

G22.3033-003 Logic and Verification

Spring 2004

Lecture 6

1

Outline

- Union/Find in OCaml
- Review of Homomorphisms
- Equality
- Congruence Closure

Sources:

Harrison, John. *Introduction to Logic and Automated Theorem Proving*. Unpublished manuscript. Used by permission.

G. Nelson and D. Oppen. *Fast Decision Procedures Based on Congruence Closure*. JACM 27(2), 1980, pp. 356-364.

P. Downey, R. Sethi, and R. Tarjan. *Variations on the Common Subexpression Problem*. JACM 27(4), 1980, pp. 758-771.

3

Review

- Using Unification
- Tableaux
- Resolution

2

Union/Find

union and *find* are abstract operations for manipulating equivalence classes.

union(x, y) merges the equivalence classes of x and y .

find(x) returns a unique representative of the equivalence class of x .

In Harrison's OCaml library, these are implemented with partial functions in *lib.ml*.

union is called *equate*.

find is called *canonize*.

We will look at an example in *union_find.ml*.

4

Homomorphisms

Suppose that \mathcal{A} and \mathcal{B} are models over the same signature Σ .

A (strong) *homomorphism* h of \mathcal{A} into \mathcal{B} is a function $h : \text{dom}(\mathcal{A}) \rightarrow \text{dom}(\mathcal{B})$ such that

1. For each n -ary predicate symbol $P \in \Sigma$ and each n -tuple $\langle a_1, \dots, a_n \rangle$ of elements of $\text{dom}(\mathcal{A})$,

$$\langle a_1, \dots, a_n \rangle \in P^{\mathcal{A}} \text{ iff } \langle h(a_1), \dots, h(a_n) \rangle \in P^{\mathcal{B}}.$$

2. For each n -ary function symbol $f \in \Sigma$ and each n -tuple $\langle a_1, \dots, a_n \rangle$ of elements of $\text{dom}(\mathcal{A})$,

$$h(f^{\mathcal{A}}(a_1, \dots, a_n)) = f^{\mathcal{B}}(h(a_1), \dots, h(a_n)).$$

3. For each constant symbol $c \in \Sigma$, $h(c^{\mathcal{A}}) = c^{\mathcal{B}}$.

5

Equality

Up to now, we have treated all function and predicate symbols as *uninterpreted*, meaning that a model is free to give any interpretation to these symbols.

Now we will start considering what happens if we restrict the meaning of certain symbols.

The most obvious example is the equality predicate, $=$.

We say that a model M is *normal* if the equality symbol $=$ is interpreted as equality on the domain of M .

We can make use of the general first order methods described thus far if we can somehow restrict them to consider only normal models.

How can we do this?

7

Homomorphism Theorem

Let h be a strong homomorphism from \mathcal{A} to \mathcal{B} , and let s map the set of variables into $\text{dom}(\mathcal{A})$.

1. For any term t , $h(\bar{s}(t)) = \overline{h \circ s}(t)$, where \bar{s} is computed in \mathcal{A} , and $\overline{h \circ s}(t)$ is computed in \mathcal{B} .

2. For any quantifier-free formula α

$$\mathcal{A} \models_s \alpha \text{ iff } \mathcal{B} \models_{h \circ s} \alpha.$$

3. If h is surjective (onto), then the above holds even if α contains quantifiers.

6

Equality

Recall that a class \mathcal{K} of models is *first-order axiomatizable* (EC_{Δ}), iff $\mathcal{K} = \text{Mod}T$ for some set of sentences T .

Is the class of normal models first-order axiomatizable?

Theorem

The class of normal models is not first-order axiomatizable.

Proof

Suppose M is a normal model. Let a be an element of the domain of M . Consider adding an element b to the domain of M and defining a function $h : \text{dom}(M) \cup \{b\} \rightarrow \text{dom}(M)$ which is the identity function except at b which it maps to a . It is not hard to see that we can construct an extension M' of M with domain $\text{dom}(M) \cup \{b\}$ in such a way that h is a homomorphism of M' onto M .

