Proof-producing Congruence Closure

Robert Nieuwenhuis and Albert Oliveras

Tech. Univ. Catalonia, Barcelona

The University of Iowa
March 2005

Overview of this talk

1. SMT: Satisfiability Modulo Theories
2. Lazy Approach to SMT

   Need for Explain
3. DPLL(T) Approach to SMT

   Need for Explain
4. Solvers: Union-Find and Congruence Closure
5. Union-Find with Explain
6. Congruence Closure with Explain
7. Conclusions and Open Problem

SMT: Satisfiability Modulo Theories

Example where the theory T is $=$ (congruence):

$$g(a)=c \quad \wedge \quad c \neq d \quad \wedge \quad (f(g(a)) \neq f(c) \vee g(a)=d)$$

- Theories of interest:
  EUF [Burch and Dill '94],
  CLU [Bryant, Lahiri and Seshia '02],
  separation logic [BLS '03],
  arrays, lists, queues,
  ...

- Applications: software/hardware verification, circuit design, compiler optimization, planning, scheduling, ...

Lazy approach to SMT:

–Consider formula as propositional, i.e., "forget" theory T.

–REPEAT

SAT solver looks for a propositional model, while incremental T-solver for conjunctions of literals checks T-consistency of (partial) model being built.

If T-inconsistent, a lemma is added precluding the model.

UNTIL T-model found OR propositionally unsat.

Constraints imposed by the theory are introduced on demand.

Lazy/eager notification, online/offline SAT solver, extraction of inconsistency proofs [Barret, Dill and Levitt '96; deMoura and Ruess '02; Barret, Dill and Stump '02; Flanagan et al '03, etc]

EXAMPLE of Lazy approach: T is $=$ (congruence)

Model being built by SAT solver and being fed into T-solver:

$$\ldots \quad b{=}c, \quad \ldots \quad \ldots \quad f(b){=}c \quad \ldots \quad \ldots \quad f(c){=}a \quad \ldots$$

Upon additional input $a \neq b$: incompatible with T!
Solver must generate lemma:

$$b{=}c \wedge f(b){=}c \wedge f(c){=}a \longrightarrow a{=}b$$

because the first three atoms are the explanation of $a{=}b$.

Crucial to efficiently find small explanations among the (many) input equations!

Another SMT approach needing explanations:

DPLL(T) = General DPLL(X) engine + $Solver_T$ for given T

[GHNOT, CAV'04]

- Idea similar to CLP(X) framework for Constraint LP
- Improves upon Lazy Approach because DPLL gets pruned as well by T-consequences L (communicated by $Solver_T$) from T-consistent partial models
  (not only from T-inconsistent ones as in lazy approach).
- Also outperforms ad-hoc eager translation methods of Bryant et al on their own processor verification benchmarks.

For backjumping, DPLL(T) builds Conflict Graphs, where the predecessors of T-consequence nodes L must be the literals in the explanation of L.

Implementing the solver for EUF: Union-Find and Congruence Closure

Union-find (U-F) data structures maintain equivalence relation induced by sequence of input unions $a_1{=}b_1, a_2{=}b_2, \ldots$

Tarjan: sequence of n unions and finds in $O(n\,\alpha(n))$ time

Congruence Closure (CC) algorithms maintain a *congruence* relation given by sequence of pairs of equations between ground terms: $s_1{=}t_1, s_2{=}t_2, \ldots$

Difference w.r.t. equivalence rel.: also monotonicity axioms:

$$f(x_1 \ldots x_n){=}f(y_1 \ldots y_n) \text{ if } x_1{=}y_1 \ldots x_n{=}y_n$$

Here wlog. consider only flat eqs: $f(a, b){=}c$ or $a{=}b$ [NO03]

Sequence of n merges in $O(n \log n)$ time [e.g., DST80]

The Explain operation

INPUT: E and $s = t$ (ground equations) such that $E \models s = t$
OUTPUT: A small subset $E' \subseteq E$

But, what do we understand by small ?

