CS 433 Automated Reasoning 2024

Lecture 11: Theory of equality and uninterpreted functions (QF_EUF)

Instructor: Ashutosh Gupta

IITB India

Compile date: 2024-02-14

Topic 11.1

Theory of equality and function symbols (EUF)

Reminder: Theory of equality and function symbols (EUF)

EUF syntax: first-order formulas with signature $\mathbf{S} = (\mathbf{F}, \emptyset)$,
i.e., countably many function symbols and no predicates.

The theory axioms include

1. $\forall x.\ x = x$
2. $\forall x, y.\ x = y \Rightarrow y = x$
3. $\forall x, y, z.\ x = y \wedge y = z \Rightarrow x = z$
4. for each $f/n \in \mathbf{F}$, $\forall x_1, .., x_n, y_1, .., y_n.\ x_1 = y_1 \wedge .. \wedge x_n = y_n \Rightarrow f(x_1, .., x_n) = f(y_1, .., y_n)$

Note: Predicates can be easily added if desired

Commentary: Since the axioms are valid in FOL with equality, the theory is sometimes referred as the base theory.

Proofs in quantifier-free fragment of $\mathcal{T}_{EUF}$(QF_EUF)

The axioms translates to the proof rules of $\mathcal{T}_{EUF}$ as follows

$$\frac{x = y}{y = x}\,Symmetry \qquad \frac{x = y \quad y = z}{x = z}\,Transitivity \qquad \frac{x_1 = y_1 \quad .. \quad x_n = y_n}{f(x_1, .., x_n) = f(y_1, .., y_n)}\,Congruence$$

Example 11.1

Consider: $y = x \wedge y = z \wedge f(x, u) \neq f(z, u)$

$$\dfrac{\dfrac{\dfrac{\dfrac{y = x}{x = y} \quad y = z}{x = z}}{f(x, u) = f(z, u)} \quad f(x, u) \neq f(z, u)}{\bot}$$

Commentary: Proof rules capture the intention of axioms. The rules are complete, i,e., they allow you to prove $F \models_{EUF} G$ for any F and G if it holds.

Exercise: equality with uninterpreted functions

Exercise 11.1

If unsat, give proof of unsatisfiability

- $f(f(c)) \neq c \wedge f(c) = c$
- $f(f(c)) = c \wedge f(c) \neq c$
- $f(f(c)) = c \wedge f(f(f(c))) \neq c$
- $f^3(a) = a \wedge f^5(a) = a \wedge f(a) \neq a$

Topic 11.2

QF_EUF solving via SAT solver

Eager solving

Explicate all the theory reasoning as Boolean clauses.

Use SAT solver alone to check satisfiability.

Only possible for the theories, where we can bound the relevant instantiations of the theory axioms.

The eager solving for QF_EUF is called Ackermann's Reduction.

Notation: term encoder

Let *en* be a function that maps terms to new constants.

We can apply *en* on a formula to obtain a formula over the fresh constants.

Example 11.2

Consider $en = \{f(x) \mapsto t_1, f(y) \mapsto t_2, x \mapsto t_3, y \mapsto t_4\}$.

$$en(x = y \Rightarrow f(x) = f(y)) = (t_3 = t_4 \Rightarrow t_1 = t_2)$$

Notation: Boolean encoder

For a formula F, let boolean encoder e be a partial map from $atoms(F)$ to fresh boolean variables.

Definition 11.1

For a formula F, let $e(F)$ denote the term obtained by replacing each atom a by $e(a)$ if $e(a)$ is defined.

Example 11.3

Consider $e = \{t_3 = t_4 \mapsto p_1, t_1 = t_2 \mapsto p_2\}$

$e(t_3 = t_4 \Rightarrow t_1 = t_2) = (p_1 \Rightarrow p_2)$

Ackermann's Reduction

The insight: the rules needed to be applied only finitely many possible ways.

Algorithm 11.1: QF_EUF_Sat(F)

Input: F formula QF_EUF

Output: SAT/UNSAT

Let Ts be subterms of F, en be $Ts \rightarrow$ fresh constants, e be a Boolean encoder;

$G := en(F)$;

foreach $f(x_1, .., x_n), f(y_1, .., y_n) \in Ts$ **do**

$\quad G := G \wedge en(x_1 = y_1 \wedge .. \wedge x_n = y_n \Rightarrow f(x_1, .., x_n) = f(y_1, .., y_n))$

foreach $t_1, t_2, t_3 \in Ts$ **do**

$\quad G := G \wedge en(t_1 = t_2 \wedge t_2 = t_3 \Rightarrow t_1 = t_3)$

foreach $t_1, t_2 \in Ts$ **do**

$\quad G := G \wedge en(t_1 = t_2 \Leftrightarrow t_2 = t_1)$

$G' := e(G)$;

return *CDCL(G')*

Exercise 11.2

Can we avoid clauses for the symmetry rule?

