Review

- The state of NO prover can be characterized
  \[ F_1 \land \ldots \land F_n \implies \bot \]
  (trying to derive a contradiction from a set of literals)

- A satisfiability procedure can also declare subgoals
  \[ G \text{ is a subgoal if} \]
  \[ F_1 \land \ldots \land F_n \land G \implies \bot \quad \text{(G helps prove a contradiction)} \]

  This makes the prover try to prove
  \[ F_1 \land \ldots \land \overline{F_n} \implies G \]

- This mechanism generalizes the equality showing
  \[ F_1 \land \ldots \land F_n \implies E \text{ or } \overline{F_1 \land \ldots \land F_n \land \overline{E}} \implies \bot \]
  \[ \overline{E} \text{ is a subgoal} \]
  \[ \text{Next state is } F_1 \land \ldots \land F_n \land \overline{E} \implies \bot \]

- It also generalizes disjunction of equality showing
  \[ F_1 \land \ldots \land F_n \implies E_1 \lor E_2 \]
  \[ \text{two subgoal states} \]
  \[ F_1 \land \ldots \land F_n \land E_1 \implies \bot \]
  \[ F_1 \land \ldots \land F_n \land E_2 \implies \bot \]
**Question**: Is it possible to generate a tactic automatically from the axiomatization of a theory?

**Answer**: Yes, in some cases

Consider the axiomatization in the form of clauses:

\[
H_1 \ldots H_m \quad \vdash c
\]

If this rule is appropriate for backwards chaining (\(\text{Var}(H_i) \subseteq \text{Var}(C)\)) then after instantiating the conclusion we have instances of the hypotheses.

Thus, if the state is \(F_1 \land \ldots \land F_n \equiv 1\) and there is an instantiation \(\Theta\) such that

\[\Theta(c) = F_i\]

then we can announce the subgoal \(\Theta(H_i) \land \ldots \land \Theta(H_m)\).

This leads to \(m\) subgoals:

\[F_1 \land \ldots \land F_n \Rightarrow \Theta(H_i) \land i=1 \ldots m\]

This is in fact how Prolog interprets a set of clauses

- it is somewhat simpler here because of the condition on variables
But this form of matching is not enough.
Consider:

\[
g(y) \quad \not\forall (s(x_1y), x)
\]

and trying to prove \( p(a) = s(a, b) \land g(b) \Rightarrow \forall (p(a), a) \)

There is no \( \Theta \) such that \( \Theta(\neg \forall (s(x_1y), x)) = \top \)

- if we do regular matching.

But if we do E-DAG matching then

\[
[a/x, b/y] (\neg \forall (s(x_1y), x)) = \neg \forall (p(a), a)
\]

(because \( p(a) = s(a, b) \))

Matching in the E-DAG is a slight modification on the regular matching.

Instead of checking that \( \forall(e) \equiv e' \)
check that \( \forall(e) \) is congruent to \( e' \)

Take into consideration all members in an equivalence class.

The problem of determining whether there is a match is NP-complete.

But experience suggests it is worth the cost.
Still, this form of matching is too weak:
- expects to find an exact instance of the
  negation of the conclusion among the literals
- still has the restriction that the
  conclusion has all the variables

\[ \text{Extension 1} \]
\[ \theta(H_i) = \neg \theta_j \text{ then } \theta(H_1) \land \ldots \land \theta(H_{i-1}) \land \theta(H_i) \land \theta(C) \text{ is a subgoal} \]

- this might be a good idea when \( H_i \) contains
  all the variables
- the point here is to view the rule in a
  symmetric way as \( \neg H_1 \lor \ldots \lor \neg H_n \lor C \)
  and to separate among \( H_i, C \) those that
  contain all variables

But what about a rule of the form

\[
\frac{\ell(x, y) \quad \ell(y, z)}{\ell(x, z)}
\]

- no hypothesis or conclusion contains all
  variables
Extension 2

Watch a few $H_i$ and/or TC among the $F_i$'s such that all variables are instantiated.

