Integrating Gandalf and HOL

Joe Hurd

University of Cambridge

TPHOLs

17 September 1999

1. Introduction
2. How it works
3. Results
4. Conclusion

Introduction

HOL

- Higher-order logic
- Emphasis on consistency
- Highly general meta-language

Gandalf

- First-order classical logic with equality
- Emphasis on speed
- Highly specific tool

Aim to get the best of both worlds.

Introduction

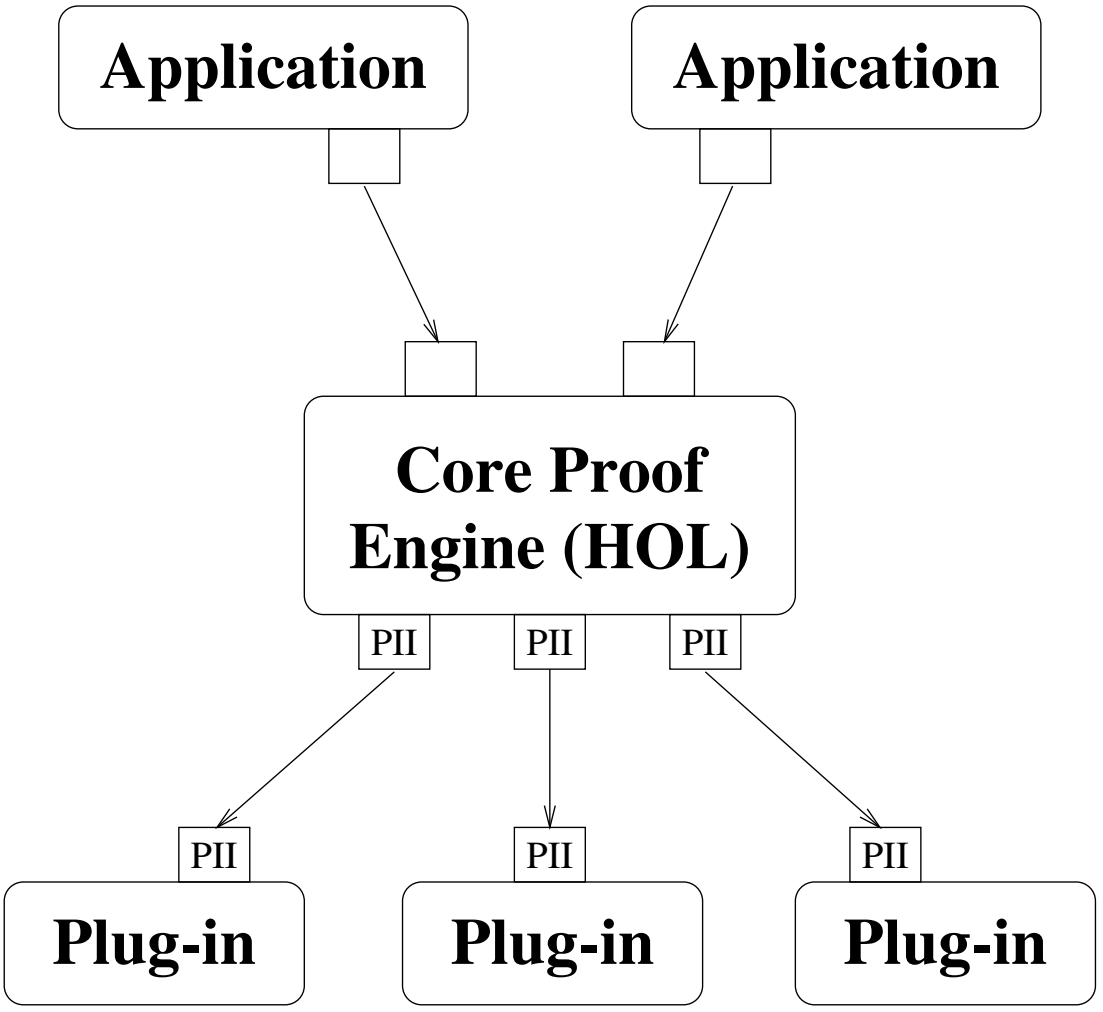
What's new here?

Two significant novelties in this work:

