Decision Procedures in Verification

Decision Procedures (1)

17.12.2018

Viorica Sofronie-Stokkermans

e-mail: sofronie@uni-koblenz.de

Until now:

Syntax (one-sorted signatures vs. many-sorted signatures)

Semantics

Theories (Syntactic vs. Semantics view)

Herbrand models $\mapsto$ The Bernays-Schönfinkel class

Algorithmic Problems

Decidability/Undecidability

Methods: Ordered Resolution with Selection

$\mapsto$ Craig Interpolation

$\mapsto$ redundancy

Decidable classes:

Bernays-SchPönfinkel class, Ackermenn class, Monadic class

3.2 Deduction problems

Satisfiability w.r.t. a theory

Satisfiability w.r.t. a theory

Example

Let $\Sigma = (\{e/0, */2, i/1\}, \emptyset)$

Let $\mathcal{F}$ consist of all (universally quantified) group axioms:

$$\begin{array}{ll} \forall x, y, z & x * (y * z) \approx (x * y) * z \\ \forall x & x * i(x) \approx e \quad \wedge \quad i(x) * x \approx e \\ \forall x & x * e \approx x \quad \wedge \quad e * x \approx x \end{array}$$

Question: Is $\forall x, y (x * y = y * x)$ entailed by $\mathcal{F}$?

Satisfiability w.r.t. a theory

Example

Let $\Sigma = (\{e/0, */2, i/1\}, \emptyset)$

Let $\mathcal{F}$ consist of all (universally quantified) group axioms:

$$\begin{array}{ll} \forall x, y, z & x * (y * z) \approx (x * y) * z \\ \forall x & x * i(x) \approx e \quad \wedge \quad i(x) * x \approx e \\ \forall x & x * e \approx x \quad \wedge \quad e * x \approx x \end{array}$$

Question: Is $\forall x, y(x * y = y * x)$ entailed by $\mathcal{F}$?

Alternative question:

Is $\forall x, y(x * y = y * x)$ true in the class of all groups?

Logical theories

Syntactic view

first-order theory: given by a set $\mathcal{F}$ of (closed) first-order Σ-formulae.

the models of $\mathcal{F}$: $\quad \mathsf{Mod}(\mathcal{F}) = \{\mathcal{A} \in \Sigma\text{-alg} \mid \mathcal{A} \models G, \text{ for all } G \text{ in } \mathcal{F}\}$

Semantic view

given a class $\mathcal{M}$ of Σ-algebras

the first-order theory of $\mathcal{M}$: $\mathsf{Th}(\mathcal{M}) = \{G \in F_\Sigma(X) \text{ closed} \mid \mathcal{M} \models G\}$

Decidable theories

Let $\Sigma = (\Omega, \Pi)$ be a signature.

$\mathcal{M}$: class of Σ-algebras. $\mathcal{T} = \mathsf{Th}(\mathcal{M})$ is decidable

iff

there is an algorithm which, for every closed first-order formula ϕ, can decide (after a finite number of steps) whether ϕ is in $\mathcal{T}$ or not.

$\mathcal{F}$: class of (closed) first-order formulae.

The theory $\mathcal{T} = \mathsf{Th}(\mathsf{Mod}(\mathcal{F}))$ is decidable

iff

there is an algorithm which, for every closed first-order formula ϕ, can decide (in finite time) whether $\mathcal{F} \models \phi$ or not.

Examples

Undecidable theories

- $\text{Th}((\mathbb{Z}, \{0, 1, +, *\}, \{\leq\}))$
- Peano arithmetic

- $\text{Th}(\Sigma\text{-alg})$

Peano arithmetic

Peano axioms:

$\forall x\, \neg(x + 1 \approx 0)$	(zero)
$\forall x \forall y\, (x + 1 \approx y + 1 \to x \approx y$	(successor)
$F[0] \wedge (\forall x\, (F[x] \to F[x + 1]) \to \forall x F[x])$	(induction)
$\forall x\, (x + 0 \approx x)$	(plus zero)
$\forall x, y\, (x + (y + 1) \approx (x + y) + 1)$	(plus successor)
$\forall x, y\, (x * 0 \approx 0)$	(times 0)
$\forall x, y\, (x * (y + 1) \approx x * y + x)$	(times successor)