Example: Ackermann's Reduction

Assumed that symmetric atoms mapped to same variable.

Example 11.4

Consider formula $F = f(f(x)) \neq x \wedge f(x) = x$
$Ts := \{f(f(x)), f(x), x\}$.

$en := \{f(f(x)) \mapsto f_1, f(x) \mapsto f_2, x \mapsto f_3\}$

Boolean encoding:
$\{f_1 = f_3 \mapsto p_1, f_2 = f_3 \mapsto p_2, f_1 = f_2 \mapsto p_3\}$

$G := en(F) := f_1 \neq f_3 \wedge f_2 = f_3$

$G' := \neg p_1 \wedge p_2$

Adding congruence consequences:
$G := G \wedge (f_2 = f_3 \Rightarrow f_1 = f_2)$.

$G' := G' \wedge (p_2 \Rightarrow p_3)$.

Adding transitivity consequences:

$$G := G \wedge (f_1 = f_2 \wedge f_2 = f_3 \Rightarrow f_1 = f_3) \wedge (f_1 = f_3 \wedge f_2 = f_3 \Rightarrow f_1 = f_2) \wedge (f_1 = f_2 \wedge f_1 = f_3 \Rightarrow f_2 = f_3).$$

$$G' := G' \wedge (p_3 \wedge p_2 \Rightarrow p_1) \wedge (p_3 \wedge p_2 \Rightarrow p_1) \wedge (p_1 \wedge p_3 \Rightarrow p_2).$$

Since G' is UNSAT, F is UNSAT.

Other eager encoding

Byrant's Encoding is another method of encoding EUF formulas into a SAT problem.

Exercise 11.3

How Byrant's Encoding encoding work?

Topic 11.3

Lazy QF_EUF solver

Eager is too eager

- Eager solver wastefully instantiates too many clauses
- Eager solvers do not scale

Exercise 11.4

What is the size blow up in the Ackermann's reduction?

Lazy incremental solver

Lazy: axioms are applied on demand

Incremental: one literal is consider at a time.

Solver applies axioms only related to the literals.

Lazy solver handles only conjunction of literals. For full QF_EUF, we will integrate lazy solver with CDCL.

Algorithm 11.2: *LazyEUF*(Conjunction of EUF literals F)

```
globals: bool conflictFound := 0  // modified inside IncrEUF
foreach  t1 ⋈ t2 ∈ F do
    IncrEUF(t1 ⋈ t2);
    if conflictFound then
        return unsat;
return sat;
```

IncrEUF

General idea: maintain equivalence classes among terms

Algorithm 11.3: $IncrEUF(t_1 \bowtie t_2)$

globals: set of terms $Ts := \emptyset$, set of pairs of classes $DisEq := \emptyset$, bool $conflictFound := 0$

$Ts := Ts \cup subTerms(t_1) \cup subTerms(t_2)$;

$C_1 := getClass(t_1)$; $C_2 := getClass(t_2)$; // if t_i is seen first time, create new class

if $\bowtie\ =$ "$=$" **then**

- **if** $C_1 = C_2$ **then** **return** ;
- **if** $(C_1, C_2) \in DisEq$ **then** $\{\ conflictFound := 1$; **return**; $\}$;
- $C := mergeClasses(C_1, C_2)$; $parent(C) := (C_1, C_2, t_1 = t_2)$;
- $DisEq := DisEq[C_1 \mapsto C, C_2 \mapsto C]$

else

- $DisEq := DisEq \cup (C_1, C_2)$; // $\bowtie\ =$ "$\neq$"
- **if** $C_1 = C_2$ **then** $conflictFound := 1$; **return** ;

foreach $f(r_1, \dots, r_n), f(s_1, \dots, s_n) \in Ts \wedge \forall i \in 1..n.\ \exists C.\ r_i, s_i \in C$ **do**

- $IncrEUF(f(r_1, \dots, r_n) = f(s_1, \dots, s_n))$;

