22c:295 Seminar in AI — Decision Procedures

Equality

Cesare Tinelli

tinelli@cs.uiowa.edu

The University of Iowa

[a] These notes were developed from a set of lecture notes originally written by Clark Barrett at New York University. These notes are copyrighted material and may not be used in other course settings outside of the University of Iowa in their current form or modified form without the express written permission of the copyright holders.

Outline

- First Order Logic with and without Equality
- 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 Unpublished manuscript.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.

FOL with/without equality

So far we have considered First Order Logic *with equality*.

In this logic, the equality symbol (=) is a logical constant and every model of the logic interprets it as the identity relation.

First Order Logic *without equality* is a weaker version of FOL that has no distinguished equality symbol.

This logic, admits models that interpret the equality symbol as a relation other than the identity relation.

From now on, we will consider FOL without equality.

Homomorphisms/Embeddings

Suppose that $\mathcal{A}$ and $\mathcal{B}$ are models over the same signature Σ. A *strong homomorphism* (aka, *embedding*) h of $\mathcal{A}$ into $\mathcal{B}$ is a function $h : \mathit{dom}(\mathcal{A}) \rightarrow \mathit{dom}(\mathcal{B})$ such that

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

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

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

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

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

Embedding Theorem

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

1. For any term t, $h(\overline{s}(t)) = \overline{h \circ s}(t)$, where $\overline{s}$ is computed in $\mathcal{A}$, and $\overline{h \circ s}(t)$ is computed in $\mathcal{B}$.
2. For any quantifier-free formula α

   $\mathcal{A} \models_s \alpha$ iff $\mathcal{B} \models_{h \circ s} \alpha$.
3. If h is surjective (onto), then the above holds even if α contains quantifiers.

Capturing equality in FOL without equality

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

Can we restrict ourselves to just normal models?

More precisely, recall that a class $\mathcal{K}$ of Σ-models is *elementary* (EC_Σ), iff $\mathcal{K} = \mathit{Mod}\, T$ for some set T of first order Σ-sentences.

Question: Is the class of normal (Σ-)models elementary?

Capturing Equality in FOL without equality

Theorem The class of normal models is not elementary.

Proof Sketch

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 $h : \mathit{dom}(M) \cup \{b\} \to \mathit{dom}(M)$ 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 $\mathit{dom}(M) \cup \{b\}$ in such a way that h is a *surjective* embedding of M' onto M.

By the Embedding Theorem, for any sentence φ, $M \models \varphi$ iff $M' \models \varphi$. Thus, for any set of sentences T, $M \in \mathit{Mod}\, T$ iff $M' \in \mathit{Mod}\, T$. But M is a normal model and M' is not. $\square$

Capturing Equality in FOL without equality

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 using which the questions of validity and satisfiability can be answered with the existing apparatus.

Equality Axioms

Let Δ be a set of formulas and let *eqaxioms* (Δ) 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. $\forall x_1 \cdots x_n\, y_1 \cdots y_n.\, x_1 = y_1 \wedge \cdots \wedge x_n = y_n \rightarrow f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$,

   for each function symbol f occurring in Δ
5. $\forall x_1 \cdots x_n\, y_1 \cdots y_n.\, x_1 = y_1 \wedge \cdots \wedge x_n = y_n \rightarrow R(x_1, \ldots, x_n) \rightarrow R(y_1, \ldots, y_n)$,

   for each predicate symbol R occurring in Δ

Equality Axioms

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

Proof Sketch

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 $\Delta \cup \mathit{eqaxioms}(\Delta)$. Let $\sim$ be the relation on $\mathit{dom}(M)$ 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 surjective embedding of M onto M'. It follows by the Embedding Theorem that M' is a model of Δ. $\square$

Equality Axioms

Corollary Any formula φ is satisfiable in a normal model iff $\varphi \wedge \mathit{eqaxioms}(\varphi)$ is satisfiable.

Corollary A formula φ is valid in normal models iff $\mathit{eqaxioms}(\varphi) \rightarrow \varphi$ 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.

Satisfi ability and Validity Modulo the Theory of Equality

By the above results, the problem of checking satisfiability/validity in normal models can be reduced to a satisfiability/validity modulo theories problem.

Let Σ be a finite signature and let $T_{\mathcal{E}}$ be the set of equality axioms over the symbols in Σ.

Corollary Any Σ-formula φ is satisfiable in a normal model iff φ is $T_{\mathcal{E}}$-satisfiable.

Corollary A Σ-formula φ is valid in normal models iff φ is $T_{\mathcal{E}}$-valid.

Satisfi ability and Validity Modulo the Theory of Equality

Theorem The $T_{\mathcal{E}}$-satisfiability of *quantifier-free* formulas is decidable.

We will study decision procedures for this problem in the following.

Observation: For any theory T, a quantifier-free formula φ with free variables $x_1, \ldots, x_n$ is T-unsatisfiable iff $\forall x_1 \cdots \forall x_n \neg\varphi$ is T-valid.
Therefore, the T-unsatisfiability of quantifier-free formulas and the T-validity of universal formulas are equivalent problems for any theory T.

