Attempts to Bridge the Gap between Equality Reasoning and Logical Proving

Joe Hurd

University of Cambridge

1. What Gap?
2. Congruence Classes with Logic Variables
3. Proving without Normalisation

What Gap?

Many of the best modern automatic provers perform a rewriting stage, then a logical proving phase. They will miss many theorems that require interleaving.

But interleaving really would be helpful: logical proving would benefit from tightly-coupled equality reasoning to expand definitions, perform rewriting to normal form, and cope with if-then-else expressions.

Similarly, equality reasoning would benefit from tightly-coupled logical proving to better deal with conditional equalities.

Congruence Classes with Logic Variables

Goals:

1. Take something strong at equality reasoning, and make it a bit more user-friendly for adding logical proving steps.
2. An efficient way of storing terms, whatever the application.

We just add terms with logic variables, and congruence closure treats them as constants.

Matching Algorithm

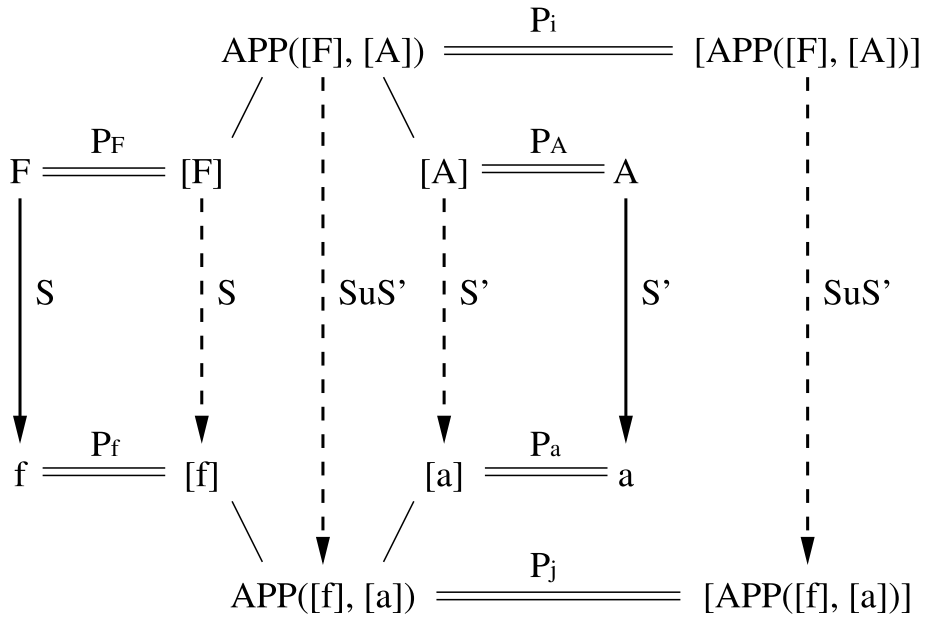
We can perform matching between classes 'modulo' the equalities implicit in the congruence classes.

Build up matches inductively:

During iniatialisation, add in logic variable matches and 'reflexive' matches.

For step case, if we have $app(C_i, C_j)$ in a class C, can use current matches to C_i and C_j to add more matches to C.

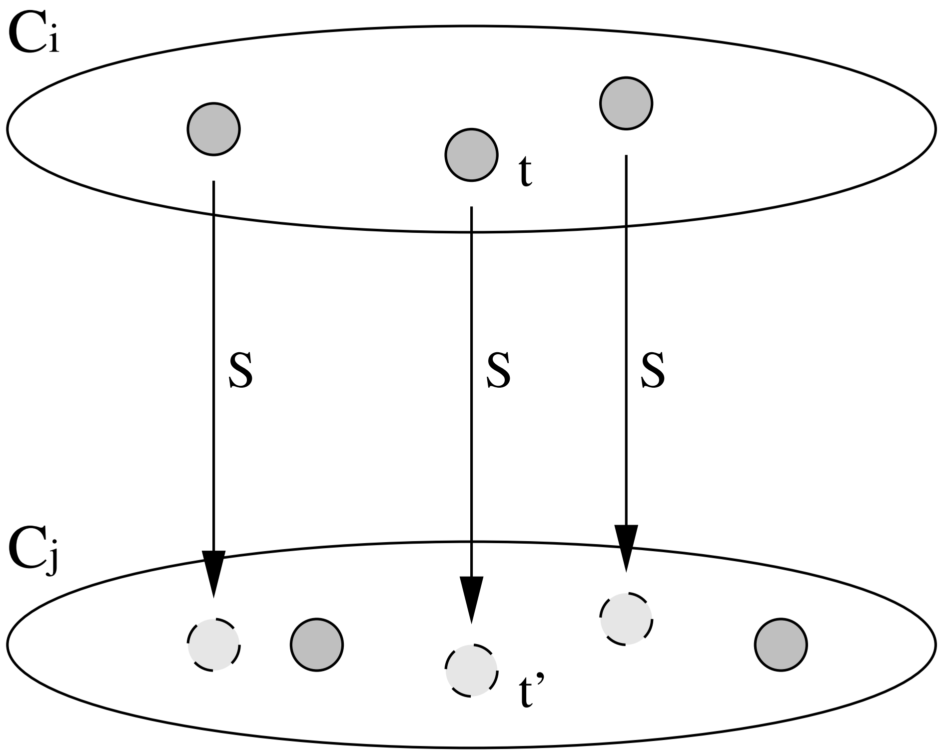
Matching Algorithm (contd)



