Interfacing Isabelle with other systems

Verification Technology: Systems and Applications
INRIA Nancy
October 2009

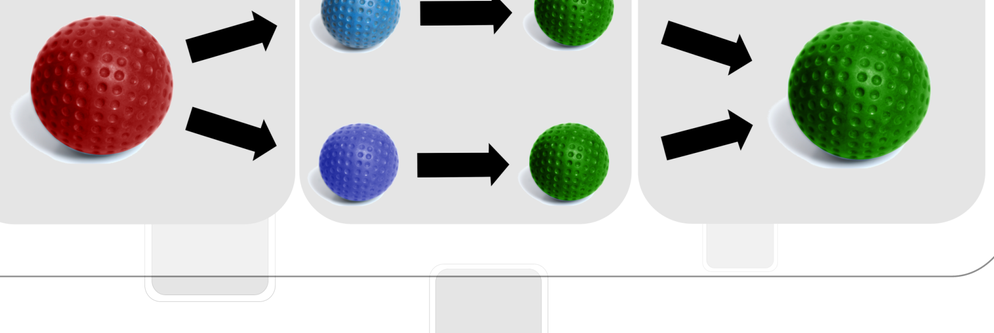
Goals

- Making it easier to write proofs by computer, through better automation.
- Achieving this by exploiting the combined strengths of specialised tools.

Strategy

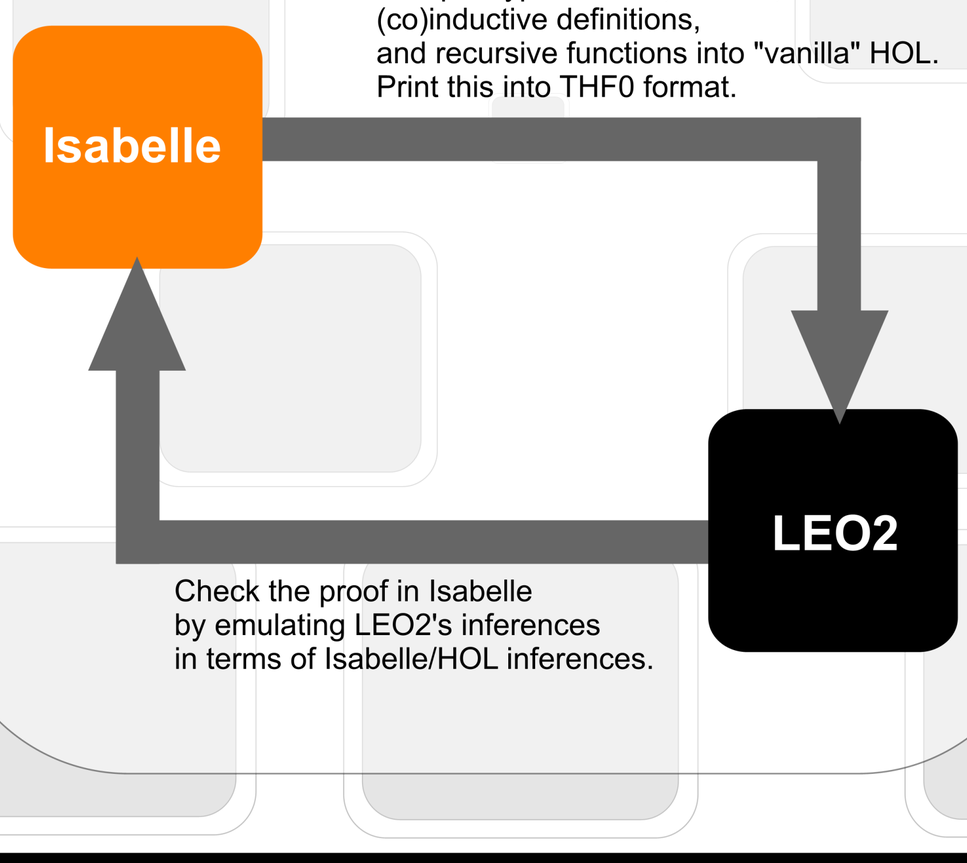
Start with a problem which has been encoded into an expressive logic, then:

1. Break problem down into simpler ones which are expressible in simpler logics.
2. Dispatch subproblems to automatic tools.
3. Compose the proofs back into a proof for the original problem, in the original logic.



Isabelle/HOL and LEO2

- Both implement classical HOL.
- Isabelle/HOL has a "higher-level" syntax which needs to be expanded into "vanilla" HOL to generate input for LEO2.
- LEO2 works by invoking other provers in turn. The proofs returned by the latter may need to be validated separately.



Properties

A successful integration would ideally have the following qualities.

Soundness

This is assured by obtaining a proof from the target system, and checking this inside the LCF kernel. This is usually done by emulating the inferences of the target system.

Feasibility

The whole process should not take long to return a result.
Soundness may compromise feasibility, by swamping the target system with lots of details.

Usability

Ideally, the user is presented with a push-button interface. This requires problem preparation and relevance filtering to be fully automated.
To avoid distracting the user, or freezing the system during proof-search, the target system should run as a background process. Different target systems can be run in parallel. Finally, the proof that is returned by the target system can be translated into a "source-level" proof script.
This avoids the user having the re-run proof-search again, and only run the proof-checker.

Prior art

Isabelle has been interfaced with a variety of systems which implement different logics. Some of this work is outlined below.

Previously, Isabelle was extended with proof-search functionality. For instance, the tactic "blast" is a tableau proof system.

SAT and SMT solvers

Higher-order features of formulas are abstracted to yield formulas which can be fed into SAT solvers (such as zChaff) and SMT solvers (such as Yices).

First-Order provers

Isabelle uses the first-order prover Metis to mediate its interface with other systems, including E, Vampire and SPASS.

Various ways of reducing higher-order formulas into first-order form have been compared. These vary in how much type-information is encoded into the target formulae.

Higher-Order provers

Isabelle/HOL theories can be imported into HOL Light. Due to the comparatively terse logic of HOL Light, OCAML code is generated during the translation to emulate the behaviour Isabelle/HOL declarations.

Other

Integrations with Computer Algebra Systems have also been studied.

Acknowledgements

I have benefitted from advice given to me by Larry Paulson, Tobias Nipkow, Tjark Weber, Christoph Benzmueller.
Thanks also to the members of the Automated Reasoning Group at the Computer Laboratory.

More on Isabelle at www.cl.cam.ac.uk/research/hvg/Isabelle

Computer Laboratory
University of Cambridge

Nik Sultana
nik.sultana@cl.cam.ac.uk
www.cl.cam.ac.uk/~ns441