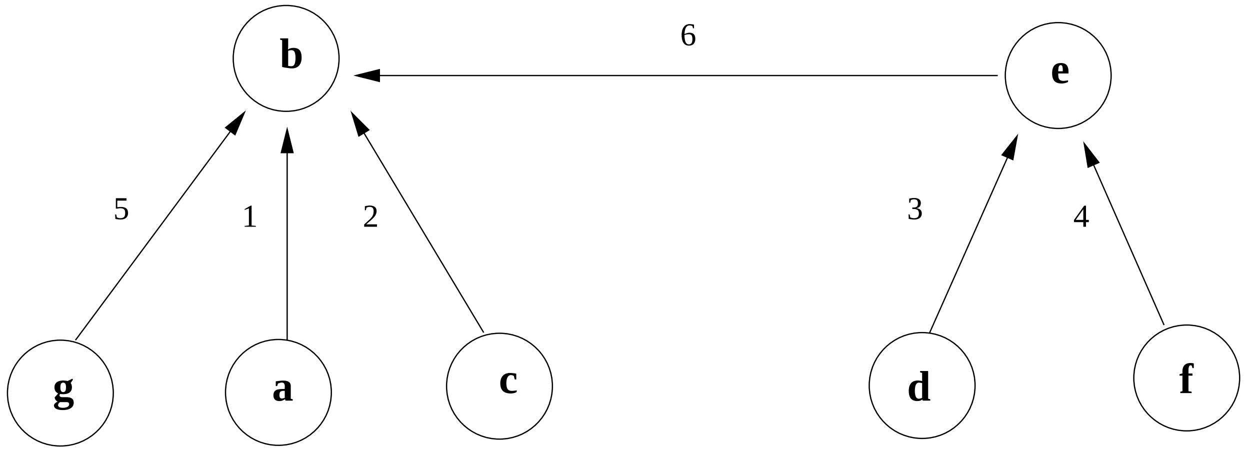
- E' is minimal if for any E'' s.t. $E'' \models s = t$ then $|E'| \leq |E''|$.
- E' is irredundant if for any $E'' \subsetneq E'$ we have $E'' \not\models s = t$.

It is clear that minimality implies irredundancy, but it may be too difficult to find minimal explanations.

Union-Find with Explain

- An irredundant explanation for $a = b$ will be of the form $a = a_1, a_1 = a_2, a_2 = a_3, \ldots a_n = b$.
- How many different irredundant explanations can we have?
- By ignoring redundant equalities, we can assume there exists only one irredundant explanation each equation.
- Therefore, in our case, irredundancy coincides with minimality.

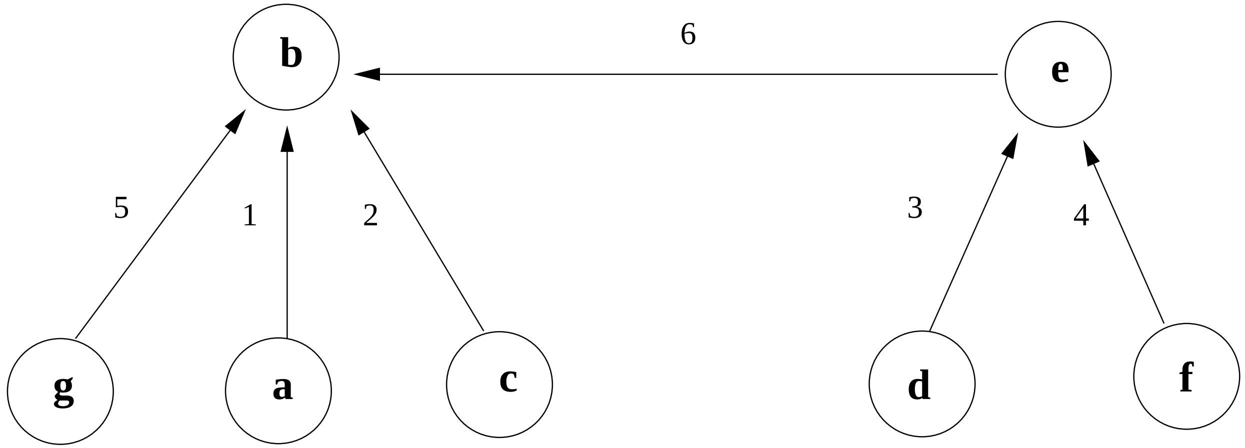
UF with Explain (First attempt)



1. $a{=}b$ 2. $c{=}a$ 3. $d{=}e$ 4. $f{=}e$ 5. $g{=}c$ 6. $c{=}f$

- Take Explain($d{=}f$) to be the equations in the paths from d and f to their nearest common ancestor, that is $d{=}e, f{=}e$. OK!
- But Explain($g{=}c$) gives $g{=}c, c{=}a$. Redudant!
- Even worse, Explain($a{=}f$) gives $a{=}b, d{=}e, c{=}f$. Not a proof!

