Interactive Formal Verification (L21)

1 Power, Sum

Power

▷ Define a (primitive recursive) function $\text{pow } x \ n$ that computes $x^n$ on natural numbers.

fun pow :: "nat ⇒ nat ⇒ nat" where
  "pow x 0 = Suc 0"
| "pow x (Suc n) = x * pow x n"

▷ 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.

lemma pow_add: "pow x (m + n) = pow x m * pow x n"
  apply (induct n)
  apply auto
  done

theorem pow_mult: "pow x (m * n) = pow (pow x m) n"
  apply (induct n)
  apply (auto simp add: pow_add)
  done

Summation

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

fun sum :: "nat list ⇒ nat" where
  "sum [] = 0"
Show that `sum` is compatible with `rev`. You may need a lemma.

```isar
define `Sum f k` that sums `f` from 0 up to `k−1`: `Sum f k = f 0 + ··· + f(k−1).

```isar```isar```