Congruence Closure

Slides by Jeremy Condit and Matthew Harren

Review

Theory of Equality.

- The theory of equality with uninterpreted functions
- Symbols: $=, \neq, f, g, \ldots$
- Axiomatically defined:

$$\text{reflexivity} \quad \frac{}{E = E} \qquad \frac{E_1 = E_2 \quad E_2 = E_3}{E_1 = E_3} \quad \text{transitivity}$$

$$\text{symmetry} \quad \frac{E_2 = E_1}{E_1 = E_2} \qquad \frac{E_1 = E_2}{f(E_1) = f(E_2)} \quad \text{congruence}$$

- Example of a satisfiability problem:

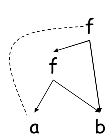
$$g(g(g(x)) = x \wedge g(g(g(g(g(x))))) = x \wedge g(x) \neq x$$

A Satisfiability Procedure for Equality

- Definitions:
  - Let R be a relation on terms
  - The equivalence closure of R is the smallest relation that is closed under reflexivity, symmetry and transitivity
    - An equivalence relation
- Equivalence classes
  - Given a term t we say that t^* is its representative
  - Two terms t_1 and t_2 are equivalent iff $t_1^* = t_2^*$
  - Computable in near-linear time (union-find)
- The congruence closure of a relation is the smallest relation that is closed under equivalence and congruence

A Representation for Symbolic Terms

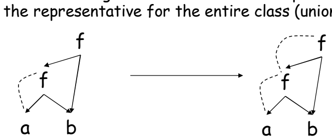
- We represent terms as DAGs
  - Share common subexpressions
  - E.g. $f(f(a, b), b)$:



- Equalities are represented as dotted edges
  - E.g. $f(f(a, b), b) = a$
  - Called an E-DAG
- We consider the transitive closure of dotted edges

Computing Congruence Closure

- We pick arbitrary representatives for all equivalence classes (nodes connected by dotted edges)
- For all nodes $t = f(t_1, \ldots, t_n)$ and $s = f(s_1, \ldots, s_n)$
  - If $t_i^* = s_i^*$ for all $i = 1..n$ (find)
  - We add an edge between t^* and s^* and pick one of them as the representative for the entire class (union)



Computing Congruence Closure (Cont.)

- Congruence closure is an inference procedure for the theory of equality
  - Always terminates because it does not add nodes
- The hard part is to detect the congruent pairs or terms
  - There are tricks to do this in O(n log n)
- We say that $f(t_1, \ldots, t_n)$ is represented in the DAG if there is a node $f(s_1, \ldots, s_n)$ such that $s_i^* = t_i^*$

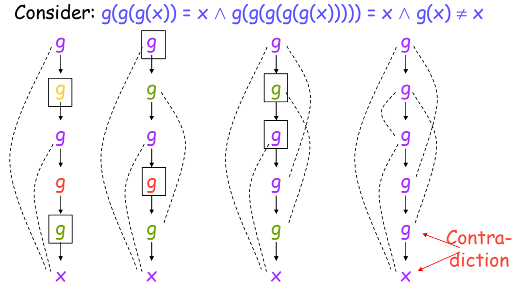
Satisfiability Procedure for Equality

1. Given $F = \bigwedge_i t_i = t_i' \ \wedge \ \bigwedge_j u_j \neq u_j'$
2. Represent all terms in the same E-DAG
3. Add dotted edges for $t_I = t_I'$
4. Construct the congruence closure of those edges
5. Check that $\forall j.\ u_j^* \neq u_j'^*$

Theorem: F is satisfiable iff $\forall j.\ u_j^* \neq u_j'^*$

Example with Congruence Closure

- Consider: $g(g(g(x))) = x \wedge g(g(g(g(g(x))))) = x \wedge g(x) \neq x$



Congruence Closure. Discussion.

