Prove that `intersection` yields a finite multiset if one of its arguments is finite. Prove that `difference` yields a finite multiset if its first argument is finite.

```
finite_multiset_defs[T: TYPE]: THEORY
BEGIN
  IMPORTING multiset_defs[T]
  IMPORTING finite_sets@finite_sets_sum_real[T]

  n: VAR nat
  x: VAR T
  s: VAR multiset[T]

  members(s): [T -> bool] = LAMBDA x: member(x,s)

  is_finite(s): bool = finite_sets_def.is_finite(members(s))

  finite_multiset: TYPE = (is_finite) CONTAINING empty[T]

  fs: VAR finite_multiset

  size(fs): nat = sum(members(fs),fs)
END finite_multiset_defs
```

Hint: Have a look into the prelude where finite sets are defined.

Exercise 5.2 In order to work with finite multisets one needs induction principles for them. Prove the following two induction principles for finite multisets:

```
finite_multiset_induction: THEOREM
  FORALL (P: [finite_multiset[T] -> bool]):
    P(empty) AND
    (FORALL x, fs: P(fs) IMPLIES P(add(x,fs))) IMPLIES
      FORALL ft: P(ft)

finite_multiset_induction_union: THEOREM
  FORALL (P: [finite_multiset[T] -> bool]):
    P(empty) AND
    (FORALL x: P(singleton(x))) AND
    (FORALL fs, ft: P(fs) AND P(ft) IMPLIES P(union(fs,ft))) IMPLIES
      FORALL ft: P(ft)
```

Hints:

- Do `measure-induct` and use `size` as the measure.
- The following lemmata[1] may be helpful:

```
x: VAR T
fs, ft, ft1, ft2: VAR finite_multiset[T]

empty_or_add: LEMMA
  fs = empty OR
  (EXISTS x, ft: fs = add(x,ft))

empty_or_singleton_or_union: LEMMA
  fs = empty OR
  (EXISTS x: fs = singleton(x)) OR
  (EXISTS ft1, ft2: ft1 /= empty AND ft2 /= empty AND fs = union(ft1,ft2))

size_empty:     LEMMA size(empty) = 0
size_singleton: LEMMA size(singleton(x)) = 1
size_union:     LEMMA size(union(fs,ft)) = size(fs) + size(ft)
size_add:       LEMMA size(add(x,fs)) = size(fs) + 1
```

Exercise 5.3 In exercise 4.2, you have proven that `mergesort` takes a list `xs` to a sorted list `ys`. To complete the verification of `mergesort` it remains to be shown that the list `ys` contains exactly the same members than `xs`.

1. Prove `mergesort_member: THEOREM member(x, mergesort(xs)) = member(x,xs)`.

[1] Some of these lemmata are very hard to prove. If you do not succeed in proving them you may just use them as if they were axioms.

2. Prove `mergesort_length: THEOREM length(mergesort(xs)) = length(xs)`.

Hints:

- Use `measure_induct`.
- You may have to prove lemmata about how the 'subroutines' `split` and `merge` behave with respect to `member` and `length`.

Exercise 5.4 Prove the following formulas using Coq.

1. $A \vee (A \rightarrow B)$.
2. $(A \rightarrow B) \leftrightarrow (\neg A \vee B)$.
3. $(\forall x{:}X\ P(x) \vee \forall x{:}X\ Q(x)) \rightarrow (\forall x{:}X\ P(x) \vee Q(x))$.

Hint: Some of these formulas are only classically valid. Check the Coq tutorial for examples how to prove classical formulas by using the law of excluded middle as an extra axiom.

Exercise 5.5 Using the induction rule for $\leq$ from the slides, prove the following facts in natural deduction:

- $\forall x, y, z : \mathrm{Nat}\ x \leq y \wedge y \leq z \rightarrow x \leq z$.
  Hint: Prove $\forall x, y : \mathrm{Nat}\ x \leq y \rightarrow \forall z : \mathrm{Nat}\ (y \leq z \rightarrow x \leq z)$.
- $\forall x, y : \mathrm{Nat}\ x \leq y \vee y \leq x$.
  Hint: This is proven by Nat-induction (not $\leq$-induction) over Nat.
  You need the additional lemma $\forall x, y : \mathrm{Nat}\ x \leq y \rightarrow \mathrm{succ}(x) \leq y \vee x \approx y$.