$3 * y + 5 > 2 * y$ expressed as $\exists z(z \neq 0 \wedge 3 * y + 5 \approx 2 * y + z)$

Intended interpretation: $(\mathbb{N}, \{0, 1, +, *\}, \{\approx, \leq\})$
(does not capture true arithmetic by Goedel's incompleteness theorem)

Examples

Undecidable theories

- Th$((\mathbb{Z}, \{0, 1, +, *\}, \{\leq\}))$
- Peano arithmetic
- Th(Σ-alg)

Idea of undecidability proof: Suppose there is an algorithm P that, given a formula in one of the theories above decides whether that formula is valid.

We use P to give a decision algorithm for the language

$\{(G(M), w) | G(M)$ is the Gödelisation of a TM M that accepts the string w $\}$

As the latter problem is undecidable, this will show that P cannot exist.

Examples

Undecidable theories

- $\bullet$Th(($\mathbb{Z}$, $\{0, 1, +, *\}$, $\{\leq\}$))
- Peano arithmetic

- $\bullet$Th(Σ-alg)

Idea of undecidability proof: (ctd)

(1) For Th(($\mathbb{Z}$, $\{0, 1, +, *\}$, $\{\leq\}$)) and Peano arithmetic:

multiplication can be used for modeling Gödelisation

(2) For Th(Σ-alg):

Given M and w, we create a FOL signature and a set of formulae over this signature encoding the way M functions, and a formula which is valid iff M accepts w.

Examples

In order to obtain decidability results:

- Restrict the signature
- Enrich axioms
- Look at certain fragments

Examples

In order to obtain decidability results:

- Restrict the signature
- Enrich axioms
- Look at certain fragments

Decidable theories

- Presburger arithmetic decidable in 3EXPTIME [Presburger'29]
  Signature: $(\{0, 1, +\}, \{\approx, \leq\})$ (no $*$)

  Axioms { (zero), (successor), (induction), (plus zero), (plus successor) }

- $\text{Th}(\mathbb{Z}_+)$ $\mathbb{Z}_+ = (\mathbb{Z}, 0, s, +, \leq)$ the standard interpretation of integers.

Examples

In order to obtain decidability results:

- Restrict the signature
- Enrich axioms
- Look at certain fragments

Decidable theories

- The theory of real numbers (with addition and multiplication) is decidable in 2EXPTIME [Tarski'30]

Examples

In order to obtain decidability results:

- Restrict the signature
- Enrich axioms
- Look at certain fragments

Problems

$\mathcal{T}$: first-order theory in signature Σ; $\mathcal{L}$ class of (closed) Σ-formulae

Given ϕ in $\mathcal{L}$, is it the case that $\mathcal{T} \models \phi$?

Common restrictions on $\mathcal{L}$

	Pred $= \emptyset$	$\{\phi \in \mathcal{L} \mid \mathcal{T} \models \phi\}$
$\mathcal{L}=\{\forall x A(x) \mid A \text{ atomic}\}$	word problem	
$\mathcal{L}=\{\forall x(A_1 \wedge \ldots \wedge A_n \rightarrow B) \mid A_i, B \text{ atomic}\}$	uniform word problem	$Th_{\forall Horn}$
$\mathcal{L}=\{\forall x C(x) \mid C(x) \text{ clause}\}$	clausal validity problem	$Th_{\forall, cl}$
$\mathcal{L}=\{\forall x \phi(x) \mid \phi(x) \text{ unquantified}\}$	universal validity problem	$Th_{\forall}$
$\mathcal{L}=\{\exists x A_1 \wedge \ldots \wedge A_n \mid A_i \text{ atomic}\}$	unification problem	$Th_{\exists}$
$\mathcal{L}=\{\forall x \exists x A_1 \wedge \ldots \wedge A_n \mid A_i \text{ atomic}\}$	unification with constants	$Th_{\forall\exists}$

$\mathcal{T}$-validity vs. $\mathcal{T}$-satisfiability

$\mathcal{T}$-validity: Let $\mathcal{T}$ be a first-order theory in signature Σ
Let $\mathcal{L}$ be a class of (closed) Σ-formulae

Given ϕ in $\mathcal{L}$, is it the case that $\mathcal{T} \models \phi$?

