Where we left off

Main search strategy
- DPLL backtracking search
- Enumerating over Herbrand’s universe
- Backtracking search with SAT solver (Verifun)

Cross-cutting aspects
- Quantifiers
- Induction
- Equality
- Decision procedures

Equality (Part I): Congruence closure and E-graph

Part II of equality (rewrite rules) will be later in the quarter

Equality with uninterpreted func symbols

- Want to establish, for example, that \( f(f(a,b),b) = a \) follows from \( f(a,b) = a \)
- Or that \( f(a) = a \) follows from \( f(f(f(a))) = a \) and \( f(f(f(f(f(a))))) = a \)
- These kinds of inferences are often required to perform program verification

Axioms of EUF

- Intuition behind decision procedure for EUF: repeatedly apply these axioms to infer new equalities

Representing terms with graphs

Applying TRANS

- If node a is equal to node b and node b is equal to node c, then node a is equal to node c
Applying TRANS

• If node a is equal to node b and node b is equal to node c, then node a is equal to node c

Applying EQ-PROP

• If two nodes have the same label, and all their children are pairwise equal, then the two nodes are also equal

Applying EQ-PROP

Example

• Assume $f(a,b) = a$, show $f(f(a,b),b) = a$

Example

Another example

• Assume $f(f(f(a))) = a$ and $f(f(f(f(f(a))))) = a$
• Show $f(a) = a$
Another example

- Assume \( f(f(f(a))) = a \) and \( f(f(f(f(f(a)))))) = a \)
- Show \( f(a) = a \)

Let’s formalize this

- Suppose we have a labeled graph \( G = (V,E) \), where:
  - \( \lambda(v) \) denotes the label of vertex \( v \)
  - \( \delta(v) \) denotes the outdegree (number of outgoing edges) of vertex \( v \).
  - \( v[i] \) denotes the \( i \)th successor of vertex \( v \)

Congruence

- Let \( R \) be a relation on \( V \). Two vertices \( u \) and \( v \) are congruent under \( R \) if:
  - \( \lambda(u) = \lambda(v) \)
  - \( \delta(u) = \delta(v) \)
  - for all \( i \) such that \( 1 \leq i \leq \delta(u) \), \( (u[i], v[i]) \in R \)
- Intuition:
  - \( R \) represents known equalities
  - Read congruent as “identical”
  - This is essentially an instance of the EQ-PROP rule

Congruence closure

- A relation \( R \) on \( V \) is closed under congruences if, for all vertices \( u \) and \( v \) such that \( u \) and \( v \) are congruent under \( R \), \( (u,v) \in R \)
- For any given \( R \), there exists a unique minimal extension \( R' \) of \( R \) such that \( R' \) is an equivalence relation and \( R' \) is closed under congruences; \( R' \) is the congruence closure of \( R \)

Example

- Assume \( f(a,b) = a \), show \( f(f(a,b),b) = a \)

Example

- Assume \( f(a,b) = a \), show \( f(f(a,b),b) = a \)

\[
R = \{ (f(a,b),a) \}
\]

Congruence closure \( R' \) is:

\[
R' = \{ f(a,b) = a, f(f(a,b),b) = a, a = a, b = b, f(a,b) = f(a,b), f(f(a,b),b) = f(f(a,b),b) \}
\]
Computing the congruence closure

- A simple algorithm for computing the congruence closure:
  1. For any \((u,v) \in R\), merge \(u\) and \(v\) into one node
  2. While there are vertices \(u\) and \(v\) that are congruent under \(R\), but for which \((u,v) \notin R\), merge \(u\) and \(v\) into one node
- Upon termination, each node in \(R\) represents an equivalence class in the congruence closure \(R'\)
- How do we find the vertices \(u\) and \(v\) in step 2?

Representing the equiv. relation

- Represent the equivalence relation by its corresponding partition, that is, by the set of its equivalence classes
- \(\text{UNION}(u,v)\) combines the equivalence classes of vertices \(u\) and \(v\)
- \(\text{FIND}(u)\) returns the unique name of the equivalence class of vertex \(u\)

A refined version of the algorithm

- Given \(R\) that is closed under congruences, \(\text{MERGE}(u,v)\) constructs the congruence closure of \(R \cup \{(u,v)\}\)
- \(\text{MERGE}(u,v)\)
  1. If \(\text{FIND}(u) = \text{FIND}(v)\) then return
  2. Let \(P_u\) be the set of predecessors of all vertices equivalent to \(u\), and \(P_v\) the set of predecessors of all vertices equivalent to \(v\)
  3. Call \(\text{UNION}(u,v)\)
  4. For each pair \(x \in P_u, y \in P_v\), if \(\text{FIND}(x) \neq \text{FIND}(y)\) and \(\text{CONGRUENT}(x,y) = \text{TRUE}\), then \(\text{MERGE}(x,y)\)
- \(\text{CONGRUENT}(u,v)\)
  1. If \(\delta(u) \neq \delta(v)\) then return \(\text{FALSE}\)
  2. For \(1 \leq i \leq \delta(u)\), if \(\text{FIND}(u[i]) \neq \text{FIND}(v[i])\), then return \(\text{FALSE}\)
  3. Return \(\text{TRUE}\)