UF with Explain (First algorithm)

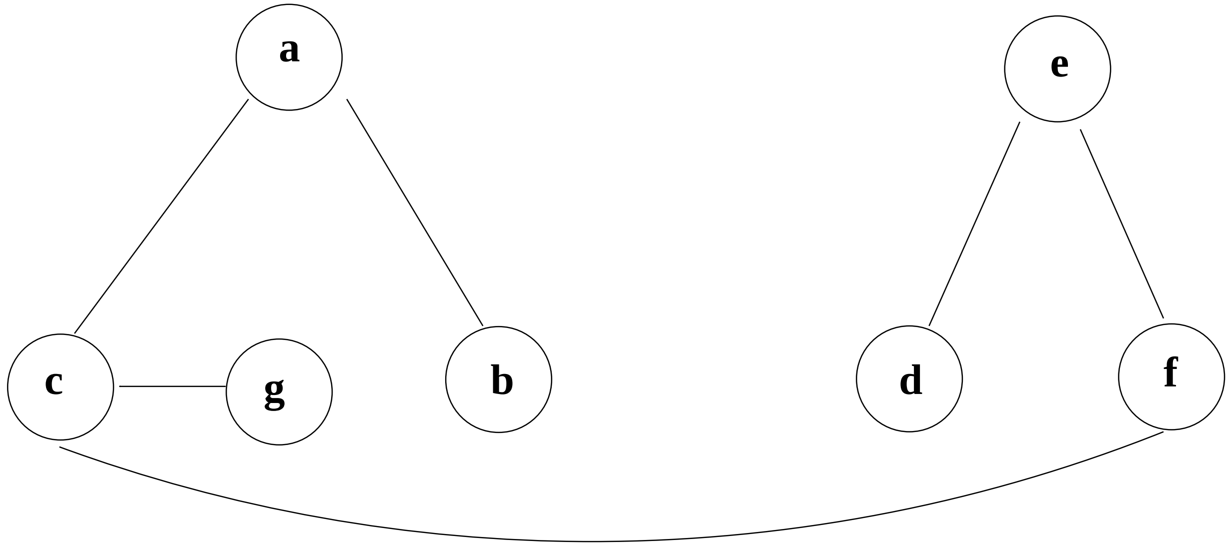


1. $a \rightarrow b$ 2. $c \rightarrow a$ 3. $d \rightarrow e$ 4. $f \rightarrow e$ 5. $g \rightarrow c$ 6. $c \leftarrow f$

- For Explain($a{=}f$) only the newest of the eqns in the paths from a to f to their NCA can be ensured to be in the proof.
- Hence, $c = f$ is part of the explanation. Now, recursive call to Explain($a{=}c$).
- Orientation of the equalities allows one to discover the recursive calls. Try Explain($g{=}d$)!!!.
- Complexity $O(k \lg n)$ for a proof of size k.

UF with Explain (Second algorithm)

- If we consider G the graph whose edges are the unions, looking for Explain(c=e) amounts to looking for the path between c and e.

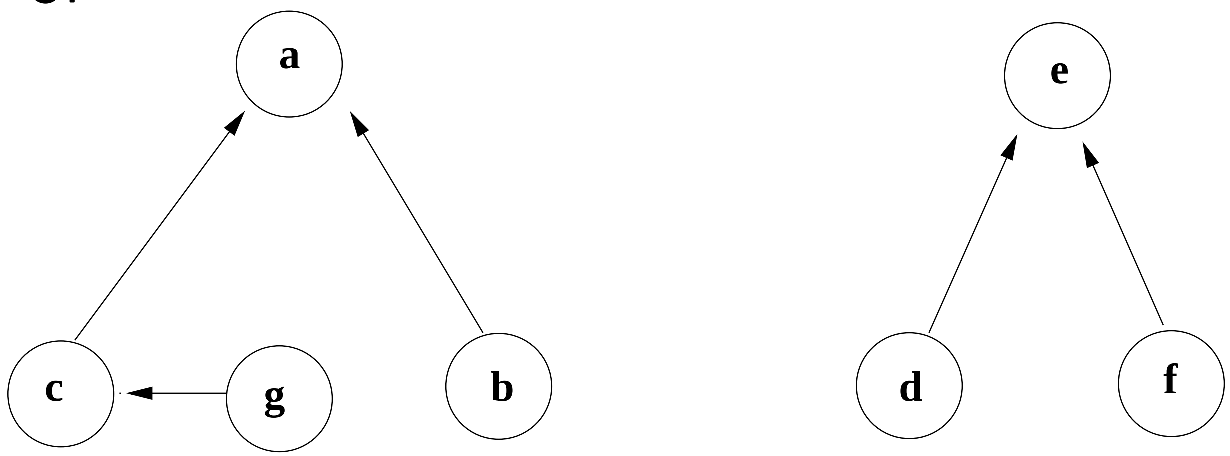


1. a=b 2. c=a 3. d=e 4. f=e 5. g=c 6. c=f

- How to find the path efficiently? We will use directed edges and rooted trees.