By the homomorphism theorem, for any sentence ϕ , $M \models \phi$ iff $M' \models \phi$. Thus, for any set of sentences T , $M \in \text{Mod}T$ iff $M' \in \text{Mod}T$. But M is a normal model and M' is not. \square

8-a

Equality Axioms

So it is impossible to exactly define the class of normal models using first order axioms.

However, all hope is not lost. We can specify a set of axioms such that the questions of validity and satisfiability can be answered with the existing apparatus.

Let $eqaxioms(\Delta)$ be defined as the following set of sentences:

1. $\forall x. x = x$
2. $\forall x y. x = y \leftrightarrow y = x$
3. $\forall x y z. x = y \wedge y = z \rightarrow x = z$
4. $\forall x_1 \cdots x_n y_1 \cdots y_n. x_1 = y_1 \wedge \cdots \wedge x_n = y_n \rightarrow f(x_1, \dots, x_n) = f(y_1, \dots, y_n)$, for each function symbol f in Δ
5. $\forall x_1 \cdots x_n y_1 \cdots y_n. x_1 = y_1 \wedge \cdots \wedge x_n = y_n \rightarrow R(x_1, \dots, x_n) \rightarrow R(y_1, \dots, y_n)$, for each predicate symbol R in Δ

9

Equality Axioms

Corollary

Any formula p is satisfiable in a normal model iff $p \wedge eqaxioms(p)$ is satisfiable.

Corollary

A formula p is valid in normal models iff $eqaxioms(p) \rightarrow p$ is valid.

Thus by generating the appropriate equality axioms, we can reduce the question of satisfiability or validity over normal models to the question of general satisfiability or validity.

11

Equality Axioms

Theorem

Any set Δ of first order formulas has a normal model iff the set $\Delta \cup eqaxioms(\Delta)$ has a model.

Proof

The only if direction is easy since clearly any normal model satisfies the equality axioms.

In the other direction, suppose M is a model of $\Delta \cup eqaxioms(\Delta)$. Let \sim be the relation $dom(M)$ which interprets the equality symbol in M . Because M satisfies the equality axioms, \sim is an equivalence relation. Now let h be a function which maps each element of M to its equivalence class. It is not hard to see that we can construct a normal model M' whose domain is the equivalence classes of \sim in such a way that h is a homomorphism of M onto M' . It follows by the homomorphism theorem that M' is a model of Δ . \square

10

Equality Axioms

Notice that $eqaxioms(p)$ includes a congruence axiom for the equality symbol itself:

$$\forall x_1 x_2 y_1 y_2. x_1 = y_1 \wedge x_2 = y_2 \rightarrow x_1 = x_2 \rightarrow y_1 = y_2.$$

This axiom implies symmetry and transitivity.

It is also implied by equality, symmetry and transitivity, so there is some redundancy in our set of axioms.

We can eliminate this by omitting the congruence axiom for equality and using only reflexivity and the following modified version of transitivity (which together with reflexivity implies symmetry):

$$\forall x y z. x = y \wedge x = z \rightarrow y = z$$

The code for adding equality axioms is in [equal.ml](#).

12

Congruence Closure

Let $G = (V, E)$ be a directed graph such that for each vertex v in G , the successors of v are ordered.

Let C be any equivalence relation on V .

The *congruence closure* C^* of C is the finest equivalence relation on V that contains C and satisfies the following property for all vertices v and w .

Let v and w have successors v_1, \dots, v_k and w_1, \dots, w_l respectively.
If $k = l$ and $(v_i, w_i) \in C^*$ for $1 \leq i \leq k$, then $(v, w) \in C^*$.

In other words, if the corresponding successors of v and w are equivalent under C^* , then v and w are themselves equivalent under C^* .

Often, the vertices are labeled by some labeling function λ . In this case, the property becomes:

If $\lambda(v) = \lambda(w)$ and if $k = l$ and $(v_i, w_i) \in C^*$ for $1 \leq i \leq k$, then $(v, w) \in C^*$.

