Formalizing Syntactic Cut-elimination for $\supset$-fragment of LJ in Isabelle/HOL

Revantha Ramanayake

May 25, 2007
First attempt — top-down approach

State the theorem in Isabelle, prove required lemmas as and when needed

Hard to figure out what to do when stuck

• Prove a new lemma?

• Is Isabelle not seeing something obvious?

• A goal with many assumptions can appear to be (visually) complicated
Second attempt — bottom-up approach

Closely follow paper proof (using Isabelle as a proof checker)

Each line in paper proof corresponds to a block of Isabelle script

paper proof $\rightarrow$ pre-script $\rightarrow$ Isabelle script

Since the step from paper proof line $n \mapsto n + 1$ is usually trivial first-order reasoning, it should be trivial in Isabelle too

(modus ponens, replacing $\forall x. P(x)$ in assumption with $P(t)$, explicitly instantiating an existential operator, fixing $n$)
Of course, setting up the defs, concept of derivation, induction principle may be time-consuming initially

(my experience with induction on lex ordered tuples)

As each step $n \mapsto n + 1$ usually involves a small set of operations maybe develop new tactics to make implementing in Isabelle easier?

EXCEPTIONS: Not all paper proof steps $n \mapsto n + 1$ are easily implementable (eg: variable conventions)
ISAR - extension of Isabelle with structured proofs in a stylized language of mathematics

theorem "∃S. S ∉ range (f :: 'a ⇒ 'a set)"
proof
  let ?S = "{x. x ∉ f x}"
  show "?S ∉ range f"
  proof
    assume "?S ∈ range f"
    then obtain y where fy: "?S = f y" ..
    show False
    proof cases
      assume "y ∈ ?S"
      hence "y ∉ f y" by simp
      hence "y ∉ ?S" by(simp add:fy)
      thus False by contradiction
    next
      assume "y ∉ ?S"
      hence "y ∈ f y" by simp
      hence "y ∈ ?S" by(simp add:fy)
      thus False by contradiction
  qed
  qed
  qed
Why formalize a proof in Isabelle?

- We are unsure about the result because existing proofs are very complicated

- To refine a paper proof (eg: ensure definitions are sufficiently precise, tight)

I’m sure the above list is not definitive. But..
To check if a particular mathematical proof, journal paper is correct? Not sure

1. How can we be sure our Isabelle implementation accurately represents original proof?

2. Even if we do find ‘hole’ in proof with the help of Isabelle, is it obvious that a similar, lesser effort directed towards paper proof would not have revealed ‘hole’?

Verifying a compiler, memory handling makes sense as we are unsure of result and the proof is very complicated, less elegant

Verifying the prime number theorem, esp. beginning from basic number theory? Not sure