UF with Explain (Second alg. cntd.)

- Suppose we have the following trees after adding edge number 5.

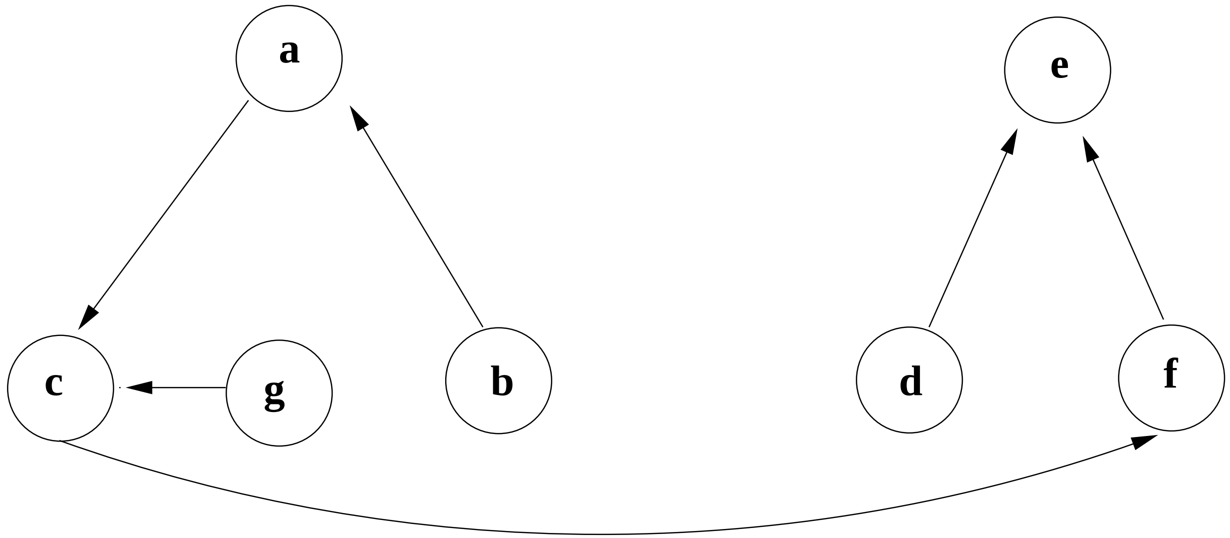


1. $a{=}b$ 2. $c{=}a$ 3. $d{=}e$ 4. $f{=}e$ 5. $g{=}c$ 6. $c{=}f$

- Any orientation of $c{=}f$ breaks the desired structure.
- **SOLUTION:** orient $c \rightarrow f$ and reverse all edges between c and its root.

UF with Explain (Second alg. cntd.)

- Suppose we have the following trees after adding edge number 5.



1. $a{=}b$ 2. $c{=}a$ 3. $d{=}e$ 4. $f{=}e$ 5. $g{=}c$ 6. $c{=}f$

- Any orientation of $c{=}f$ breaks the desired structure.
- Complexity: $O(k)$ for a proof of size k but now UF becomes $O(n \lg n)$ (if smart orientation is chosen).

Congruence Closure with Explain (1)

Try to give a modular view here: separate CC from Expl

Each new equality between constants a and b can come from:

- A single input merge $a{=}b$, or
- Two input merges $f(a_1, a_2){=}a$ and $f(b_1, b_2){=}b$

First idea: keep proof forest as in Alg.2 for U-F, where edges are labelled with the corresponding input merges.

Example: 1. $f(a_1){=}a$ 2. $f(b_1){=}b$ 3. $c{=}b$ 4. $a_1{=}b_1$

After 4., the proof forest is: $a_1 \xrightarrow{4} b_1$ $a \xrightarrow{1,2} b \xleftarrow{3} c$

$Expl(a{=}c)$ finds 1,2,3 and, recursively, 4.

