Decision Procedures for Verification

Decision Procedures (3)

12.01.2017

Viorica Sofronie-Stokkermans

sofronie@uni-koblenz.de

Until now:

Decision Procedures

- Uninterpreted functions

  congruence closure

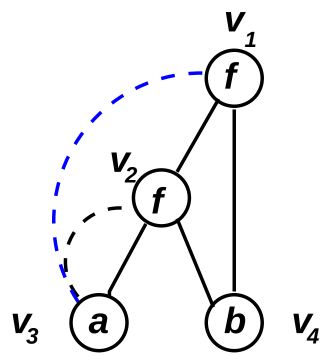
DAG Representation/Congruence Closure

Task: Check if $(s_1(\overline{c}) \approx t_1(\overline{c}) \wedge \cdots \wedge s_k(\overline{c}) \approx t_k(\overline{c}) \wedge s(\overline{c}) \not\approx t(\overline{c}))$ unsatisfiable.

Solution [Downey-Sethi, Tarjan'76; Nelson-Oppen'80]

- represent the terms occurring in the problem as DAG's
- represent premise equalities by a relation on the vertices of the DAG

Example: Check whether $f(f(a, b), b) \approx a$ is a consequence of $f(a, b) \approx a$.



v_1 : $f(f(a, b), b)$
v_2 : $f(a, b)$
v_3 : a
v_4 : b

R : $\{(v_2, v_3)\}$

- compute the "congruence closure" R^c of R
- check whether $(v_1, v_3) \in R^c$

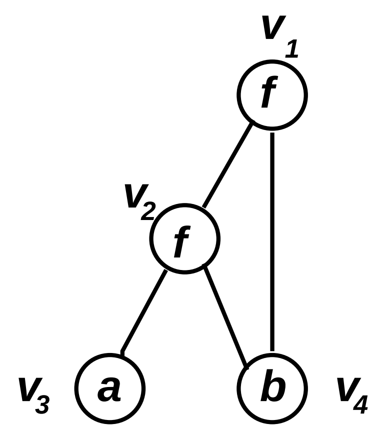
Computing the congruence closure of a DAG

- **DAG structures:**
  - $G = (V, E)$ directed graph
  - Labelling on vertices

    $\lambda(v)$: label of vertex v
    $\delta(v)$: outdegree of vertex v

  - Edges leaving the vertex v are ordered
    ($v[i]$: denotes i-th successor of v)

Example



$\lambda(v_1) = \lambda(v_2) = f$
$\lambda(v_3) = a, \lambda(v_4) = b$

$\delta(v_1) = \delta(v_2) = 2$
$\delta(v_3) = \delta(v_4) = 0$

$v_1[1] = v_2, v_2[2] = v_4$

...

Congruence closure of a DAG/Relation

Given: $G = (V, E)$ DAG + labelling

$R \subseteq V \times V$

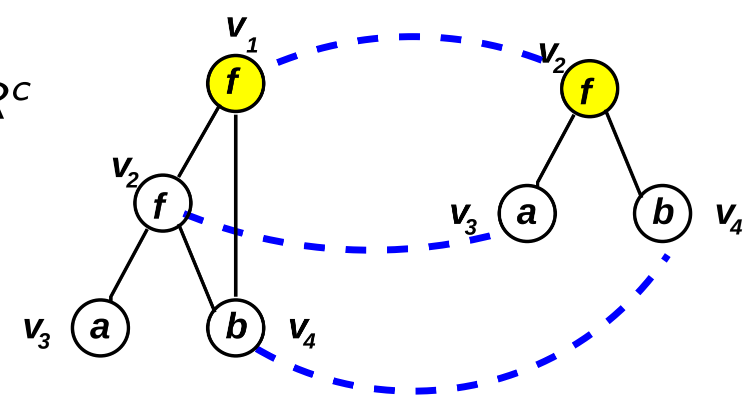
The congruence closure of R is the smallest relation R^c on V which is:

- reflexive
- symmetric
- transitive
- congruence:

  If $\lambda(u) = \lambda(v)$ and $\delta(u) = \delta(v)$

  and for all $1 \leq i \leq \delta(u)$: $(u[i], v[i]) \in R^c$

  then $(u, v) \in R^c$.



