Interactive Proof Tools
Assignment 3

Hans de Nivelle, Patrick Maier

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

Exercise 3.1 Prove the following formulas in intuitionistic Natural Deduction, i. e., construct ND derivations without assumptions.

1. $(A \vee A) \leftrightarrow (A \vee \bot)$
2. $(B \wedge A) \leftrightarrow ((A \wedge B) \wedge A)$
3. $(A \to B) \to \neg(A \wedge \neg B)$
4. $\neg(A \wedge \neg A)$
5. $(\neg\neg\neg A) \to \neg A$
6. $(A \to B \to C) \leftrightarrow (A \wedge B \to C)$
7. $(\exists y{:}Y\ \forall x{:}X\ R(x,y)) \to (\forall x{:}X\ \exists y{:}Y\ R(x,y))$

Exercise 3.2 Prove the formula $x + s(y) = s(y) + x$ in intuitionistic Natural Deduction from the premisses

$$\begin{gathered} x + y = y + x, \\ \forall x, y{:}\mathrm{Nat}\ x + s(y) = s(x + y), \\ \forall x, y{:}\mathrm{Nat}\ s(x) + y = s(x + y). \end{gathered}$$

Exercise 3.3 Prove the theorem `merge_sorted` below; it states that the `merge` function applied to sorted lists yields a sorted list.

```
sorted[t: TYPE, (IMPORTING orders[t]) <=: (total_order?)]: THEORY
BEGIN
  x, y, z: VAR t
  xs, ys, zs: VAR list[t]

  sorted?(xs): INDUCTIVE bool =
    CASES xs OF
      null: TRUE,
      cons(y,ys): CASES ys OF
                    null: TRUE,
                    cons(z,zs): y <= z AND sorted?(ys)
                  ENDCASES
```

```
    ENDCASES
END sorted


merge[t: TYPE, (IMPORTING orders[t]) <=: (total_order?)]: THEORY
BEGIN
  IMPORTING sorted[t, <=]

  u, v, x, y, z: VAR t
  us, vs, xs, ys, zs: VAR list[t]

  merge(xs,ys): RECURSIVE list[t] =
    CASES xs OF
      null: ys,
      cons(u,us): CASES ys OF
                    null: xs,
                    cons(v,vs): IF u <= v
                                  THEN cons(u,merge(us,ys))
                                  ELSE cons(v,merge(xs,vs))
                                ENDIF
                  ENDCASES
    ENDCASES
  MEASURE LAMBDA xs,ys: length(xs) + length(ys)

  merge_sorted: THEOREM
    sorted?(xs) AND sorted?(ys) IMPLIES sorted?(merge(xs,ys))
END merge
```

Hints: I solved this by defining a predicate `leqall?(x: t, ys: list[t]): bool` which is true iff `x` is less than or equals each element in `ys`. Then I have proven the following Lemmata:

```
  leqall_mono: LEMMA
    x <= y AND leqall?(y,zs) IMPLIES leqall?(x,zs)

  sorted_leqall: LEMMA
    sorted?(cons(x,xs)) = (leqall?(x,xs) AND sorted?(xs))

  merge_leqall: LEMMA
    leqall?(x,ys) AND leqall?(x,zs) IMPLIES leqall?(x, merge(ys,zs))
```