Lecture 10

Invariants maintained

1. $root(root(e)) = root(e)$
   (the root of a class belongs to the class)
   (Define $class(e) = \{ e' \mid root(e') = root(e) \}$

2. $(p, d) \in$ undoStack then d is a proof of p

3. $root(a) \equiv root(b)$ iff
   - $a \equiv b$ or
   - exist $a_1, \ldots, a_{n+1}$ such that
     - $a \equiv a_1$, $b \equiv a_{n+1}$
     - for all $i = 1..n$ either
       $(a_i =_{??} a_{i+1}, \ldots) \in$ undo Stack
       or
       $(a_{i+1} = a_i, \ldots) \in$ undo Stack

4. $a \in parents(root(b))$ iff exists $e \in class(b)$
   $a \equiv f(\ldots, e, \ldots)$

5. $class(a) \cap forbid(root(b)) \neq \emptyset$ iff exist $a' \in class(a)$
   $b' \in class(b)$
   $(a' \neq b', \ldots) \in$ undoStack

Notes
- we only keep parents, forbid for representatives
- 3 says that classes are not too big
- 5 says that forbid are not too big

From. G Necula PhD Thesis "Compiling with Proofs"


```
merge(a : node, b : node, eqab : proof) =                    /*eqab is a proof of a = b */
if root(a) ≠ root(b) then
  if class(a) ∩ forbid(root(b)) ≠ ∅ then
    raise Contradiction(mkEqContra(a, b, eqab))
  else if class(b) ∩ forbid(root(a)) ≠ ∅ then
    raise Contradiction(mkEqContra(b, a, eqsym(eqab)))
  else
    undoStack = Stack.push(undoStack, (a = b, eqab))
    forbid(root(b)) = forbid(root(b)) ∪ forbid(root(a)))
    pa = parents(root(a))
    pb = parents(root(b))
    parents(root(b)) = pa ∪ pb
    foreach a' ∈ class(a)
      root(a') = root(b)
    return {(a = b, eqab)} ∪ checkCongr(pa, pb)

checkCongr(pa : node set, pb : node set) : (pred * proof) set =
  accum = ∅
  foreach a ∈ pa, b ∈ pb
    if head(a) = head(b) ∧ nrarg(a) = nrarg(b) = n ∧ root(arg_i(a)) = root(arg_i(b))
      eqab = congr(head(a), prfEq(arg_1(a), arg_1(b)), ..., prfEq(arg_n(a), arg_n(b)))
      accum = accum ∪ merge(a, b, eqab)
  return accum

forbidMerge(a : node, b : node, neqab : proof) =
  if root(a) = root(b) then
    raise Contradiction(falsei(prfEq(a, b), neqab))
  else
    undoStack = Stack.push(undoStack, (a ≠ b, neqab))
    forbid(root(b)) = forbid(root(b)) ∪ {a}

prfEq(a : node, b : node) =
  let a_1, ..., a_{n+1} as in the invariant CC3
  pf'_i = { pf_i          if (a_i = a_{i+1}, pf_i) ∈ undoStack
          { eqsym(pf_i)   if (a_{i+1} = a_i, pf_i) ∈ undoStack
  return eqtr(pf'_1, eqtr(pf'_2, ..., eqtr(pf'_{n-1}, pf'_n) ...))

mkEqContra(a, b, eqab) =                                     /* class(a) ∩ forbid(root(b)) ≠ ∅ */
  let a' ∈ class(a), b' ∈ class(b) such that (a' ≠ b', neqab) ∈ undoStack
  return falsei(eqtr(prfEq(a', a), eqtr(eqab, prfEq(b, b'))), neqab)
```

Figure 7.9: The main algorithms defining the congruence closure decision procedure.

- reprise of congruence closure from last time

Complexity of the algorithm

- n nodes, m edges $\quad n = O(m)$
- there are $O(n)$ calls to merge
- union operations are constant time
- re-rooting total cost is $O(n \log n)$ if we reroot the smaller class always
- dominating cost is of checkCongr
  - how many times is the "if" statement in checkCongr executed
  - for a pair of nodes with outdegree k they are checked for congruences at most $2k$ times (when their successors are merged)

$$\sum n_k^2 \cdot 4k^2 \leq \left(\sum 2 n_k \cdot k\right)^2 = 4 \cdot (2m)^2 = O(m^2)$$

- Complexity of algorithm is $O(m^2)$
- We can do better

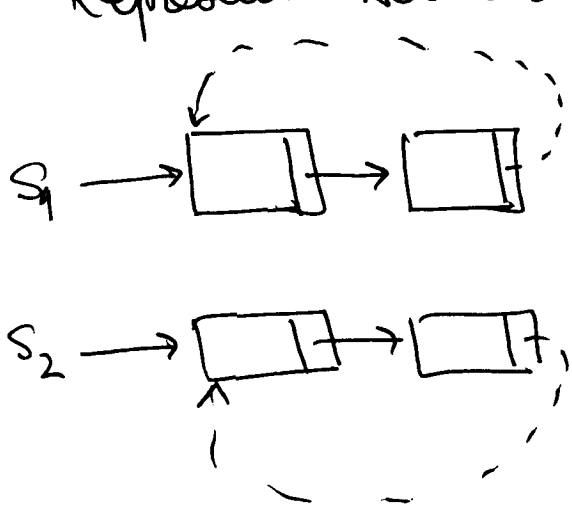
  Tarjan, Sethi, Downey "Variations on the Common Subexp. Problem" JACM v27 n4 10/1980