Congruence closure of a relation

Recursive definition

$$\frac{(u, v) \in R}{(u, v) \in R^c}$$

$$\frac{}{(v, v) \in R^c} \qquad \frac{(u, v) \in R^c}{(v, u) \in R^c} \qquad \frac{(u, v) \in R^c \quad (v, w) \in R^c}{(u, w) \in R^c}$$

$$\frac{\lambda(u) = \lambda(v) \quad u, v \text{ have } n \text{ successors} \quad \text{and } (u[i], v[i]) \in R^c \text{ for all } 1 \leq i \leq n}{(u, v) \in R^c}$$

- The congruence closure of R is the smallest set closed under these rules

Congruence closure and UIF

Assume that we have an algorithm $\mathbb{A}$ for computing the congruence closure of a graph G and a set R of pairs of vertices

- Use $\mathbb{A}$ for checking whether $\bigwedge_{i=1}^{n} s_i \approx t_i \wedge \bigwedge_{j=1}^{m} s'_j \not\approx t'_j$ is satisfiable.

  (1) Construct graph corresponding to the terms occurring in s_i, t_i, s'_j, t'_j

  Let v_t be the vertex corresponding to term t

  (2) Let $R = \{(v_{s_i}, v_{t_i}) \mid i \in \{1, \dots, n\}\}$

  (3) Compute R^c.

  (4) Output "Sat" if $(v_{s'_j}, v_{t'_j}) \notin R^c$ for all $1 \leq j \leq m$, otherwise "Unsat"

Theorem 3.3.3 (Correctness)

$\bigwedge_{i=1}^{n} s_i \approx t_i \wedge \bigwedge_{j=1}^{m} s'_j \not\approx t'_j$ is satisfiable iff $[v_{s'_j}]_{R^c} \neq [v_{t'_j}]_{R^c}$ for all $1 \leq j \leq m$.

Congruence closure and UIF

Theorem 3.3.3 (Correctness)

$\bigwedge_{i=1}^{n} s_i \approx t_i \wedge \bigwedge_{j=1}^{m} s'_j \not\approx t'_j$ is satisfiable iff $[v_{s'_j}]_{R^c} \neq [v_{t'_j}]_{R^c}$ for all $1 \leq j \leq m$.

Proof ($\Rightarrow$)

Assume $\mathcal{A}$ is a Σ-structure such that $\mathcal{A} \models \bigwedge_{i=1}^{n} s_i \approx t_i \wedge \bigwedge_{j=1}^{m} s'_j \not\approx t'_j$.

We can show that $[v_s]_{R^c} = [v_t]_{R^c}$ implies that $\mathcal{A} \models s = t$ (Exercise).

(We use the fact that if $[v_s]_{R^c} = [v_t]_{R^c}$ then there is a derivation for $(v_s, v_t) \in R^c$ in the calculus defined before; use induction on length of derivation to show that $\mathcal{A} \models s = t$.)

As $\mathcal{A} \models s'_j \not\approx t'_j$, it follows that $[v_{s'_j}]_{R^c} \neq [v_{t'_j}]_{R^c}$ for all $1 \leq j \leq m$.

Congruence closure and UIF

Theorem 3.3.3 (Correctness)

$\bigwedge_{i=1}^{n} s_i \approx t_i \wedge \bigwedge_{j=1}^{m} s'_j \not\approx t'_j$ is satisfiable iff $[v_{s'_j}]_{R^c} \neq [v_{t'_j}]_{R^c}$ for all $1 \leq j \leq m$.