$X \overset{P}{=\!=} [X]$	X is an element in the class with representative [X] (with proof P)
$X \xrightarrow{S} Y$	X matches to Y using substitution S
$X \overset{S}{\dashrightarrow} Y$	a term equal to X matches to a term equal to Y (using substitution S)

Percolation Algorithm

This makes use of the Matching Algorithm to perform undirected rewriting.



Test Examples

Ex	Lv	Theorem
1	1	$(\forall x.\ f(f(x)) = g(x))$
		$\Rightarrow (f(g(a)) = g(f(a)))$
2	2	$(\forall x\, y\, z.\ ((x \circ y) \circ z = x \circ (y \circ z))$
		$\wedge\, (e \circ x = x) \wedge (i(x) \circ x = e))$
		$\Rightarrow (x \circ i(x) = e)$
3	1	$a * b * c = c * b * a$
4	2	$a * b * c * d = d * c * b * a$
5	2	$a * b * c * d * e$
		$= e * d * c * b * a$
6	3	$(a + 1) * (a + 1)$
		$= a * a + a + a + 1$

Proving without Normalisation

Goals:

1. Combining proof tools like first-order provers, rewriters and decision procedures.
2. Would like human proof steps to roughly correspond 1-1 to automatic proof steps, so that there is a better correlation between obviousness in human terms and practical to prove in automatic terms.
3. An automatic tool that can perform a partial proof, and deliver understandable subgoals to the user.

Concepts

We perform ‘usual’ proof steps, creating logic variables for $\exists$ on a goal (or $\forall$ on a fact). Also when we remove a leading $\forall$ from a goal (or $\exists$ from a fact) we make it a function of all the logic variables in the term to preserve soundness.

When we change a goal we create a new context, and try to solve the new goal inside it (the contexts form a large tree). This device allows us to replace the goal $A \Rightarrow B$ with the fact A and the new goal B, without the fact being used inappropriately.

Every fact also carries around its proof, and if the search is successful the original goal will turn into a fact with proof. It is then possible to then translate this proof to a regular HOL proof.

An Example

Start with a root context, containing:

goal $(P1 \Rightarrow Q2) \Rightarrow (\exists x.\ Px \Rightarrow Qx)$

A child context is created, in which we have:

fact $P1 \Rightarrow Q2$

goal $\exists x.\ Px \Rightarrow Qx$

Now we create a child context of this one:

goal $PX \Rightarrow QX$

Another child context:

fact PX

goal QX

Another child context (by back-chaining):

goal $P1$

And this is solved by the fact PX in the parent context.

Translation

If there are any uninstantiated logic variables in the proof, the original goal must be true in all instantiations, so we can instantiate them to arbitrary elements of the type before we begin.

Since the proof steps are designed to be analogous to HOL steps, once the logic variables have disappeared the proof translation is straightforward.

There are 2 difficult cases.

Difficulty 1

Joining Together Two Halves of an Implication (as occurs in the example):

We had an original goal $A \Rightarrow B$, we created a context in which we put the new goal B and the fact A, and we have a proof of B within this context. How can we extract a proof of $A \Rightarrow B$?

The problem is that there might be a logic variable X in both A and B that has been instantiated in order to prove B and also differently instantiated (perhaps multiple times) in A used in the proof of B.

In the example, X is instantiated to 2 in B and 1 in A.

Not valid to claim either $P1 \Rightarrow Q1$ or $P2 \Rightarrow Q2$ (no proof of these).