Exercise 11.5

Can we drop the condition $f(r_1, \dots, r_n), f(s_1, \dots, s_n) \in Ts$*?*

Example: push

Example 11.5

Consider input $f(f(x)) \neq x \wedge f(x) = x$

- $IncrEUF(f(f(x)) \neq x)$
  - *term set* $Ts = \{x, f(x), f(f(x))\}$
  - *classes* $C_1 = \{f(f(x))\}$, *and* $C_2 = \{x\}$
  - $DisEq = \{(C_1, C_2)\}$
- $IncrEUF(f(x) = x)$
  - *classes* $C_1 = \{f(f(x))\}$, $C_2 = \{x\}$, *and* $C_3 = \{f(x)\}$ *new classes are created on demand!*
  - $C_4 = mergeClasses(C_2, C_3)$: *classes* $C_1 = \{f(f(x))\}$, $C_4 = \{f(x), x\}$
  - $DisEq = \{(C_1, C_4)\}$
  - *Apply congruence on function* f *and terms of* C_4
    - *Triggers recursive call* $IncrEUF(f(f(x)) = f(x))$
- $IncrEUF(f(f(x)) = f(x))$
  - *Since* $(C_1, C_4) \in DisEq$, $conflictFound = 1$ *and exit*

Topic 11.4

Completeness of *IncrEUF*

Completeness is not obvious

Example 11.6

Consider: $x = y \wedge y = z \wedge f(x,u) \neq f(z,u)$

$$\dfrac{\dfrac{\dfrac{x = y}{f(x,u) = f(y,u)} \quad \dfrac{y = z}{f(y,u) = f(z,u)}}{f(x,u) = f(z,u)} \qquad f(x,u) \neq f(z,u)}{\bot}$$

In the proof $f(y,u)$ occurs, which does not occur in the input formula.

Commentary: Our algorithm only derives facts consists of terms that occur in the input. If the above proof exists, does it endanger the completeness of *IncrEUF*?

Completeness of *IncrEUF*

Theorem 11.1

Let $\Sigma = \{\ell_1, .., \ell_n\}$ be a set of literals in $\mathcal{T}_{EUF}$.
$IncrEUF(\ell_1); ...; IncrEUF(\ell_n)$; finds conflict iff Σ is unsat.

Proof.

Since *IncrEUF* uses only sound proof steps of the theory, it cannot find conflict if Σ is sat.

Assume Σ is unsat and there is a proof for it.

Since *IncrEUF* applies congruence only if the resulting terms appear in Σ, we show that there is a proof that contains only such terms. ...

Completeness of *IncrEUF*(contd.)

Proof(contd.)

Since Σ is unsat, there is $\Sigma' \cup \{s \neq t\} \subseteq \Sigma$ s.t. $\Sigma' \cup \{s \neq t\}$ is unsat and Σ' contains only positive literals.(Why?)

Consider a proof that derives $s = t$ from Σ'.

Therefore, we must have a proof step such that

$$\frac{u_1 = u_2 \quad .. \quad u_{n-1} = u_n}{s = t},$$

Flattened transitivity and symmetry rules!!

where $n \geq 2$, the premises have proofs from Σ', $u_1 = s$, and $u_n = t$. ...

Exercise 11.6

Show that the last claim holds.

Commentary: We can generalize transitivity with more than two premises. $\frac{u_1=u_2 \quad u_2=u_3 \quad \dots \quad u_{n-1}=u_n}{u_1=u_n}$
We may assume that symmetry is not used if we assume $s = t$ is same as $t = s$. We interpret them in either direction as needed.

Completeness of *IncrEUF* (contd.)

Proof(contd.)

Wlog, we assume $u_i = u_{i+1}$ either occurs in Σ' or derived from congruence.

Observation: if $u_i = u_{i+1}$ is derived from congruence then the top symbols are same in u_i and u_{i+1}.

Now we show that we can transform the proof via induction over height of congruence proof steps. ...

Exercise 11.7

Justify the "wlog" claim.

Completeness of *IncrEUF* (contd.)

Proof(contd.)

Claim: If s and t occurs in Σ', any proof of $s = t$ can be turned into a proof that contains only the terms from Σ'

Base case:
If no congruence is used to derive $s = t$ then no fresh term was invented. (Why?)

Induction step:
We need not worry about $u_i = u_{i+1}$ that are coming from Σ'.

Only in the subchains of the equalities that are derived from congruences may have new terms. ...

Example 11.7

$$\dfrac{\dfrac{x = y}{f(x,u) = f(y,u)} \qquad \dfrac{y = z}{f(y,u) = f(z,u)}}{f(x,u) = f(z,u)}$$

Completeness of *IncrEUF* (contd.)

Proof(contd.)

Let $f(u_{11},..,u_{1k}) = f(u_{21},..,u_{2k}) \quad .. \quad f(u_{(j-1)1},..,u_{(j-1)k}) = f(u_{j1},..,u_{jk})$ be such a maximal subchain in the last proof step for $s = t$.

$$\frac{s = \ldots \quad \dfrac{u_{11}=u_{21} \quad .. \quad u_{1k}=u_{2k}}{f(u_{11},..,u_{1k})=f(u_{21},..,u_{2k})} \quad .. \quad \dfrac{u_{(j-1)1}=u_{j1} \quad .. \quad u_{(j-1)k}=u_{jk}}{f(u_{(j-1)1},..,u_{(j-1)k})=f(u_{j1},..,u_{jk})} \quad \ldots = t}{s = t},$$

We know $f(u_{11},..,u_{1k})$ and $f(u_{j1},..,u_{jk})$ occur in Σ'.(Why?)

