Australian National University

COMP4011/8011 Advanced Topics in Formal Methods and Programming Languages

– Software Verification with Isabelle/HOL –

Peter Höfner

August 18, 2024

Section 12

Induction

Structural induction

$P\ xs$ holds for all lists xs if

- P Nil
- and for arbitrary x and xs, $P\ xs \Longrightarrow P\ (x\#xs)$
  Induction theorem **list.induct:**
  $[\![P\ [];\ \bigwedge a\ list.\ P\ list \Longrightarrow P\ (a\#list)]\!] \Longrightarrow P\ list$
- General proof method for induction: **(induct x)**
  - x must be a free variable in the first subgoal.
  - type of x must be a datatype.

Basic heuristics

Theorems about recursive functions are proved by induction

Induction on argument number i of f
if f is defined by recursion on argument number i

Example

A tail recursive list reverse:

primrec itrev :: 'a list $\Rightarrow$ 'a list $\Rightarrow$ 'a list
where
itrev [] $ys = ys$ |
itrev $(x\#xs)$ $ys =$ itrev xs $(x\#ys)$

lemma itrev xs [] $=$ rev xs

Demo

Australian National University

Proof Attempt

Generalisation

Replace constants by variables

lemma itrev $xs\ ys = $ rev $xs@ys$

Quantify free variables by $\forall$

(except the induction variable)

lemma $\forall ys.$ itrev $xs\ ys = $ rev $xs@ys$

Or: **apply (induct xs arbitrary: ys)**

Exercises

- define a primitive recursive function **lsum** :: nat list $\Rightarrow$ nat
  that returns the sum of the elements in a list.
- show "$2 * \text{lsum } [0.. < \mathit{Suc}\ n] = n * (n + 1)$"
- show "$\text{lsum } (\text{replicate } n\ a) = n * a$"
- define a function **lsumT** using a tail recursive version of listsum.
- show that the two functions are equivalent: lsum *xs* = lsumT *xs*