Proof($\Leftarrow$) Assume that $[v_{s'_j}]_{R^c} \neq [v_{t'_j}]_{R^c}$ for all $1 \leq j \leq m$. We construct a structure that satisfies $\bigwedge_{i=1}^{n} s_i \approx t_i \wedge \bigwedge_{j=1}^{m} s'_j \not\approx t'_j$

- Universe is quotient of V w.r.t. R^c plus new element 0.
- c constant $\mapsto c_{\mathcal{A}} = [v_c]_{R^c}$.
- $f/n \mapsto f_{\mathcal{A}}([v_1]_{R^c}, \ldots, [v_n]_{R^c}) = \begin{cases} [v_{f(t_1,\ldots,t_n)}]_{R^c} & \text{if } v_{f(t_1,\ldots,t_n)} \in V, \\ & [v_{t_i}]_{R^c} = [v_i]_{R^c} \text{ for } 1 \leq i \leq n \\ 0 & \text{otherwise} \end{cases}$

well-defined because R^c is a congruence.

- It holds that $\mathcal{A} \models s'_j \not\approx t'_j$ and $\mathcal{A} \models s_i \approx t_i$

Computing the congruence closure of a DAG

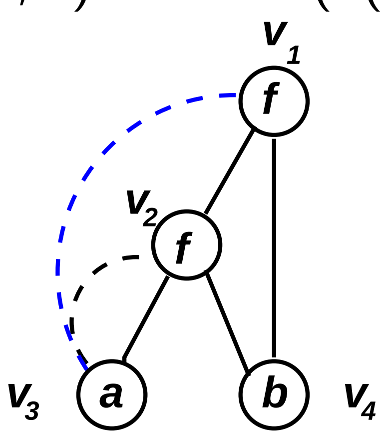
Given: $G = (V, E)$ DAG + labelling

$R \subseteq V \times V$

Task: Compute R^c (the congruence closure of R)

Example:

$f(a, b) \approx a \rightarrow f(f(a, b), b) \approx a$



$R = \{(v_2, v_3)\}$

Idea:

- Start with the identity relation $R^c = Id$
- Successively add new pairs of nodes to R^c;
  close relation under congruence.

Task: Compute R^c

Computing the congruence closure of a DAG

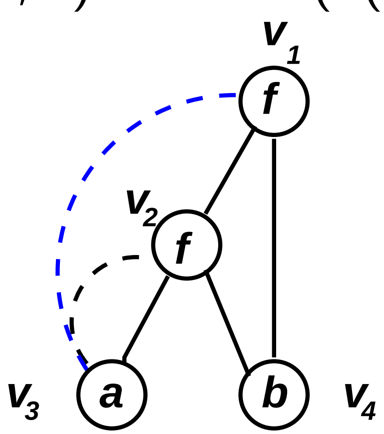
Given: $G = (V, E)$ DAG + labelling

$R \subseteq V \times V$; $(v, v') \in V^2$

Task: Check whether $(v, v') \in R^c$

Example:

$f(a, b) \approx a \rightarrow f(f(a, b), b) \approx a$



$R = \{(v_2, v_3)\}$

Idea:

- Start with the identity relation $R^c = Id$
- Successively add new pairs of nodes to R^c; close relation under congruence.

Task: Decide whether $(v_1, v_3) \in R^c$

Computing the congruence closure of a DAG

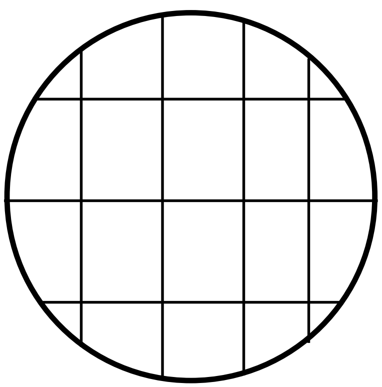
Given: $G = (V, E)$ DAG + labelling

$R \subseteq V \times V$

Task: Compute R^c (the congruence closure of R)

Idea: Recursively construct relations closed under congruence R_i (approximating R^c) by identifying congruent vertices u, v and computing $R_{i+1} :=$ congruence closure of $R_i \cup \{(u, v)\}$.

