Isa-Test

By Yannick

August 5, 2012

Contents

1 Test explicit proof and presentation document with Isabelle.

theory *Isa-Test*
imports *Main*
begin

2 Definition of *fun sep*

fun $sep :: 'a \Rightarrow 'a\ list \Rightarrow 'a\ list$
where
$sep\ s\ (x\ \#\ y\ \#\ xs) = x\ \#\ s\ \#\ (sep\ s\ (y\ \#\ xs))$
$|\ sep\ s\ xs = xs$

value $sep\ s\ [a,b,c]$
value $sep\ s\ [a]$ — Unchanged
value $sep\ s\ []$ — Idem

3 Automated proof of a lemma about *fun sep*

lemma $map\ f\ (sep\ s\ xs) = sep\ (f\ s)\ (map\ f\ xs)$

apply (*induct s xs rule*: *sep.induct*)
apply *auto*
done

Easy, nice for quick modelling, shows how much your computer and Isabelle are clever, but does not help to be aware of what's involved.

4 Explicit manual proof of the same lemma

4.1 Reminder of some of the involved rules

$sep.simps(1)$: $sep\ ?s\ (?x\ \#\ ?y\ \#\ ?xs) = ?x\ \#\ ?s\ \#\ sep\ ?s\ (?y\ \#\ ?xs)$

$sep.simps(2)$: $sep\ ?s\ [] = []$

$sep.simps(3)$: $sep\ ?s\ [?v] = [?v]$

$sep.induct(1)$: $[\![\bigwedge s\ x\ y\ xs.\ ?P\ s\ (y\ \#\ xs) \Longrightarrow ?P\ s\ (x\ \#\ y\ \#\ xs);\ \bigwedge s.\ ?P\ s\ [];\ \bigwedge s\ v.\ ?P\ s\ [v]]\!] \Longrightarrow ?P\ ?a0.0\ ?a1.0$

$map.simps(1)$: $map\ ?f\ [] = []$

$map.simps(2)$: $map\ ?f\ (?x\ \#\ ?xs) = ?f\ ?x\ \#\ map\ ?f\ ?xs$

4.2 Lemma's proof

lemma $map\ f\ (sep\ s\ xs) = sep\ (f\ s)\ (map\ f\ xs)$
proof (*induct s xs rule*: *sep.induct*)
case (*1 s x y xs*)

— A strategy comes from the observation that the sole difference
— between the hypothesis and the conclusion of that induction
— step, is that the conclusion just have $(f\ x)\ \#\ (f\ s)$
— prepended to both sides of its equality.

— We will turn $A = B \Longrightarrow C = D$ into
— $A = B \Longrightarrow (E\ A) = (E\ B)$, whose proof is easy.

let $?A1 = sep\ s\ (y\ \#\ xs)$ — Subexpression of A
let $?B1 = map\ f\ (y\ \#\ xs)$ — Subexpression of B
let $?C1 = sep\ s\ (x\ \#\ y\ \#\ xs)$ — Subexpression of C
let $?D1 = map\ f\ (x\ \#\ y\ \#\ xs)$ — Subexpression of D
let $?E = \lambda xs.\ (f\ x)\ \#\ (f\ s)\ \#\ xs$

assume $(map\ f\ ?A1) = (sep\ (f\ s)\ ?B1)$ (**is** $?A = ?B$)
show $(map\ f\ ?C1) = (sep\ (f\ s)\ ?D1)$ (**is** $?C = ?D$)
proof −
have *1*: $?C = ?E\ ?A$
proof −

```
      have ?C1 = x # s # ?A1 by simp
      then have map f ?C1 = (f x) # (f s) # (map f ?A1) by simp
      then have ?C = (f x) # (f s) # ?A by simp
      then show ?C = ?E ?A by simp
    qed
  have 2: ?D = ?E ?B
    proof –
      — For the second step, note that B1 is not empty
      have ?D1 = (f x) # ?B1 by simp
      then have sep (f s) ?D1 = (f x) # (f s) # (sep (f s) ?B1) by simp
      then have ?D = (f x) # (f s) # ?B by simp
      then show ?D = ?E ?B by simp
    qed
  show ?C = ?D using ‹?A = ?B› and 1 and 2 by simp
  qed
next
case (2-1 s)
  show map f (sep s []) = sep (f s) (map f []) (is ?A = ?B)
  proof –
    let ?C = []
    have 1: map f (sep s []) = [] (is ?A = ?C) by simp
    have 2: sep (f s) (map f []) = [] (is ?B = ?C) by simp
    show ?A = ?B using 1 and 2 by simp
  qed
next
case (2-2 s e)
  show map f (sep s [e]) = sep (f s) (map f [e]) (is ?A = ?B)
  proof –
    let ?C = map f [e]
    have 1: map f (sep s [e]) = map f [e] (is ?A = ?C) by simp
    have 2: sep (f s) (map f [e]) = map f [e] (is ?B = ?C) by simp
    show ?A = ?B using 1 and 2 by simp
  qed
qed
```

4.3 Comments about the style

May looks too much verbose, but the proof structure allows to easily skeep over details. Furthermore, you can benefit from that verbosity to figure some generic strategy, especially if you make good usage of schematic variables.

The first case, maps the assumption and conclusion to two patterns, which are $?A = ?B$ and $?C = ?D$. The four schematics variables are in turn subject to extraction of subexpressions; but those subexpressions, which are *A1*, *B1*, *C1* and *C1*, are defined before *A*, *B*, *C* and *D*, which may looks surprising.

The second and third case, make use of pattern matching too, but in the purpose of annotations. This is readable, but not formally clean, as the intent is rather to define $?C$ and then check that it is indeed matched in $?A$

$= ?C$ and $?B = ?C$, but it formally assignes it instead.

For readability, all cases starts with an optional *assume* and a not optional *show*, as on their own, the case statement are not verbose at all when viewed in a PDF document.

Bad page break makes that document hugly. Also, indentation guide, present during document authoring, and which help readability, are not there anymore.

end