- The example from before has little to do with program verification
- But equality is still very useful
- The congruence closure algorithm is the basis for many unification-based satisfiability procedures
  - We add the additional axiom:

$$\frac{f(E_1) = f(E_2)}{E_1 = E_2}$$

  - Or equivalently:

$$\frac{E_1 = E_2}{f^{-1}(E_1) = f^{-1}(E_2)}$$

Soundness of Satisfiability Procedure

- To show: F satisfiable $\Rightarrow \forall j.\ u_j^* \neq u_j'^*$
- Let ψ be an interpretation that satisfies F
  - We will show that $t^* = t'^* \Rightarrow \psi(t) = \psi(t')$
- Proof by induction on the steps in congruence closure
- Base case: $(t, t') \in R$ comes from an equality in F
- Inductive case:
  - *Transivity*: $(t, t'') \in R$ because $(t, t'), (t', t'') \in R$
    - By induction, $\psi(t) = \psi(t')$ and $\psi(t') = \psi(t'')$, so $\psi(t) = \psi(t'')$
  - *Congruence*: $(f(t), f(t')) \in R$ because $(t, t') \in R$
    - By induction, $\psi(t) = \psi(t')$
    - $\psi(f(t)) = \psi(f)(\psi(t)) = \psi(f)(\psi(t')) = \psi(f(t'))$

Completeness of Satisfiability Procedure

- To show: $\forall j.\ u_j^* \neq u_j'^* \Rightarrow$ F satisfiable
- Must show a universe and an interpretation ψ s.t.
  - $\forall$ i. $\psi(t_i) = \psi(t_i')$ (1)
  - $\forall$ j. $\psi(u_j) \neq \psi(u_j')$ (2)
- Pick universe that includes representatives from the E-DAG and a special term 0
- Define ψ as follows:
  - $\psi(x) = x^*$
  - $\psi(f)(n_1, \ldots, n_k) = f(n_1, \ldots, n_k)^*$ if $f(n_1, \ldots, n_k)$ is repr in E-DAG
  - $\psi(f)(n_1, \ldots, n_k) = 0$ otherwise
- (1) & (2) satisfied by construction

Completeness (cont'd)

- Must show that ψ satisfies axioms
- Congruence is the interesting case
  - Must show that $\psi(t) = \psi(t') \Rightarrow \psi(f(t)) = \psi(f(t'))$
- Case 1: $\psi(t) = \psi(t') = 0$
  - Then $\psi(f(t)) = \psi(f(t')) = 0$
- Case 2a: $\psi(t) = \psi(t') \neq 0$ and $f(t)$ is represented
  - Then $f(t)^* = f(t')^*$, so $\psi(f(t)) = \psi(f(t'))$
- Case 2b: $\psi(t) = \psi(t') \neq 0$ and $f(t)$ is not represented
  - Then $f(t')$ is not represented, so $\psi(f(t)) = 0 = \psi(f(t'))$
- We have a constructive proof of completeness!

Convexity of Uninterpreted Functions

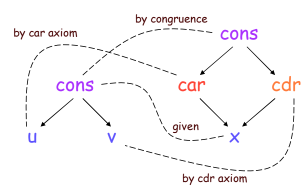
- The theory of uninterpreted functions is convex
- Proof:
  - Let E be a conjunction of equalities
  - Let E_1 through E_n be equalities
  - Suppose that E entails $E_1 \vee \ldots \vee E_n$
  - Then $E \wedge \neg E_1 \wedge \ldots \neg E_n$ is unsatisfiable
  - Now run congruence closure
    - Consider the first contradiction that we find
    - Now we have E_i such that $E \wedge \neg E_i$ is unsatisfiable!
  - Thus E entails E_i alone

Theory of Lists

- Add new symbols: car, cdr, cons
- Add new axioms:
  - $\forall x,y.\ car(cons(x, y)) = x$
  - $\forall x,y.\ cdr(cons(x, y)) = y$
- Extend congruence closure algorithm to close over these new axioms as well
- Is the extended satisfiability procedure complete?

List Example

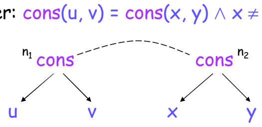
- Consider: $x = cons(u, v) \wedge cons(car(x), cdr(x)) \neq x$



- We've shown that $cons(car(x), cdr(x)) = x$
  - Thus we prove the overall formula to be unsatisfiable

List Example 2

- Consider: $cons(u, v) = cons(x, y) \wedge x \neq u$



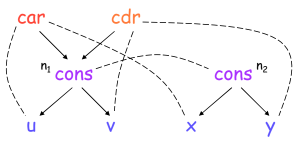
- We did not discover any contradictions
  - But this formula is unsatisfiable!
- For any interpretation ψ that satisfies the axioms:
  - $\psi(x) = \psi(car)(n_1) = \psi(car)(n_2) = \psi(u)$
  - The algorithm does not discover this equality
  - Thus our algorithm is incomplete!