Remark: $\mathcal{T} \models \phi$ iff $\mathcal{T} \cup \neg\phi$ unsatisfiable

Every $\mathcal{T}$-validity problem has a dual $\mathcal{T}$-satisfiability problem:

$\mathcal{T}$-satisfiability: Let $\mathcal{T}$ be a first-order theory in signature Σ
Let $\mathcal{L}$ be a class of (closed) Σ-formulae

$$\neg\mathcal{L} = \{\neg\phi \mid \phi \in \mathcal{L}\}$$

Given ψ in $\neg\mathcal{L}$, is it the case that $\mathcal{T} \cup \psi$ is satisfiable?

$\mathcal{T}$-validity vs. $\mathcal{T}$-satisfiability

Common restrictions on $\mathcal{L}$ / $\neg\mathcal{L}$

$\mathcal{L}$	$\neg\mathcal{L}$
$\{\forall x A(x) \mid A \text{ atomic}\}$	$\{\exists x \neg A(x) \mid A \text{ atomic}\}$
$\{\forall x (A_1 \wedge \ldots \wedge A_n \rightarrow B) \mid A_i, B \text{ atomic}\}$	$\{\exists x (A_1 \wedge \ldots \wedge A_n \wedge \neg B) \mid A_i, B \text{ atomic}\}$
$\{\forall x \bigvee L_i \mid L_i \text{ literals}\}$	$\{\exists x \bigwedge L'_i \mid L'_i \text{ literals}\}$
$\{\forall x \phi(x) \mid \phi(x) \text{ unquantified}\}$	$\{\exists x \phi'(x) \mid \phi'(x) \text{ unquantified}\}$

validity problem for universal formulae — ground satisfiability problem

$\mathcal{T}$-validity vs. $\mathcal{T}$-satisfiability

Common restrictions on $\mathcal{L}$ / $\neg\mathcal{L}$

$\mathcal{L}$	$\neg\mathcal{L}$
$\{\forall x A(x) \mid A \text{ atomic}\}$	$\{\exists x \neg A(x) \mid A \text{ atomic}\}$
$\{\forall x (A_1 \wedge \ldots \wedge A_n \rightarrow B) \mid A_i, B \text{ atomic}\}$	$\{\exists x (A_1 \wedge \ldots \wedge A_n \wedge \neg B) \mid A_i, B \text{ atomic}\}$
$\{\forall x \bigvee L_i \mid L_i \text{ literals}\}$	$\{\exists x \bigwedge L'_i \mid L'_i \text{ literals}\}$
$\{\forall x \phi(x) \mid \phi(x) \text{ unquantified}\}$	$\{\exists x \phi'(x) \mid \phi'(x) \text{ unquantified}\}$

validity problem for universal formulae — ground satisfiability problem

In what follows we will focus on the problem of checking the satisfiability of conjunctions of ground literals

$\mathcal{T}$-validity vs. $\mathcal{T}$-satisfiability

$\mathcal{T} \models \forall x A(x)$	iff	$\mathcal{T} \cup \exists x \neg A(x)$ unsatisfiable
$\mathcal{T} \models \forall x (A_1 \wedge \cdots \wedge A_n \rightarrow B)$	iff	$\mathcal{T} \cup \exists x (A_1 \wedge \cdots \wedge A_n \wedge \neg B)$ unsatisfiable
$\mathcal{T} \models \forall x (\bigvee_{i=1}^{n} A_i \vee \bigvee_{j=1}^{m} \neg B_j)$	iff	$\mathcal{T} \cup \exists x (\neg A_1 \wedge \cdots \wedge \neg A_n \wedge B_1 \wedge \cdots \wedge B_m)$ unsatisfiable

$\mathcal{T}$-satisfiability vs. Constraint Solving

The field of Constraint Solving also deals with satisfiability problems

But be careful:

- in Constraint Solving one is interested if a formula is satisfiable in a given, fixed model of $\mathcal{T}$.
- in $\mathcal{T}$-satisfiability one is interested if a formula is satisfiable in any model of $\mathcal{T}$ at all.

