Congruence Classes with Logic Variables

Joe Hurd

University of Cambridge

1. Motivation
2. Review of Congruence Classes
3. Matching Algorithm
4. Percolation Algorithm

Motivation

We have observed that the performance of automatic first-order provers is much worse on problems featuring equality, for example:

$$\begin{array}{ll} \textbf{Thm:} & \{\forall\, x\, y\, z.\ (x * y) * z = x * (y * z)\} \\ & \wedge \quad \{\forall\, x.\ e * x = x\} \\ & \wedge \quad \{\forall\, x.\ i(x) * x = e\} \\ & \Rightarrow \quad x * i(x) = e \end{array}$$

$$\begin{array}{llll} \textbf{Proof:} & e & = & i(x * i(x)) * (x * i(x)) \\ & & = & i(x * i(x)) * (x * (e * i(x))) \\ & & = & i(x * i(x)) * (x * ((i(x) * x) * i(x))) \\ & & = & i(x * i(x)) * ((x * (i(x) * x)) * i(x)) \\ & & = & i(x * i(x)) * (((x * i(x)) * x) * i(x)) \\ & & = & i(x * i(x)) * ((x * i(x)) * (x * i(x))) \\ & & = & (i(x * i(x)) * (x * i(x))) * (x * i(x)) \\ & & = & e * (x * i(x)) \\ & & = & x * i(x) \end{array}$$

Review of Congruence Classes

Provers struggle because of the vast number of ways of expressing a given term. Congruence classes are a way of storing terms that maximizes sharing of equal subterms, and this helps because:

- it uses less memory to store the terms;
- it performs the congruence closure decision procedure, which can cut down the search;
- it allows the prover to deal with values, instead of representations.

If our terms include logic variables, then congruence closure will treat them as constants.

Can we provide anything more than this to the client prover?

Matching Algorithm

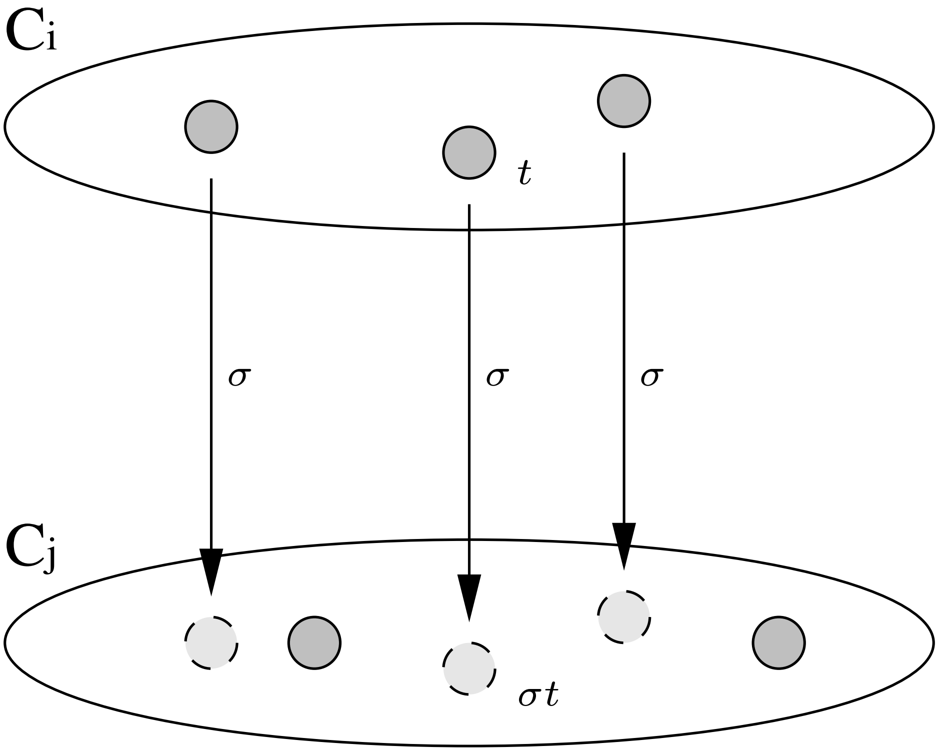
We can perform matching between classes 'modulo' the equalities implicit in the congruence classes.

Build up matches inductively:

- During initialization, 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.

Percolation Algorithm

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



For every match σ between classes C_i and C_j, if $t \in C_i$ then we add σt to C_j.