2 Power, Sum

2.1 Power

Define a primitive recursive function *pow x n* that computes x^n on natural numbers.

consts
  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*.

2.2 Summation

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

consts
  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 $k - 1$: $Sum\ f\ k = f\ 0 + \cdots + f(k-1)$.

consts
  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* (%*i*. *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.