3.3. Theory of Uninterpreted Function Symbols

Why?

- Reasoning about equalities is important in automated reasoning
- Applications to program verification
  (approximation: abstract from additional properties)

Application: Compiler Validation

Example: prove equivalence of source and target program

```
1:  y := 1
2:  if z = x*x*x
3:    then y := x*x + y
4:  endif
```

```
1: y := 1
2: R1 := x*x
3: R2 := R1*x
4: jmpNE(z,R2,6)
5: y := R1+1
```

To prove: (indexes refer to values at line numbers)

$$y_1 \approx 1 \wedge [(z_0 \approx x_0 * x_0 * x_0 \wedge y_3 \approx x_0 * x_0 + y_1) \vee (z_0 \not\approx x_0 * x_0 * x_0 \wedge y_3 \approx y_1)] \wedge$$

$$y_1' \approx 1 \wedge R1_2 \approx x_0' * x_0' \wedge R2_3 \approx R1_2 * x_0' \wedge$$

$$\wedge [(z_0' \approx R2_3 \wedge y_5' \approx R1_2 + 1) \vee (z_0' \neq R2_3 \wedge y_5' \approx y_1')] \wedge$$

$$x_0 \approx x_0' \wedge y_0 \approx y_0' \wedge z_0 \approx z_0' \implies x_0 \approx x_0' \wedge y_3 \approx y_5' \wedge z_0 \approx z_0'$$

Possibilities for checking it

(1) **Abstraction.**
Consider $*$ to be a "free" function symbol (forget its properties). Test it property can be proved in this approximation. If so, then we know that implication holds also under the normal interpretation of $*$.

(2) **Reasoning about formulae in fragments of arithmetic.**

Uninterpreted function symbols

Let $\Sigma = (\Omega, \Pi)$ be arbitrary

Let $\mathcal{M} = \Sigma$-alg be the class of all Σ-structures

The theory of uninterpreted function symbols is Th(Σ-alg) the family of all first-order formulae which are true in all Σ-algebras.

in general undecidable

Decidable fragment:

e.g. the class $\mathrm{Th}_\forall(\Sigma\text{-alg})$ of all universal formulae which are true in all Σ-algebras.

Uninterpreted function symbols

Assume $\Pi = \emptyset$ (and $\approx$ is the only predicate)

In this case we denote the theory of uninterpreted function symbols by $UIF(\Sigma)$ (or UIF when the signature is clear from the context).

This theory is sometimes called the theory of free functions and denoted Free(Σ)

Uninterpreted function symbols

Theorem 3.3.1

The following are equivalent:

(1) testing validity of universal formulae w.r.t. UIF is decidable

(2) testing validity of (universally quantified) clauses w.r.t. UIF is decidable

Proof: Follows from the fact that any universal formula is equivalent to a conjunction of (universally quantified) clauses.

Solution 1

Task:

Check if $UIF \models \forall \overline{x}(s_1(\overline{x}) \approx t_1(\overline{x}) \wedge \cdots \wedge s_k(\overline{x}) \approx t_k(\overline{x}) \rightarrow \bigvee_{j=1}^{m} s'_j(\overline{x}) \approx t'_j t(\overline{x}))$

Solution 1:

The following are equivalent:

(1) $(\bigwedge_i s_i \approx t_i) \rightarrow \bigvee_j s'_j \approx t'_j$ is valid

(2) $Eq(\sim) \wedge \text{Con}(f) \wedge (\bigwedge_i s_i \sim t_i) \wedge (\bigwedge_j s'_j \not\sim t'_j)$ is unsatisfiable.

where $Eq(\sim)$: $\text{Refl}(\sim) \wedge \text{Sim}(\sim) \wedge \text{Trans}(\sim)$

