Decision Procedures for Verification

Decision Procedures (2)

9.01.2017

Viorica Sofronie-Stokkermans

sofronie@uni-koblenz.de

Exam

Possibilities: To be discussed during the class

Doodle (in the next days)

Until now:

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

Semantics

Structures (also many-sorted)

Models, Validity, and Satisfiability

Entailment and Equivalence

Theories (Syntactic vs. Semantics view)

Algorithmic Problems: Check satisfiability

Until now:

Normal Forms

Herbrand Models

Resolution

– Soundness, refutational completeness, refinements

– Consequences: Compactness of FOL; The Löwenheim-Skolem Theorem; Craig interpolation

Decidable subclasses of FOL

The Bernays-Schönfinkel class

(definition; decidability;tractable fragment: Horn clauses)

The Ackermann class

Decision procedures: generalities

Today

Theory of Uninterpreted Function Symbols:

Congruence closure

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)

$$
\begin{aligned}
& y_1 \approx 1 \wedge [(z_0 \approx x_0 * x_0 * x_0 \wedge y_3 \approx x_0 * x_0 + y_1) \vee (z_0 \not\approx x_0 * x_0 * x_0 \wedge y_3 \approx y_1)] \wedge \\
& y'_1 \approx 1 \wedge R1_2 \approx x'_0 * x'_0 \wedge R2_3 \approx R1_2 * x'_0 \wedge \\
& \qquad \wedge [(z'_0 \approx R2_3 \wedge y'_5 \approx R1_2 + 1) \vee (z'_0 \neq R2_3 \wedge y'_5 \approx y'_1)] \wedge \\
& x_0 \approx x'_0 \wedge y_0 \approx y'_0 \wedge z_0 \approx z'_0 \quad \Longrightarrow \quad x_0 \approx x'_0 \wedge y_3 \approx y'_5 \wedge z_0 \approx z'_0
\end{aligned}
$$

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 $\mathrm{Free}(\Sigma)$

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

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

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

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

Resolution: inferences between transitivity axioms – nontermination

Solution 2

Task:

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

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:

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

$$f(a, b) = a_1$$
$$f(a_1, b) = a_2$$

- $FC := (a \approx a_1 \rightarrow a_1 \approx a_2)$

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

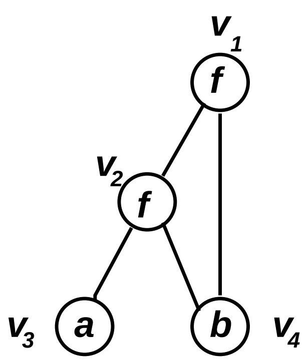
Task:

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



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

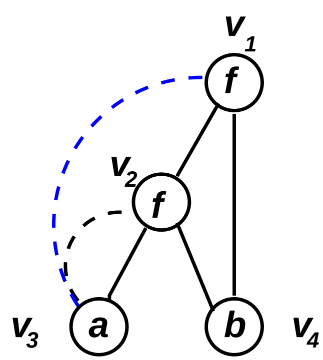
Solution 3

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



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

- 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

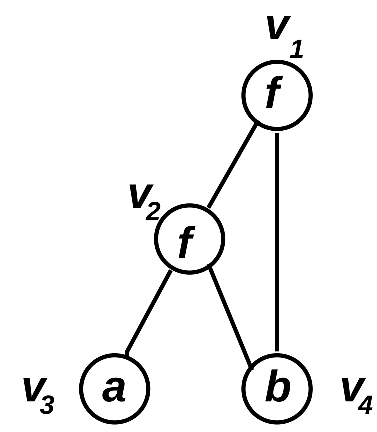
- **DAG structures:**

  - $G = (V, E)$ 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

$R \subseteq V \times V$

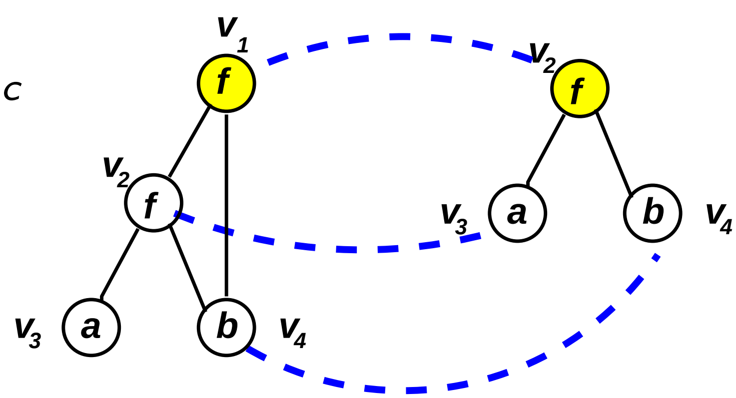
The congruence closure of R is the smallest relation R^c on V which is:

- 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 $R = \{(v_{s_i}, v_{t_i}) \mid i \in \{1, \ldots, n\}\}$

  (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

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