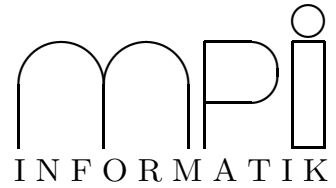




## Interactive Proof Tools Assignment 1

Hans de Nivelle, Patrick Maier



---

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

---

**Exercise 1.1** Give proofs for the following formulas in Sequent Calculus.

1.  $A \vee (A \rightarrow B)$ .
2.  $(A \rightarrow B) \leftrightarrow (\neg A \vee B)$ .
3.  $((A \wedge B) \vee (C \wedge D)) \rightarrow ((A \vee C) \wedge (B \vee D))$ .
4.  $\exists x:X \neg P(x) \vee \forall x:X P(x)$ .
5.  $(\exists y:Y \forall x:X R(x, y)) \rightarrow (\forall x:X \exists y:Y R(x, y))$ .
6.  $(\forall x:X P(x) \vee \forall x:X Q(x)) \rightarrow (\forall x:X P(x) \vee Q(x))$ .
7.  $(\forall x, y:X P(x) \vee Q(y)) \vee (\forall y, z:X Q(y) \vee T(z)) \rightarrow (\forall x, y, z:X P(x) \vee Q(y) \vee T(z))$ .

**Exercise 1.2** Use PVS to prove the formulas from exercise 1.1. Use only the commands `case`, `flatten`, `split`, `propax`, `instantiate` and `skolem`. These are the basic commands that implement the Sequent Calculus of PVS<sup>1</sup>, see chapter 3 of the PVS Prover Guide. Which commands correspond to which rules of the calculus?

**Exercise 1.3** Splitting a list into two parts in PVS.

1. Define a function `split(n, xs)` which takes a natural number `n` and a list `xs` and returns a pair of lists `(ys, zs)` such that `ys` is the prefix of `xs` of length `n` and `zs` is the rest of `xs`.  
**Hint:** You may assume that `n ≤ length(xs)` whenever `split(n, xs)` is called.
2. Prove that appending the results of splitting gives back the original list, i. e., prove the equation `app(split(n, xs)) = xs`.  
**Hint:** You may have to induct over both variables `n` and `xs` (one after the other).
3. Prove that `split(n, xs) = (ys, zs)` implies `length(ys) = n`.
4. Argue (informally) that the formal statements 2 and 3 above are just a reformulation of the informal specification 1 of `split`.

---

<sup>1</sup>PVS treats equality and the conditional IF on the level of the calculus, which is reflected by the commands `replace` and `lift-if`. The command `lemma` provides a variant of the Cut rule, with the proof of one branch deferred.