Decision procedure for EUF

- SAT-EUF(F), where \(F\) is a conjunction of equalities and inequalities:
  - Build a graph \(G\) for all terms in the equalities and inequalities, where \(\tau(t)\) is the node for term \(t\)
  - For each equality \(t_1 = t_2\), call \(\text{Merge}(\tau(t_1), \tau(t_2))\)
  - For each inequality \(t_1 \neq t_2\), if \(\text{FIND}(\tau(t_1)) = \text{FIND}(\tau(t_2))\), then return UNSAT
  - Return SAT

E-graph

- The graph from Nelson-Oppen 80 was later called the E-graph (equality graph)
- Nodes in the E-graph are the equivalence classes
- We can represent an inequality \(a \neq b\) in the E-graph with a special inequality edge between \(\text{FIND}(\tau(a))\) and \(\text{FIND}(\tau(b))\)
- If two equivalence classes connected by an inequality edge are merged, then there is an inconsistency

E-graph

- The E-graph can be constructed incrementally
  - Adding an equality causes cascading merges
  - Adding an inequality causes a new inequality edge to appear
- We can incorporate the E-graph into our backtracking search algorithm
  - Use E-graph as the context
  - Adding an assumption to the context now means inserting the (in)equality into the E-graph
Example

• Show the following is unsatisfiable:

\[ a = b \land (\neg (f(a) = f(b)) \lor b = c) \land \neg (f(a) = f(c)) \]

\[ \vdash \vdash \vdash \vdash \]

Example

• Show the following is unsatisfiable:

\[ a = b \land (\neg (f(a) = f(b)) \lor b = c) \land \neg (f(a) = f(c)) \]

\[ \vdash \vdash \vdash \vdash \]

We have a theorem prover for EUF!

• The theorem prover we have so far does a backtracking search in semantic domain, with an E-graph for keeping track of the environment
• Let’s call this Simplify--
  – This name was carefully chosen: our theorem prover has the same core as Simplify
• What’s in Simplify that’s missing from Simplify--?
  – Quantifiers. We’ll see this next.
  – Interpreted function symbols. For example, how can we prove \( x = y \Rightarrow x \geq y \). We’ll see this next time.

Recap

Next: matching heuristic for universal quantifiers

- Main search strategy
  - Proof-system search (\( \vdash \vdash \vdash \vdash \) )
  - Interpretation search (\( \vdash \vdash \vdash \vdash \) )

Cross-cutting aspects

- Equality
- Induction
- Quantifiers
- Decision procedures

• DPDL backtracking search
• Enumerating over Herbrand’s universe
• Backtracking search with SAT solver

• Prenex normal form
• Replacing \( \exists \) with function symbols

Instantiating universal quantifiers

• Suppose that \( \forall x_1, \ldots, x_n, \overline{P} \) is known to hold
  – For example, because it is an axiom
  – Or because it is added to the environment during a backtracking search
• We want to find substitutions \( \theta \) giving values to \( x_1, \ldots, x_n \) such that \( \theta(P) \) will be useful in the proof

Matching heuristic
General idea in matching

- Pick a term $t$ from P called a trigger (also called a pattern)
- Instantiate P with a substitution $\theta$ if $\theta(t)$ is a term that the prover is likely to require information about
- Intuition of why this works:
  - Since P contains the term $t$, $\theta(P)$ will contain the term $\theta(t)$, and so it provides information about $\theta(t)$
  - This is likely to be useful, since the prover wants information about $\theta(t)$

Example

- Assume $\forall x,y \ . \ \text{car}(\text{cons}(x,y)) = x$
- Let's use the trigger $\text{car}(\text{cons}(x,y))$
- We are therefore looking for values of $x$ and $y$ such that $\theta(\text{car}(\text{cons}(x,y)))$ appears in the E-graph

Matching with backtracking

- While performing the backtracking search, periodically perform matching to add new assumptions into the context
- Let's try this to prove the following:
  - Assume the BG axiom $\forall x,y \ . \ \text{car}(\text{cons}(x,y)) = x$
  - Show $\text{cons}(a,b) = \text{cons}(c,d) \Rightarrow a = c$
Matching with backtracking

\[ \forall x. (x = f(x)) \Rightarrow x = f(x) \]

Choice of trigger is important

- Goal: find a term or set of terms that will contain all the variables in the universal quantifier
- But... Many possible choices
- Smaller triggers are more general, but may lead to instantiating more times than needed
- Larger triggers are less general, but may lead to missed instantiations

Matching in the E-graph

- Matching in the E-graph is done up to equivalence
- Consider an E-graph that represents the equality \( f(a) = a \).
- Despite having only two nodes, this E-graph represents not only \( a \) and \( f(a) \), but also \( f(f(a)) \) and actually \( f^n(a) \) for any \( n \).
- Because the E-graph represents more terms than it has nodes, matching in the E-graph is more powerful than simple conventional pattern-matching, since the matcher is able to exploit the equality information in the E-graph.

Example of exploiting E-graph info

- Assume \( \forall x. f(x) = x \quad \forall x. g(g(x)) = x \)
- Show \( g(f(g(a))) = a \)

\[ f(a) \neq f(b) \]

\[ a \neq b \]