Representation:



- Congruence relation $\mapsto$ corresponding partition

Computing the congruence closure of a DAG

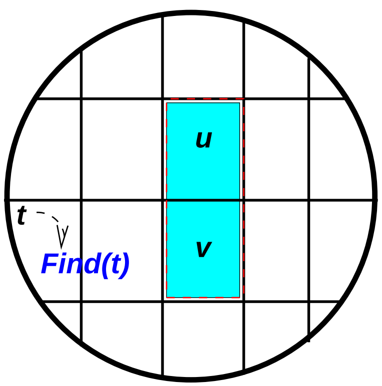
Given: $G = (V, E)$ DAG + labelling

$R \subseteq V \times V$

Task: Compute R^c (the congruence closure of R)

Idea: Recursively construct relations closed under congruence R_i (approximating R^c) by identifying congruent vertices u, v and computing $R_{i+1} :=$ congruence closure of $R_i \cup \{(u, v)\}$.

Representation:



- Congruence relation $\mapsto$ corresponding partition
- Use procedures which operate on the partition:

  FIND(u): unique name of equivalence class of u

  UNION(u, v) combines equivalence classes of u, v

  finds repr. t_u, t_v of equiv.cl. of u, v; sets FIND(u) to

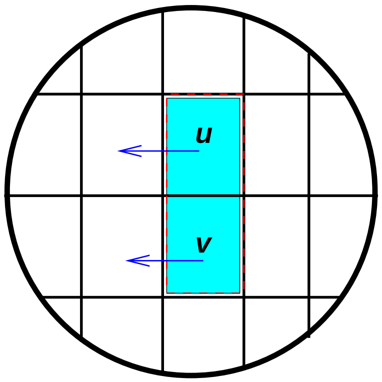
Computing the congruence closure of a DAG

MERGE(u, v)

g

Input:	$G = (V, E)$ DAG + labelling
	R relation on V closed under congruence
	$u, v \in V$
Output:	the congruence closure of $R \cup \{(u, v)\}$

If FIND(u) = FIND(v) [same canonical representative] **then** Return
If FIND(u) $\neq$ FIND(v) **then** [merge u, v; recursively-predecessors]
P_u := set of all predecessors of vertices w with FIND(w) = FIND(u)
P_v := set of all predecessors of vertices w with FIND(w) = FIND(v)
Call UNION(u, v) [merge congruence classes]
For all $(x, y) \in P_u \times P_v$ **do**: [merge congruent predecessors]
if FIND(x) $\neq$ FIND(y) **and** CONGRUENT(x, y) **then** MERGE(x, y)



CONGRUENT(x, y)

if $\lambda(x) \neq \lambda(y)$ **then** Return FALSE
For $1 \leq i \leq \delta(x)$ **if** FIND($x[i]$) $\neq$ FIND($y[i]$) **then** Return FALSE

Return TRUE.

Correctness

Proof:

(1) Returned equivalence relation is not too coarse

If x, y merged then $(x, y) \in (R \cup \{(u, v)\})^c$
(UNION only on initial pair and on congruent pairs)

(2) Returned equivalence relation is not too fine

If x, y vertices s.t. $(x, y) \in (R \cup \{(u, v)\})^c$ then they are merged by the algorithm.
Induction of length of derivation of (x, y) from $(R \cup \{(u, v)\})^c$

(1) $(x, y) \in R$ OK (they are merged)
(2) $(x, y) \notin R$. The only non-trivial case is the following:
$\lambda(x) = \lambda(y), x, y$ have n successors x_i, y_i where
$(x_i, y_i) \in (R \cup \{(u, v)\})^c$ for all $1 \leq i \leq b$.
Induction hypothesis: (x_i, y_i) are merged at some point

(become equal during some call of UNION(a, b), made in some MERGE(a, b))
Successor of x equivalent to a (or b) before this call of UNION; same for y.