$\text{Con}(f) : \forall x_1, \dots, x_n, y_1, \dots, y_n(\bigwedge x_i \sim y_i \rightarrow f(x_1, \dots, x_n) \sim f(y_1, \dots, y_n))$

Resolution: inferences between transitivity axioms – nontermination

Solution 2

Task:

Check if $UIF \models \forall \overline{x}(s_1(\overline{x}) \approx t_1(\overline{x}) \wedge \cdots \wedge s_k(\overline{x}) \approx t_k(\overline{x}) \rightarrow \bigvee_{j=1}^{m} s'_j(\overline{x}) \approx t'_j(\overline{x}))$

Solution 2: Ackermann's reduction.

Flatten the formula (replace, bottom-up, $f(c)$ with a new constant c_f

$\phi \mapsto FLAT(\phi)$

Theorem 3.3.2: The following are equivalent:

(1) $(\bigwedge_i s_i(\overline{c}) \approx t_i(\overline{c})) \wedge \bigwedge_j s'_j(\overline{c}) \not\approx t'_j(\overline{c})$ is satisfiable

(2) $FC \wedge FLAT[(\bigwedge_i s_i(\overline{c}) \approx t_i(\overline{c})) \wedge \bigwedge_j s'_j(\overline{c}) \not\approx t'_j(\overline{c})]$ is satisfiable

where $FC = \{c_1 \approx d_1, \dots c_n \approx d_n \rightarrow c_f \approx d_f \mid$ whenever $f(c_1, \dots, c_n)$ was renamed to c_f
$f(d_1, \dots, d_n)$ was renamed to $d_f\}$

Note: The problem is decidable in PTIME (see next pages)

Problem: Naive handling of transitivity/congruence axiom $\mapsto O(n^3)$

Goal: Give a faster algorithm

Example

The following are equivalent:

(1) $C := f(a,b) \approx a \wedge f(f(a,b),b) \not\approx a$ is satisfiable

(2) $FC \wedge FLAT[C]$ is satisfiable, where:

$FLAT[f(a,b) \approx a \wedge f(f(a,b),b) \not\approx a]$ is computed by introducing new constants renaming terms starting with f and then replacing in C the terms with the constants:

- $FLAT[\underbrace{f(a,b)}_{a_1} \approx a \wedge \underbrace{f(\underbrace{f(a,b)}_{a_1},b)}_{a_2} \not\approx a] := a_1 \approx a \wedge a_2 \not\approx a$
- $FC := (a \approx a_1 \rightarrow a_1 \approx a_2)$

$f(a,b) = a_1$

$f(a_1,b) = a_2$

Thus, the following are equivalent:

(1) $C := f(a,b) \approx a \wedge f(f(a,b),b) \not\approx a$ is satisfiable

(2) $\underbrace{(a \approx a_1 \rightarrow a_1 \approx a_2)}_{FC} \wedge \underbrace{a_1 \approx a \wedge a_2 \not\approx a}_{FLAT[C]}$ is satisfiable

Solution 3

Task:

Check if $UIF \models \forall \overline{x}(s_1(\overline{x}) \approx t_1(\overline{x}) \wedge \cdots \wedge s_k(\overline{x}) \approx t_k(\overline{x}) \rightarrow \bigvee_{j=1}^{m} s'_j(\overline{x}) \approx t'_j(\overline{x}))$

i.e. if $(s_1(\overline{c}) \approx t_1(\overline{c}) \wedge \cdots \wedge s_k(\overline{c}) \approx t_k(\overline{c}) \wedge \bigwedge_j s'_j(\overline{c}) \not\approx t'_j(\overline{c}))$ unsatisfiable.