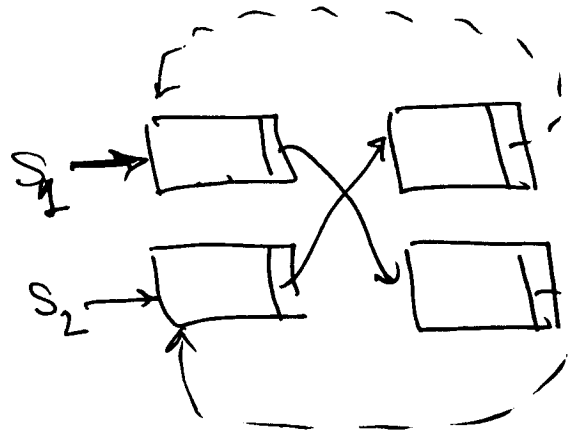
  $\longrightarrow O(m \log m)$

One more things

- We need to be able to undo
  - rerooting operation
  - union of sets (class, parents, forbid)
- We keep track on undoStack of all invokations of merge.
- To undo a set union

  Represent set as a circular list





Union

$t = s_1 \rightarrow next$

$s_1 \rightarrow next = s_2 \rightarrow next$

$s_2 \rightarrow next = t$

undo

Some operation

Extending congruence closure to a sat proc for other simple theories

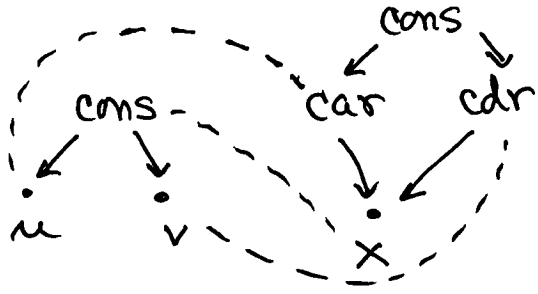
- Consider the theory with car, cdr, cons and axioms (LISP structures, with no atoms)

$$Ax: \begin{cases} \forall x, y & car(cons(x,y)) = x \\ \forall x, y & cdr(cons(x,y)) = y \end{cases}$$

- And the unsatisfiable formula

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

- Represent the terms in the E-DAG

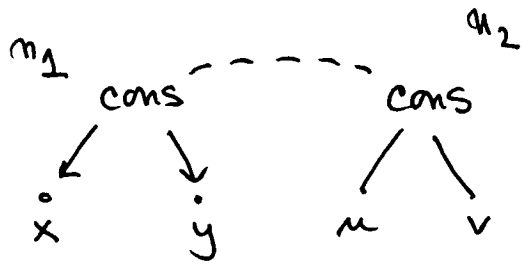


- Close with respect to equality + Ax
  (whenever you find nodes that Eq+Ax say should be equal, make them equal)
  - add $x = cons(u,v)$
  - use Ax_1 : $car(x) = u$
  - use Ax_2 : $cdr(x) = v$
  - use congruence : $cons(u,v) = cons(car(x), cdr(x))$
  - now check disequalities $\Longrightarrow$ contradiction

SEEMS TO WORK

Does this algorithm really work? Almost

- the algorithm discovers only <u>necessary</u> equalities
  - whenever an equality is discovered, it is satisfied in all interpretations that satisfy the formula F, Eg and Ax
  - whenever the algorithm says UNSAT the formula is UNSAT
  - the algorithm is sound for use as a unsat procedure (as we need)
- the algorithm <u>does not</u> discover all implied equalities
  - consider $F = cons(x,y) = cons(u,v) \wedge x \neq u$



and notice that $x^* \neq u^*$ hence report satisfiability

- but take any interpretation ψ that satisfies F, Eg and Ax

$$\psi(car)(n_1) \underset{Ax_1}{=} \psi(x)$$
$$n_1 = n_2 \;\|\qquad\qquad \|$$
$$\psi(car)(n_2) \underset{Ax_1}{=} \psi(y)$$

$\Longrightarrow \psi$ cannot satisfy $x \neq u$

- the algorithm is incomplete. May fail to ~~prove true things~~ report unsatisfiability

Problem if $x = cons(u, v) = cons(y, z)$
and $\psi(u), \psi(y)$ are defined and distinct

We cannot ~~give a rule~~ define
$\psi'(car)$ on $car(x)$ as an extension of ψ !

Solutions

1). Extend the axioms with

$\forall x y u v \quad cons(x,y) = cons(u,v) \Rightarrow x = u \wedge y = v$

- Now you can check that local extension works

2) Require that whenever $cons(x,y)$ is represented then $car(cons(x,y))$ and $cdr(cons(x,y))$ are

also represented
(Idea: DAG didn't have enough nodes to allow the closure to discover all equalities)

- Now you can check that all local extensions of $\psi(car)$, $\psi(cdr)$ and $\psi(cons)$ work
- We define the universe inductively
  U is the smallest such that
  - $terms(F) \subseteq U$
  - $n \in U \Rightarrow car(n) \in U$
  - ...

How do we fix the algorithm

· Identify the problem.

- an E-DAG G determines a canonical <u>partial</u> interpretation Ψ_G
- Ψ_G is only defined on terms that are represented in the DAG
- if the DAG is closed under $Eq + Ax$ then it satisfies the axioms (when it is defined)
- <u>Question</u>: can we extend Ψ_G to a <u>total</u> interpretation?
  - If yes then whenever we finish computing the closure with no contradictions, there exists an interpretation Ψ → we have satisfiability

(extending a partial function means to define it on more points)

- <u>Idea</u>: verify that all local extensions are possible
  - pick $n = car\ x$ such that $\Psi(x)$ is defined but $\Psi(car)(\Psi(x))$ is not