13

Congruence Closure and $T_{\mathcal{E}}$

Recall that $T_{\mathcal{E}}$ is the empty theory with equality over some signature Σ containing only function symbols.

If Γ is a set of ground Σ -equalities and Δ is a set of ground Σ -disequalities, then the satisfiability of $\Gamma \cup \Delta$ can be determined as follows.

- Let G be a graph which corresponds to the abstract syntax trees of terms in $\Gamma \cup \Delta$, and let v_t denote the vertex of G associated with the term t .
- Let C be the equivalence relation on the vertices of G induced by Γ .
- $\Gamma \cup \Delta$ is satisfiable iff for each $s \neq t \in \Delta$, $(v_s, v_t) \notin C^*$.

15

A Simple Algorithm

Let $C_0 = C$ and $i = 0$.

1. Number the equivalence classes in C_i consecutively from 1.
2. Let α assign to each vertex v the number $\alpha(v)$ of the equivalence class containing v .
3. For each vertex v construct a *signature* $s(v) = \lambda(v)(\alpha(v_1), \dots, \alpha(v_k))$, where v_1, \dots, v_k are the successors of v .
4. Group the vertices into classes of vertices having equal signatures.
5. Let C_{i+1} be the finest equivalence relation on V such that two vertices equivalent under C_i or having the same signature are equivalent under C_{i+1} .
6. If $C_{i+1} = C_i$, let $C^* = C_i$; otherwise increment i and repeat.

14

An Algorithm for $T_{\mathcal{E}}$

$cc(\Gamma, \Delta)$

Construct $G(V, E)$ from terms in Γ and Δ .

while $\Gamma \neq \emptyset$

 Remove some equality $a = b$ from Γ ;

 Merge(a, b);

if $find(a) = find(b)$ for some $a \neq b \in \Delta$ **then**

return false;

return true;

16

An Algorithm for $T_{\mathcal{E}}$

Merge(a, b)

if *find(a) = find(b)* then return;

Let *A* be the set of all predecessors
of all vertices equivalent to *a*;

Let *B* be the set of all predecessors
of all vertices equivalent to *b*;

union(a, b);

foreach *x* ∈ *A* and *y* ∈ *B*

if *signature(x) = signature(y)* then *Merge(x, y)*;

The OCaml code is in *cong.ml*.

17

DST Algorithm

cc(Γ, Δ)

Construct $G(V, E)$ from terms in Γ and Δ .

Merge(Γ);

if *find(a) = find(b)* for some $a \neq b \in \Delta$ then

return *false*;

return *true*;

19

Congruence Closure

DST Algorithm

The Downey-Sethi-Tarjan Congruence Closure algorithm is more efficient. It makes use of some additional data structures and methods.

Additional Helper Methods

- *union(a, b)* in this algorithm, the *first* argument always becomes the new equivalence class representative.
- *list(e)* returns the list of vertices with at least one successor in equivalence class *e*.
- *enter(v)* stores $(v, \text{signature}(v))$ in a signature table.
- *delete(v)* removes $(v, \text{signature}(v))$ from the signature table if it is there. Note that this operation does *not* remove any other entry, even if it has the same signature as *v*.
- *query(v)* if there is an entry $(w, \text{signature}(w))$ in the signature table, and $\text{signature}(w) = \text{signature}(v)$, then return *w*; otherwise, return \perp .

18

DST Algorithm

Merge(combine)

pending := set of all vertices;

while *pending* ≠ ∅

foreach *v* ∈ *pending*

if *query(v) = ⊥* then *enter(v)*;

else add $(v, \text{query}(v))$ to *combine*;

pending := ∅;

foreach $(a, b) \in \text{combine}$

if *find(a) ≠ find(b)* then

if $|\text{list}(\text{find}(a))| < |\text{list}(\text{find}(b))|$ then swap *a* and *b*;

foreach *u* ∈ *list(find(b))*

delete(u); add *u* to *pending*;

union(find(a), find(b));

combine := ∅;

20