$\Rightarrow$ MERGE must merge x and y

Computing the Congruence Closure

Let $G = (V, E)$ graph and $R \subseteq V \times V$

$CC(G, R)$ computes the R^c:

(1) $R_0 := \emptyset$; $i := 1$

(2) while R contains "fresh" elements do:

pick "fresh" element $(u, v) \in R$

$R_i :=$ MERGE(u, v) for G and R_{i-1}; $i := i + 1$.

Complexity: $O(n^2)$
Downey-Sethi-Tarjan congruence closure algorithm:
more sophisticated version of MERGE (complexity $O(n \cdot logn)$)

Reference: G. Nelson and D.C. Oppen. Fast decision procedures based on congruence closure. Journal of the ACM, 27(2):356-364, 1980.

Decision procedure for the QF theory of equality

Signature: Σ (function symbols)

Problem: Test satisfiability of the formula

$$F \quad = \quad s_1 \approx t_1 \wedge \cdots \wedge s_n \approx t_n \quad \wedge \quad s'_1 \not\approx t'_1 \wedge \cdots \wedge s'_m \not\approx t'_m$$

Solution: Let S_F be the set of all subterms occurring in F

1. Construct the DAG for S_F; $R_0 = Id$
2. [Build R_n the congruence closure of $\{(v(s_1), v(t_1)), \dots, (v(s_n), v(t_n))\}$]

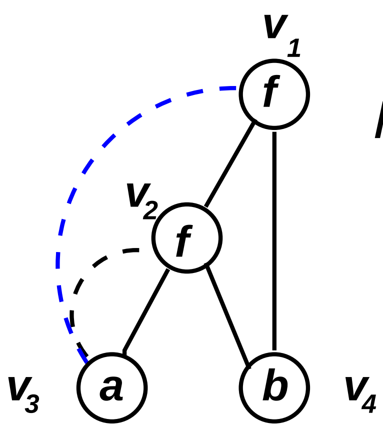
   For $i \in \{1, \dots, n\}$ **do** $R_i :=$ MERGE(v_{s_i}, v_{t_i}) w.r.t. R_{i-1}
3. **If** FIND$(v_{s'_j})$ = FIND$(v_{t'_j})$ for some $j \in \{1, \dots, m\}$ **then** return unsatisfiable
4. **else** [if FIND$(v_{s'_j}) \neq$ FIND$(v_{t'_j})$ for all $j \in \{1, \dots, m\}$] **then** return satisfiable

Example

$f(a,b) \approx a \rightarrow f(f(a,b),b) \approx a$

Test: unsatisfiability of

$f(a,b) \approx a \wedge f(f(a,b),b) \not\approx a$



$R = \{(v_2, v_3)\}$

Task:

- Compute R^c
- Decide whether $(v_1, v_3) \in R^c$

Solution:

1. Construct DAG in the figure; $R_0 = Id$.
2. Compute $R_1 := \text{MERGE}((v_2, v_3)$

   [Test representatives]

   $\text{FIND}(v_2) = v_2 \neq v_3 = \text{FIND}(v_3)$

   $P_{v_2} := \{v_1\}; P_{v_3} := \{v_2\}$

   [Merge congruence classes]

   $\text{UNION}(v_2, v_3)$: sets $\text{FIND}(v_2)$ to v_3.

   [Compute and recursively merge predecessors]

   Test: $\text{FIND}(v_1) = v_1 \neq v_3 = \text{FIND}(v_2)$

   $\text{CONGR}(v_1, v_2)$

   $\text{MERGE}(v_1, v_2)$: (different representatives)

   calls $\text{UNION}(v_1, v_2)$ which

   sets $\text{FIND}(v_1)$ to v_3.
3. Test whether $\text{FIND}(v_1) = \text{FIND}(v_3)$. Yes.

   Return unsatisfiable.