Solution 1

If b is the instantiation of X in B, and $a_1, a_2, \ldots, a_n$ are the instantiations of X in A used in the proof of B, then we CAN claim $A \Rightarrow B$ with X set to:

if $A[a_1/X]$ then
    if $A[a_2/X]$ then
        $\ldots$
            if $A[a_n/X]$ then
                b
            else a_n
        $\ldots$
    else a_2
else a_1

Since this makes A false if any of $A[a_i/X]$ are false, and B true otherwise.

Solution 1 contd

Now we can show the proof translation of the example.

In the root context:
goal $(P1 \Rightarrow Q2) \Rightarrow (\exists x.\ Px \Rightarrow Qx)$
proof $\vdash (P1 \Rightarrow Q2) \Rightarrow (\exists x.\ Px \Rightarrow Qx)$

The first child context:
fact $P1 \Rightarrow Q2$
goal $\exists x.\ Px \Rightarrow Qx$
proof $[P1 \Rightarrow Q2] \vdash \exists x.\ Px \Rightarrow Qx$

A child context of this one:
goal $PX \Rightarrow QX$
proof $[P1 \Rightarrow Q2] \vdash P(\text{if } P1 \text{ then } 2 \text{ else } 1) \Rightarrow Q(\text{if } P1 \text{ then } 2 \text{ else } 1)$

Another child context:
fact PX
goal QX
proof $[P1,\ P1 \Rightarrow Q2] \vdash Q2$

Another child context (by back-chaining):
goal $P1$
proof $[P1] \vdash P1$

Difficulty 2

$\forall$ Variables in a Goal can Escape their Scope!

Consider the goal $\exists x.\ \forall y.\ Px \Rightarrow Py$.

This is how it is proved:

root context:

goal $\exists x.\ \forall y.\ Px \Rightarrow Py$

child context:

goal $\forall y.\ PX \Rightarrow Py$

child context:

goal $PX \Rightarrow P(yX)$

child context:

fact PX

goal $P(yZ)$

Difficulty 2 contd

This is how the proof is translated:

root context:

goal $\exists x.\ \forall y.\ Px \Rightarrow Py$

proof AARGH!

child context:

goal $\forall y.\ PX \Rightarrow Py$

proof $\vdash \forall y. P(\text{if } Py \text{ then } Z \text{ else } y) \Rightarrow Py$

child context:

goal $PX \Rightarrow P(yX)$

proof $\vdash P(\text{if } P(yZ) \text{ then } Z \text{ else } (yZ)) \Rightarrow P(y(\text{if } P(yZ) \text{ then } Z \text{ else } (yZ)))$

child context:

fact PX

goal $P(yZ)$

proof $[P(yZ)] \vdash P(yZ)$

We want x to map to (if Py then Z else y), but it contains a y.

Solution 2

Instead of just replacing the bound variable y in the goal with a free variable of the same name, replace the bound variable with a new variable with the definition:

$$h = \lambda x.\ (\varepsilon y.\ \neg(Px \Rightarrow Py))$$

Now h will escape the scope of y, but it doesn't matter, it can exist in any scope. The only thing we must remember to do is erase all the definitions from the assumptions at the very end of the proof translation (taking care to erase them in the right order, since they might depend on each other!).

Solution 2 contd

Now the proof becomes:

root context:

goal $\exists x.\ \forall y.\ Px \Rightarrow Py$

proof $[h = \lambda x.\ (\varepsilon y.\ \neg(Px \Rightarrow Py))] \vdash \exists x.\ \forall y.\ Px \Rightarrow Py$

child context:

goal $\forall y.\ PX \Rightarrow Py$

proof $[h = \lambda x.\ (\varepsilon y.\ \neg(Px \Rightarrow Py))] \vdash$
$\forall y.\ P(\text{if } P(hZ) \text{ then } Z \text{ else } (hZ)) \Rightarrow Py$

child context:

goal $PX \Rightarrow P(hX)$

proof $\vdash P(\text{if } P(hZ) \text{ then } Z \text{ else } (hZ)) \Rightarrow$
$P(h(\text{if } P(hZ) \text{ then } Z \text{ else } (hZ)))$

child context:

fact PX

goal $P(hZ)$

proof $[P(hZ)] \vdash P(hZ)$

Results

- Works on small examples, but get bogged down very quickly.
- Next will most likely be adding congruence closure at a low level, and then hopefully the fast equality processing will make it a useful tool.
- Perhaps could rewrite the goalstack code to include logic variables, and then it could work interactively.
- Suggestions welcome!