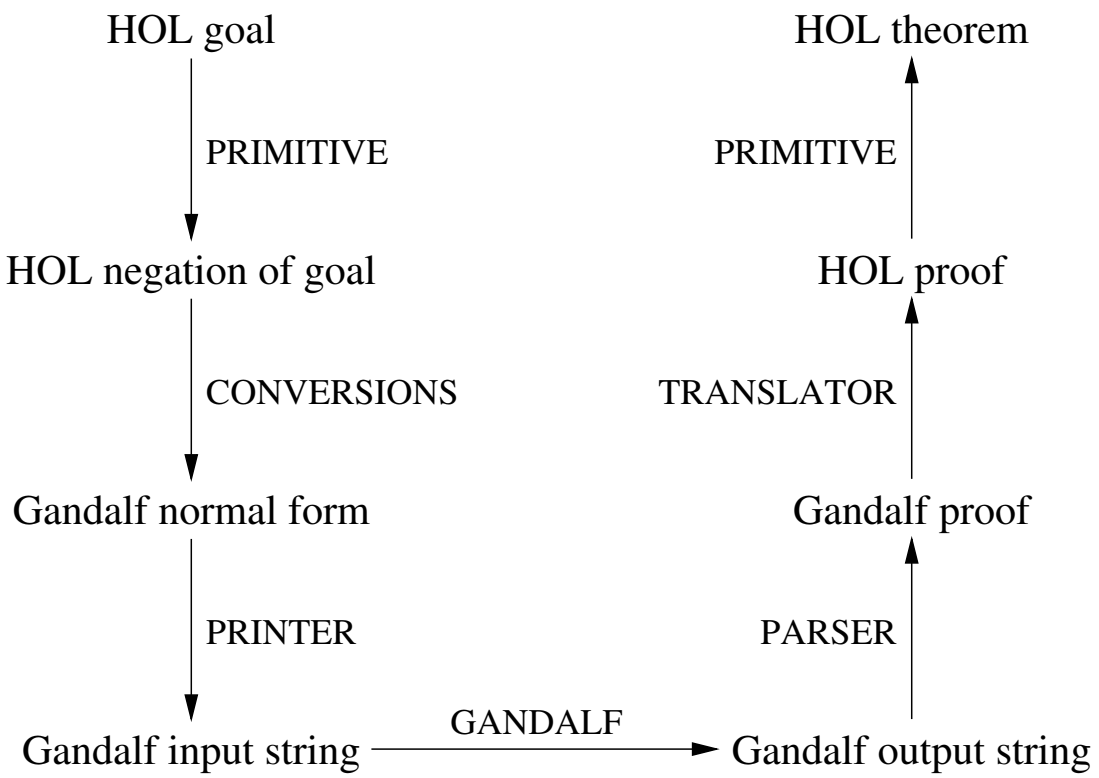
- The use of a completely separate 'off-the-shelf' theorem-prover, treating it as a black box.
- The systematic use of a generic plug-in interface.

Introduction

Prosper overview



How it works


HOL goal
PRIMITIVE
HOL negation of goal
CONVERSIONS
Gandalf normal form
PRINTER
Gandalf input string
GANDALF
Gandalf output string
PARSER
Gandalf proof
TRANSLATOR
HOL proof
PRIMITIVE
HOL theorem

How it works

An example goal

Let's use GANDALF_TAC to prove the goal

$$\forall ab.\ \exists x.\ Pa \vee Pb \Rightarrow Px$$

How it works

Initial primitive steps

In the first stage of processing we assume the negation of the goal:

$$\{\neg(\forall ab.\ \exists x.\ Pa \vee Pb \Rightarrow Px)\}$$
$$\vdash \neg(\forall ab.\ \exists x.\ Pa \vee Pb \Rightarrow Px)$$

How it works

Conversions

We now use standard conversions to change the conclusion to CNF:

$$\{\neg(\forall ab.\ \exists x.\ Pa \vee Pb \Rightarrow Px)\}$$
$$\vdash \exists ab.\ \forall x'.\ \neg(Px') \wedge (Pa \vee Pb)$$

How it works

Printing

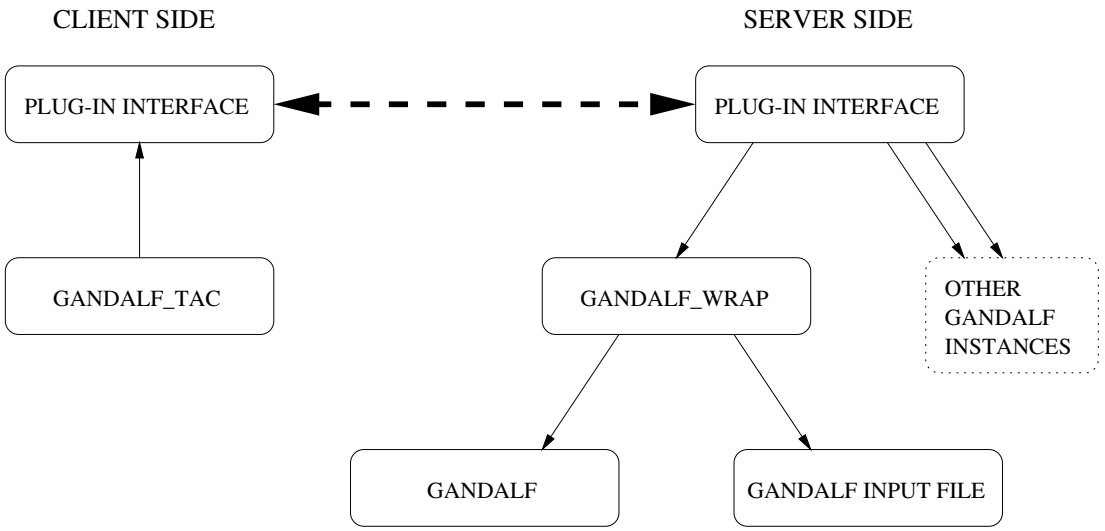
Next we print the formula in a format acceptable to Gandalf:

```
%-----------------------%
% hol -> gandalf formula %
%-----------------------%
set(auto).
assign(max_seconds,300).
assign(print_level,30).

list(sos).

-c10(x1).

c10(c5) |
c10(c6).
end_of_list.
```

How it works

Calling Gandalf (part 1)

We now use the Prosper plug-in interface to call Gandalf.