Restoring Completeness

- Two possible solutions:
- Solution 1: Add new axioms
  - $\forall x,y,u,v.\ cons(u, v) = cons(x, y) \Rightarrow x = u \wedge y = v$
  - This axiom suffices for lists
  - But other theories (e.g. arrays) would need an infinite number of these axioms.
- Solution 2: Add new nodes to the graph during closure
  - If $cons(x, y) \in G$, then $car(cons(x, y)) \in G$
  - If $cons(x, y) \in G$, then $cdr(cons(x, y)) \in G$
  - These additional closure rules suffice for lists

List Example 2 (revisited)

- Consider: cons(u, v) = cons(x, y) ∧ x ≠ u
- Add new nodes…



- Now we discover all equalities, including x = u
- This new satisfiability procedure is complete!

Closure

Let's define closure more formally:

- close(G) is the closure of graph G with respect to the axioms.
- G+t is graph G extended with term t.
- We require that if G is closed, then

  Property 1: close(G+t) terminates.

  Property 2: Let G' = close(G+t). $\forall n \in G.\ repr_G(n) \equiv repr_{G'}(n)$
- Property 2 says that close(G+t) must add no new edges between existing nodes.
  - The weaker statement $\forall n_1,n_2 \in G.\ repr_G(n_1) \equiv repr_G(n_2)$ iff $repr_{G'}(n_1) \equiv repr_{G'}(n_2)$ would also encode this requirement, but the strong Property 2 helps us prove completeness.

Outline

- We require that if G is closed, then

  Property 1: close(G+t) terminates.

  Property 2: Let G' = close(G+t). $\forall n \in G.\ repr_G(n) \equiv repr_{G'}(n)$

- In the rest of this lecture, we will:
  1. Prove completeness, assuming that we have been given a closure operation that satisfies these properties.
  2. Derive a suitable closure operation for Lists.
  3. And do the same for Arrays.

Completeness

- Consider E ∧ D
  - E is conjunction of equalities
  - D is a conjunction of disequalities
- Let G_0 be a closed graph for E
  - Using any closure operation that satisfies Property 1 and Property 2.
- Statement of completeness:

  if ∀ "x≠y" ∈ D . $repr_{G_0}(x) \neq repr_{G_0}(y)$

  then E ∧ D is satisfiable

Proof of Completeness

- G_0 was built from E and satisfies D.
- Define an order for the universe U of terms.
- Define the family $G_{i+1} \triangleq close(G_i + t_i)$,

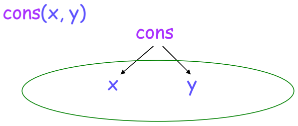
  where t_i is the smallest term not in G_i
- $\forall i.\ G_i$ is closed. (by def'n of G_i)
- $\forall i.\ G_i$ satisfies D. (by closure Property 2)
- $\forall t \in U.\ \exists k.\ t$ is represented in G_k.

  (because U must be countable for Nelson-Oppen)

- Now let $\Psi(t) = repr_{G_K}(t)$

List closure

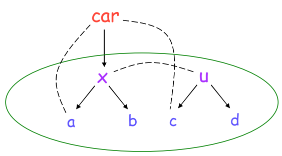
- Recall the axioms for car, cdr, cons:
  - $\forall x,y.\ car(cons(x, y)) = x$
  - $\forall x,y.\ cdr(cons(x, y)) = y$
- Close G + t with respect to these axioms, where t is not yet represented in G.
- Case: G + cons(x, y)



  - No axioms triggered, so Property 2 holds

List closure (cont'd)

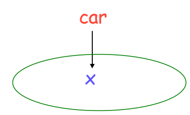
- Case: G + car(x)



- If x is a cons node, or if x equals a cons node, then add edges.
- Problem: we've violated Property 2.