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

1. contains C and
2. satisfies the following property for all vertices v and w with respective successors $v_1, \ldots, v_k$ and $w_1, \ldots, w_l$:

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

Congruence Closure

The essence of congruence closure:

> If the corresponding successors of the nodes 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 $\lambda(v) = \lambda(w)$, $k = l$, and $(v_i, w_i) \in C^*$ for $1 \leq i \leq k$,
> then $(v, w) \in C^*$.

A Simple Algorithm for Congruence Closure

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* $s(v) = \lambda(v)(\alpha(v_1), \ldots, \alpha(v_k))$, where $v_1, \ldots, v_k$ are the successors of v.
4. 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}.
5. If $C_{i+1} = C_i$, let $C^* = C_i$; otherwise increment i and repeat.

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

Let Σ be a signature with no predicate symbols and let $T_{\mathcal{E}}$ be the theory of equality over Σ.

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

- Let G be a graph corresponding to the abstract syntax trees of terms in $\Gamma \cup \Delta$.
- 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 $T_{\mathcal{E}}$-satisfiable iff for each $s \neq t \in \Delta$, $(v_s, v_t) \notin C^*$.

An Algorithm for $T_{\mathcal{E}}$-satisfi ability

Input:
Γ, a (finite) set of Σ-equalities
Δ, a (finite) set of Σ-disequalities

```
cc(Γ, Δ)
    Construct G(V, E) from terms in Γ and Δ;
    while Γ ≠ ∅
        Remove some equality a = b from Γ;
        Merge(a, b);
    if find(a) = find(b) for some a ≠ b ∈ Δ then
        return false
    else
        return true
```

An Algorithm for $T_{\mathcal{E}}$-satisfi ability

Input: a, b, two vertices of $G(V, E)$

```
Merge(a, b)
   if find(a) = find(b) then return;
   Let A be the set of all predecessors of
    all vertices c s.t. find(c) = find(a);
   Let B be the set of all predecessors of
    all vertices c s.t. find(c) = find(b);
   union(a, b);
   foreach x ∈ A and y ∈ B
      if signature(x) = signature(y) then
         Merge(x, y)
```

Union/Find

Recall:
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 current equivalence class of x.

Correctness of CC

Proposition The CC procedure terminates on all inputs.

Proof Exercise.

Proposition Let Γ be a finite set of Σ-equalities and let Δ be a finite set of Σ-disequalities.
The set $\Gamma \cup \Delta$ is $T_{\mathcal{E}}$-satisfiable iff $CC(\Gamma, \Delta) = \mathit{true}$.

Proof Non-trivial. See references on page 2.

Decidability results

Theorem The $T_{\mathcal{E}}$-satisfiability of quantifier-free formulas with no predicate symbols is decidable.

Proof

By the usual DNF translation, every quantifier-free formula with no predicate symbols can be effectively converted into an equisatisfiable disjunction of sets of equalities and disequalities. Then the two results above apply. □

Corollary The $T_{\mathcal{E}}$-satisfiability of quantifier-free formulas is decidable.

Proof

Treat the predicate symbols of the given formula as function symbols by replacing each atom $p(t_1, \ldots, t_n)$ with the equality $p(t_1, \ldots, t_n) = \mathbf{T}$, where $\mathbf{T}$ is a fresh constant symbol. Then the previous theorem applies. □

Complexity of Congruence Closure

The previous *CC* algorithm has time complexity $O(n^2)$ where n is the number of vertices in the initial graph.

DST Algorithm The Downey-Sethi-Tarjan Congruence Closure algorithm is more efficient, with complexity $O(n \log n)$.

The lower complexity is achieved thanks to additional data structures and methods.

DST Algorithm

Additional Helper Functions

- *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 $(v, \textit{signature}(v))$ in a signature table.
- *delete*(v) removes $(v, \textit{signature}(v))$ 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 $(w, \textit{signature}(w))$ in the signature table with $w \neq v$ and $\textit{signature}(w) = \textit{signature}(v)$, then return w; otherwise, return $\perp$.

DST Algorithm

Γ, a (finite) set of Σ-equalities
Δ, a (finite) set of Σ-disequalities

```
cc(Γ, Δ)
    Construct G(V, E) from terms in Γ and Δ;
    C := {(a, b) | a = b ∈ Γ};
    Merge(C);
    if find(a) = find(b) for some a ≠ b ∈ Δ then
        return false
    else
        return true
```

DST Algorithm

```
Merge(C)
  pending := set of all vertices;
  while pending ≠ ∅
     foreach v ∈ pending
        if query(v) = ⊥ then enter(v)
        else add (v, query(v)) to C;
     pending := ∅;
     foreach (a, b) ∈ C
        if find(a) ≠ find(b) then
          if |list(find(a))| < |list(find(b))| then
             swap a and b;
          foreach u ∈ list(find(b))
             delete(u); add u to pending;
          union(find(a), find(b));
     C := ∅
```