Solution 3

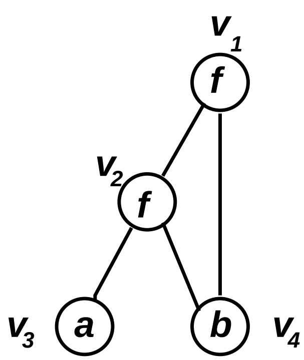
Task:

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

Solution 3 [Downey-Sethi, Tarjan'76; Nelson-Oppen'80]

represent the terms occurring in the problem as DAG's

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



$$
\begin{array}{ll}
v_1: & f(f(a,b),b) \\
v_2: & f(a,b) \\
v_3: & a \\
v_4: & b
\end{array}
$$

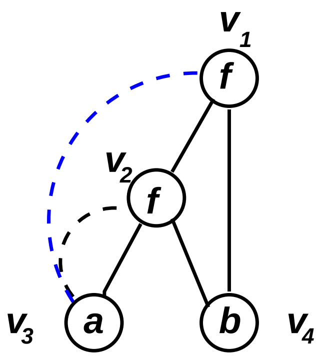
Solution 3

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 3 [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

Example

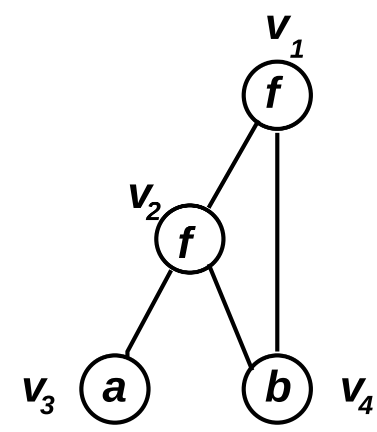
- **DAG structures:**

  - $G = (V, E)$ 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)



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

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

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

...

Congruence closure of a DAG/Relation

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

$R \subseteq V \times V$

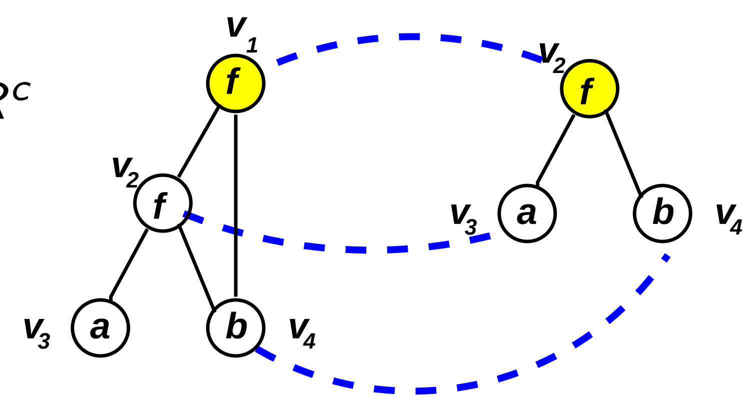
The congruence closure of R is the smallest relation R^c on V which is:

- 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 $\mapsto c_{\mathcal{A}} = [v_c]_{R^c}$.
- $f/n \mapsto f_{\mathcal{A}}([v_1]_{R^c}, \dots, [v_n]_{R^c}) = \begin{cases} [v_{f(t_1,\dots,t_n)}]_{R^c} & \text{if } v_{f(t_1,\dots,t_n)} \in V, \\ & [v_{t_i}]_{R^c} = [v_i]_{R^c} \text{ for } 1 \leq i \leq n \\ 0 & \text{otherwise} \end{cases}$

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

We will show how to algorithmically determine R^c next time.