E.g. $H_1 \ldots H_k \ H_{k+1} \ldots H_n \ \frac{\text{c}}{\text{c}}$

and $\text{Var}(c) \cup \text{Var}(H_i) \supseteq \text{Var}(H_j)$

If there is $\theta$ such that

$\{\theta(c), \theta(H_1), \ldots, \theta(H_k)\} \subseteq F$

then add the subgoal

$\theta(H_{k+1}) \land \ldots \land \theta(H_n)$

and $H_1 \ldots H_k$ are trigger terms.

E.g. $l(x,y) \quad l(y,z) \quad l(x,z)$

triggers if $l(x,y) \land l(y,z) \rightarrow \neg l(x,z)$

or if $l(x,y) \land \neg l(x,z) \rightarrow l(y,z)$

which is how it should be
E.g.
\[
\frac{g(x)}{l(p(x), x)} \quad \frac{g(y)}{l(x, s(x, y))} \quad \frac{l(x, y) \ l(y, z)}{l(x, z)}
\]

try to prove
\[
g(a) \land g(b) \land \neg l(p(a), s(a, b))
\]

(2) and (3) cannot fire
(1) can fire and produces
\[
\frac{l(p(b), b)}{l(p(a), a)} \quad \text{does not enable anything}
\]

Now (2) still cannot fire
But (3) can fire
\[
l(p(a), a) \land \neg l(p(a), s(a, b))
\]
so we add literal \(\neg l(a, s(a, b))\)

(Note that 3 will never fire again with instantiation \(x \mapsto p(a), y \mapsto a, z \mapsto s(a, b)\))

Now 2 can fire to add literal \(\neg g(b)\)

Which leads to a contradiction.
• But this is still not enough because we require a hypothesis, or conclusion to match completely an existing literal.

• We need instead a notion of "close" match.

• Several theorem provers say that the instantiation $\theta$ is a close match for clause $C_1 \lor \ldots \lor C_n$ if for all variables $x$ there is a subterm of $C_i$ that properly contains $x$ such that $\theta(t)$ is represented in the E-DAG.

  • we don't want to watch the entire $C_i$ but just a subterm
  • we take into consideration the equalities in the E-DAG

Example

$$
\frac{g(x)}{l(p(x), x)} \quad 1 \quad \frac{g(y)}{l(x, s(f(x), y))} \quad 2 \quad \frac{l(x, y)}{l(y, z)} \quad 3
$$

• Two close matches for Ax 1 $x \rightarrow a$ and $x \rightarrow b$

• No close match for Ax 2

Trying to falsify

\[ c = f(a) \land g(a) \land g(b) \land \neg l(p(a), s(c, b)) \]

• No close match for Ax 3
One close match for \( \text{Ax. 2} \)

\[
\begin{align*}
  x &\rightarrow a \\
  y &\rightarrow b
\end{align*}
\]

\((f(x))\) is represented

\((g(x))\) is represented.

also \(s(f(x), y)\) is represented

Say we do \( \text{Ax 1} \) first

. we add literal \( l(p(a), a) \)

Now \( \text{Ax. 3} \) has a close instance

\[
\begin{align*}
  x &\rightarrow p(a) \\
  y &\rightarrow a \\
  z &\rightarrow s(c, b)
\end{align*}
\]

. add literal \( \neg l(a, s(c, b)) \)

Now \( \text{Ax. 2} \) has a close match. (some as before)

. add literal \( \neg g(y) \)

\( \Rightarrow \) contradiction

But

. easy to go in a infinite loop

\( \Rightarrow \) do breadth first

. a multitude of other heuristics can be used
Once we have a good solution to matching we can think of extending the language of predicates that we handle.

- We can add
  - implication
  - universal quantification to the right-hand side of $\Rightarrow$
  - existential quantification to the right-hand side.