How it works

Calling Gandalf (part 2)

And here is the string we receive from Gandalf:

```
Gandalf v. c-1.0c starting to prove: gandalf.26884

strategies selected:
((binary 30 #t) (binary-unit 90 #f) (hyper 30 #f)
(binary-order 15 #f) (binary-nameorder 60 #f 1 3)
(binary-nameorder 75 #f))

********* EMPTY CLAUSE DERIVED *********


timer checkpoints: c(2,0,28,2,30,28)


1 [] c10(c5) | c10(c6).
2 [] -c10(x).
3 [binary,1,2,binary_s,2] contradiction.


1.794 1.911
```

How it works

Parsing

We parse the output string into special-purpose datatypes:

```
[(1, (Axiom(), []),
  [(true, Branch(Leaf "c10", Leaf "c5")),
   (true, Branch(Leaf "c10", Leaf "c6"))]),
 (2, (Axiom(), []),
  [(false, Branch(Leaf "c10", Leaf "x"))]),
 (3, (Binary((1, 1), (2, 1)), [Binary_s(2, 1)]),
  [])]
: (int
   * (Proofstep * Clausesimp list)
   * (bool * Tree) list) list
```

Note: Variable names can arbitrarily change!

How it works

Translating

We now translate the Gandalf proof into a HOL proof, using a Prolog-style backtracking algorithm to match each proof line.

Here is the theorem we end up with:

$$\{\neg(\forall ab.\ \exists x.\ Pa \vee Pb \Rightarrow Px)\} \vdash \bot$$

How it works

Final primitive steps

Now we need only use the contradiction axiom to establish our original goal:

$$\vdash \forall ab.\ \exists x.\ Pa \vee Pb \Rightarrow Px$$

Results

Performance (part 1)

Goal	MESON_TAC		GANDALF_TAC		
Non-equality					
T	0.027	31	0.034	32	(–)
$P \vee \neg P$	0.075	72	2.003	108	(0)
P50	0.159	243	1.638	441	(0)
Agatha	0.438	872	3.916	1891	(0)
PRV006_1	1.064	1501	41.044	8097	(109)
GRP031_2	1.267	713	8.083	3699	(251)
COL001_2	2.150	847	39.240	2620	(0)
LOS	5.705	917	5.110	2565	(0)
NUM021_1	7.535	1246	17.210	4352	(0)
CAT018_1	12.226	2630	61.585	13477	(0)
CAT005_1	63.849	2609	66.200	13371	(0)
Equality					
$x = x$	0.090	54	0.041	35	(–)
P48	0.394	636	2.707	495	(0)
PRV006_1	0.648	1053	13.558	4015	(0)
NUM001_1	0.768	876	7.032	3012	(0)
P52	1.157	1122	–	–	(–)
GRP031_2	1.377	757	7.946	3699	(251)
GRP037_3	3.402	1466	26.844	8242	(0)
CAT018_1	7.646	1809	28.560	8611	(0)
NUM021_1	7.737	1026	10.765	3423	(0)
CAT005_1	30.514	1784	29.490	8505	(0)
COL001_2	56.948	700	4.930	1273	(0)
Agatha	2003.00	1313	12.626	3409	(0)

Results

Performance (part 2)

	Conv.	Proof	Trans.	Total
GANDALF_TAC				
Non-equality	1.67	5.55	2.14	11.57
Equality	4.20	5.27	7.19	17.82
Combined	2.51	5.42	3.64	13.99
MESON_TAC				
Non-equality	0.26	0.36	0.06	0.90
Equality	0.43	1.27	0.09	2.61
Combined	0.33	0.66	0.07	1.50

Results

Gandalf the plug-in

GANDALF_TAC has contributed to the development of the plug-in concept.

- Evidence that plug-ins can really work.
- Plug-ins can happily coexist with the LCF logical core, and don't have to be oracles.
- Useful to test the plug-in interface.
- Perhaps could serve as a template to potential plug-in authors (at least for this class of black-box plug-ins).

Conclusions

- Need for good standards (e.g., formula formats, proof formats, APIs such as the plug-in interface).
- Good tool for hard problems, and still scope for optimization (e.g., altering Gandalf to output more explicit proofs, lifting search/conversion outside the logic).
- Step towards distributed theorem-proving (easy to farm out problem to multiple Gandalf servers, and accept first proof).
- Would be a lot more useful if proofs could be recorded, so that proof search is not necessary every time the theory is built.