University of Cambridge
Computer Laboratory
Dr Tjark Weber

Easter Term 2010/11
Exercises 2
May 6, 2011

Interactive Formal Verification (L21)

1 Power, Sum

Power

▷ Define a (primitive recursive) function `pow x n` that computes x^n on natural numbers.

`pow :: "nat ⇒ nat ⇒ nat"`

▷ Prove the well known equation $x^{m \cdot n} = (x^m)^n$:

theorem `pow_mult: "pow x (m * n) = pow (pow x m) n"`

Hint: prove a suitable lemma first. If you need to appeal to associativity and commutativity of multiplication: the corresponding simplification rules are named `mult_ac`.

Summation

▷ Define a (primitive recursive) function `sum ns` that sums a list of natural numbers: $\mathit{sum}\ [n_1, \ldots, n_k] = n_1 + \cdots + n_k$.

`sum :: "nat list ⇒ nat"`

▷ Show that `sum` is compatible with `rev`. You may need a lemma.

theorem `sum_rev: "sum (rev ns) = sum ns"`

▷ Define a function `Sum f k` that sums f from 0 up to $k-1$: $\mathit{Sum}\ f\ k = f\ 0 + \cdots + f(k-1)$.

`Sum :: "(nat ⇒ nat) ⇒ nat ⇒ nat"`

▷ Show the following equations for the pointwise summation of functions. Determine first what the expression `whatever` should be.

theorem `"Sum (λi. f i + g i) k = Sum f k + Sum g k"`
theorem `"Sum f (k + l) = Sum f k + Sum whatever l"`

▷ What is the relationship between *sum* and *Sum*? Prove the following equation, suitably instantiated.

theorem `"Sum f k = sum whatever"`

Hint: familiarize yourself with the predefined functions `map` and `[i..<j]` on lists in theory `List`.