Computing the congruence closure of a DAG

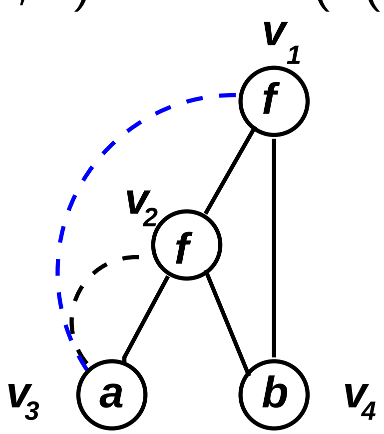
Given: $G = (V, E)$ DAG + labelling

$R \subseteq V \times V$

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

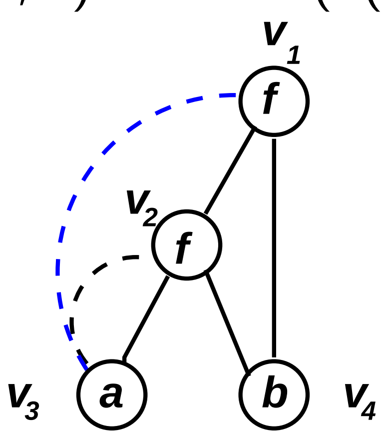
Given: $G = (V, E)$ DAG + labelling

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

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

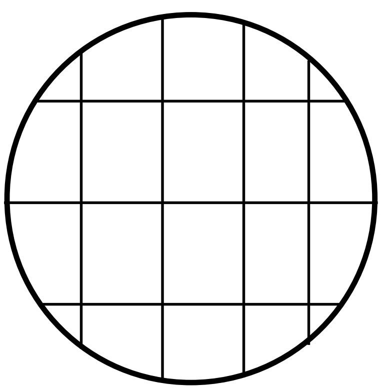
Given: $G = (V, E)$ DAG + labelling

$R \subseteq V \times V$

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

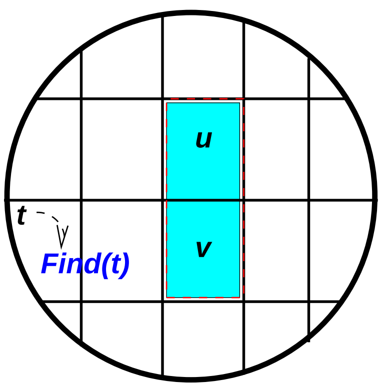
Given: $G = (V, E)$ DAG + labelling

$R \subseteq V \times V$

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 t_v

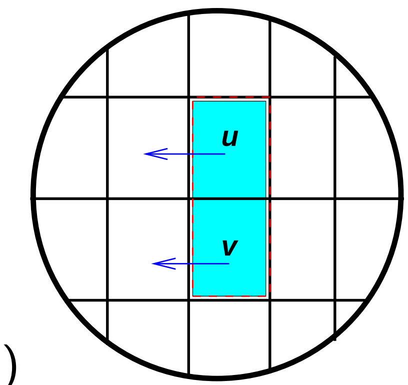
Computing the congruence closure of a DAG

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

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

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 $\{(v(s_1), v(t_1)), \ldots, (v(s_n), v(t_n))\}$]

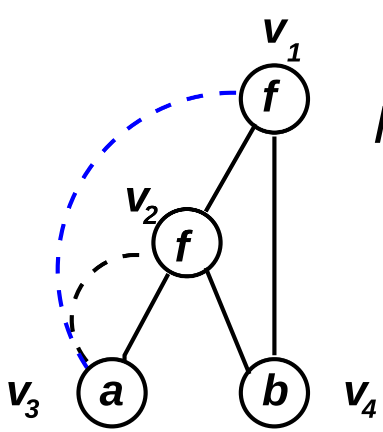
   For $i \in \{1, \ldots, n\}$ **do** $R_i :=$ MERGE(v_{s_i}, v_{t_i}) w.r.t. R_{i-1}
3. **If** FIND$(v_{s'_j})$ = FIND$(v_{t'_j})$ for some $j \in \{1, \ldots, m\}$ **then** return unsatisfiable
4. **else** [if FIND$(v_{s'_j}) \neq$ FIND$(v_{t'_j})$ for all $j \in \{1, \ldots, m\}$] **then** return satisfiable

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 $R_1 :=$ MERGE$((v_2, v_3)$

   [Test representatives]

   FIND$(v_2) = v_2 \neq v_3 =$ FIND(v_3)

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

   [Merge congruence classes]

   UNION(v_2, v_3): sets FIND(v_2) to v_3.

   [Compute and recursively merge predecessors]

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

   CONGR(v_1, v_2)

   MERGE(v_1, v_2): (different representatives)

   calls UNION(v_1, v_2) which

   sets FIND(v_1) to v_3.
3. Test whether FIND$(v_1) =$ FIND(v_3). Yes.

   Return unsatisfiable.