Cost: $O(k\,\alpha(k))$, where CC still $O(n \log n)$.
The α comes from tricks to avoid exploring twice same edges.

Congruence Closure with Explain (2)

Problematic example for first idea:

1. $f(a_1)=a$ 2. $f(b_1)=b$ 3. $f(c_1)=c$ 4. $a_1=b_1$ 5. $a_1=c_1$

After 4., the proof forest is: $a_1 \xrightarrow{4} b_1$ $a \xrightarrow{1,2} b$

After 5., the proof forest is: $c_1 \xrightarrow{5} a_1 \xleftarrow{4} b_1$ $a \xrightarrow{1,2} b \xleftarrow{2,3} c$

$Expl(a=c)$ finds 1,2,3 and, recursively, 4,5.

But its subset 1,3,5 is already a proof!

Congruence Closure with Explain (3)

Second idea: Maintain proof forest where:

- edges are for input merges between cts. only
- nodes are classes of cts equal by direct monotonicity

Wlog. assume c occurs at most once as rhs of $f(a,b)=c$ eqs.

Previous example ok now:

1. $f(a_1)=a$ 2. $f(b_1)=b$ 3. $f(c_1)=c$ 4. $a_1=b_1$ 5. $a_1=c_1$

After 4., the proof forest is: $a_1 \xrightarrow{4} b_1$ $[a,b]$

After 5., the proof forest is: $c_1 \xrightarrow{5} a_1 \xleftarrow{4} b_1$ $[a,b,c]$

$Expl(a=c)$ finds eqs 1,3 (the only ones of which a and c are rhs) and, recursively, 5.

Congruence Closure with Explain (4)

Another example:

1. $f(a_1){=}a$ 2. $f(b_1){=}b$ 3. $f(c_1){=}c$ 4. $f(d_1){=}d$

5. $a_1{=}b_1$ 6. $c_1{=}d_1$ 7. $a{=}e$ 8. $d{=}e$

After 6.: $a_1 \xrightarrow{5} b_1$ $c_1 \xrightarrow{6} d_1$ $[a,b]$ $[c,d]$

After 7.: $a_1 \xrightarrow{5} b_1$ $c_1 \xrightarrow{6} d_1$ $[a,b] \xrightarrow{7} e$ $[c,d]$

After 8.: $a_1 \xrightarrow{5} b_1$ $c_1 \xrightarrow{6} d_1$ $[a,b] \xrightarrow{7} e \xleftarrow{8} [c,d]$

$Expl(b{=}c)$ finds eqs 7,8 (for going to common ancestor e).
For using 7., $Expl(a{=}b)$ is required: 1,2, and, recursively, 5.
For using 8., $Expl(c{=}d)$ is required: 3,4, and, recursively, 6.

Congruence Closure with Explain (5)

Maintaining class nodes while keeping CC $O(n \log n)$ can be done if class merge causes tree merge.

But...[this is our current problem]   Same example continued:

1. $f(a_1)=a$ 2. $f(b_1)=b$ 3. $f(c_1)=c$ 4. $f(d_1)=d$

5. $a_1=b_1$ 6. $c_1=d_1$ 7. $a=e$ 8. $d=e$ 9. $b_1=c_1$

After 8.: $a_1 \xrightarrow{5} b_1$ $c_1 \xrightarrow{6} d_1$ $[a,b] \xrightarrow{7} e \xleftarrow{8} [c,d]$

Should 9. $b_1=c_1$ have any effect on the rightmost tree? **Yes:**

Assume 9 belongs to our explanation, and recursively we call $Expl(b=c)$. Then $\{3,4,\ldots,9\}$ suffices, instead of $\{1,2,3,4,5,6,7,8,\ldots,9\}$.

Thm: minimal proofs if $\nexists$ two congruent non-singleton classes.

Cong. Cl. w/ Explain (Conclusions)

Interesting open problem:

- Keep CC $O(n \log n)$ and
- Find k-step proofs depending only on k (Ours: $O(k\ \alpha(k))$)
- Proofs minimal (w.r.t. $\subseteq$). This is where we sometimes fail.

In practice (on large set of industrial benchmarks of Bryant et al for EUF and EUF w/ integer offsets):

- Algorithm given here finds almost always (99 % of the cases?) minimal explanations
- Most explanations are small: 4 or 5 steps, sometimes 10.

Many details omitted: Incrementality, Backtracking, Theory Extensions,...

Stay tuned at www.lsi.upc.es/~oliveras