List closure, 2^{nd} attempt

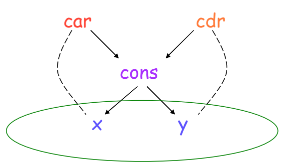
- Add two closure rules
  - Rule 1: if cons(x,y) $\in$ G, then car(cons(x,y)) $\in$ G.
  - Rule 2: if cons(x,y) $\in$ G, then cdr(cons(x,y)) $\in$ G.
- Now close w.r.t. both the axioms and the closure rules.
- Case: G + car(x)



- No axioms triggered.
  - x can't be a cons node, or else Rule 1 would already have added the car.

List closure, 2^{nd} attempt (cont'd)

- Case: G + cons(x, y)



- We also add a car and a cdr.
- The new nodes are equal to at most one equivalence class in G. (Otherwise, cons(x, y) would already be represented.) So Property 2 holds.

List closure, summary

- Add closure rules specifying when to add extra nodes to the graph.
- The extra nodes ensure we'll never have to join existing equivalence classes.
- We must also show that Property 1 (termination) holds.
  - because we can't repeat patterns.
- Using these rules, the decision procedure is complete, by the proof shown earlier.
- Claim: this theory is convex.

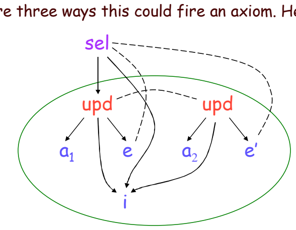
Arrays

- Axiom 1: sel(upd(a,i,e),i) = e
- Axiom 2: $i \neq j \Rightarrow$ sel(upd(a,i,e),j) = sel(a,j)
- These rules have the same completeness problem as Lists.
- What nodes can cause us to violate Property 2?
- Case: G + upd(a,i,e)
  - Nothing to do here, since the axioms only fire for sel.

Arrays (cont'd)

Axiom1: sel(upd(a,i,e),i) = e
Axiom2: $i \neq j \Rightarrow$ sel(upd(a,i,e),j) = sel(a,j)

- Case: G + sel(a,i)
  - There are three ways this could fire an axiom. Here's Axiom1:

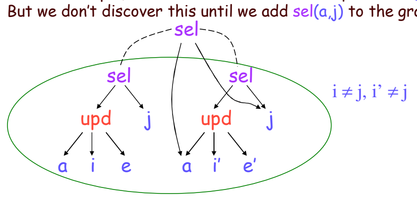


- So add Rule1: upd(a,i,e) $\in$ G $\Rightarrow$ sel(upd(a,i,e),i) $\in$ G

Arrays (cont'd)

Axiom1: sel(upd(a,i,e),i) = e
Axiom2: $i \neq j \Rightarrow$ sel(upd(a,i,e),j) = sel(a,j)

- Case: G + sel(a,i) (cont'd). This could also fire Axiom2.
  - In this example, the two sel nodes are both equal to sel(a,j). But we don't discover this until we add sel(a,j) to the graph.



So add Rule2: sel(upd(a,i,e),j) $\in G \Rightarrow$ sel(a,j) $\in G$
and Rule3: (upd(a,i,e)$\in G \wedge$ sel(a,j)$\in G$) $\Rightarrow$ sel(upd(a,i,e),j)$\in G$

Array Summary

Axiom1: sel(upd(a,i,e),i) = e
Axiom2: $i \neq j \Rightarrow$ sel(upd(a,i,e),j) = sel(a,j)

- We need three new closure rules that add nodes "early".
  - Rule1: upd(a,i,e) $\in G \Rightarrow$ sel(upd(a,i,e),i) $\in G$
  - Rule2: sel(upd(a,i,e),j) $\in G \Rightarrow$ sel(a,j) $\in G$
  - Rule3: (upd(a,i,e)$\in G \wedge$ sel(a,j)$\in G$) $\Rightarrow$ sel(upd(a,i,e),j)$\in G$
    - Note that if i = j, then the node added by Rule3 is the same as the one added by Rule1.
- What about convexity?
  - Consider upd(upd(a,i_1,x), i_2, x) = upd(a,j,x)
    - This implies $i_1 = i_2 \vee$ sel(a,i_1) = x $\vee$ sel(a,i_2) = x
  - So arrays are complete (with our extra rules), but not convex for Nelson-Oppen.
  - Axiom2 introduces a disequality. We'd need a case analysis to handle it.