For $1 < i < j$, $f(u_{i1},..,u_{ik})$ may not occur in Σ'. ...

Exercise 11.8

Justify the (Why?). (Hint: Maximal subchain requirement ensures that either $f(u_{11},..,u_{1k})$ is s or equality before is not derived by congruence.)

Completeness of *IncrEUF* (contd.)

Proof(contd.)

We can rewrite the proof in the following form.

$$\cfrac{s = \ldots \quad \cfrac{\cfrac{u_{11}=u_{21} \quad .. \quad u_{(j-1)1}=u_{j1}}{u_{11}=u_{j1}} \quad .. \quad \cfrac{u_{1k}=u_{2k} \quad .. \quad u_{(j-1)k}=u_{jk}}{u_{1k}=u_{jk}}}{f(u_{11},..,u_{1k})=f(u_{j1},..,u_{jk})} \quad \ldots = t}{s = t}$$

Due to induction hypothesis, for each $i \in 1..k$,

since u_{1i} and u_{ji} occur in Σ', $u_{1i} = u_{ji}$ has a proof with the restriction. □

Example 11.8

$$\cfrac{\cfrac{x = y}{f(x,u) = f(y,u)} \quad \cfrac{y = z}{f(y,u) = f(z,u)}}{f(x,u) = f(z,u)} \rightsquigarrow \cfrac{\cfrac{x = y \quad y = z}{x = z}}{f(x,u) = f(z,u)}$$

Topic 11.5

Model generation

Model generation

After running *LazyEUF*, if we have no contradiction then we construct a satisfying model.

- Each equivalence class is mapped to a value from the universe of model.
- We may assign a value to multiple classes while respecting disequality constraints
  - The problem of finding optimum model reduces into graph coloring problem. (How?)
- The models of functions are read from the class value map and their term parent relation.

Example: model generation

Example 11.9

Consider formula $f(f(a)) = a \wedge f(a) \neq a$.

We have terms $Ts = \{f^2(a), f(a), a\}$.

Due to the constraint, we have classes $C_1 = \{f^2(a), a\}$ *and* $C_2 = \{f(a)\}$.

Since C_1 *and* C_2 *can not be merged, we assign values* v_1 *and* v_2 *respectively.*

Therefore, we construct model m *as follows*

- $D_m = \{v_1, v_2\}$
- $a = v_1$
- $f = \{v_1 \mapsto v_2, v_2 \mapsto v_1\}$ *because* $f(C_1)$ *is going to* C_2 *and vice versa.*

Exercise 11.9

Is it possible for some class C *andA function* $f/1$, $f(t)$ *is not in any class for all* $t \in C$?

Topic 11.6

Problems

Hybrid approach

Exercise 11.10

We have seen both lazy and eager approach. How can we have a mixed lasy/eager approach for EUF solving?

WrongIncrEUF

Exercise 11.11

Show that the following implementation is incomplete

Algorithm 11.4: $WrongIncrEUF(t_1 \bowtie t_2)$

globals: set of terms $Ts := \emptyset$, set of pairs of classes $DisEq := \emptyset$, bool $conflictFound := 0$

$Ts := Ts \cup subTerms(t_1) \cup subTerms(t_2)$;

$C_1 := getClass(t_1)$; $C_2 := getClass(t_2)$; // if t_i is seen first time, create new class

if $\bowtie$ = "=" **then**

- **if** $C_1 = C_2$ **then** **return** ;
- **if** $(C_1, C_2) \in DisEq$ **then** { $conflictFound := 1$; **return**; } ;
- $C := mergeClasses(C_1, C_2)$; $parent(C) := (C_1, C_2, t_1 = t_2)$;
- $DisEq := DisEq[C_1 \mapsto C, C_2 \mapsto C]$;
- **foreach** $f(r_1, \ldots, r_n), f(s_1, \ldots, s_n) \in Ts \wedge \forall i \in 1..n.\ \exists C.\ r_i, s_i \in C$ **do**
  - $WrongIncrEUF(f(r_1, \ldots, r_n) = f(s_1, \ldots, s_n))$;

else

- $DisEq := DisEq \cup (C_1, C_2)$; // $\bowtie$ = "$\neq$"
- **if** $C_1 = C_2$ **then** $conflictFound := 1$; **return** ;

Equality reasoning

Exercise 11.12

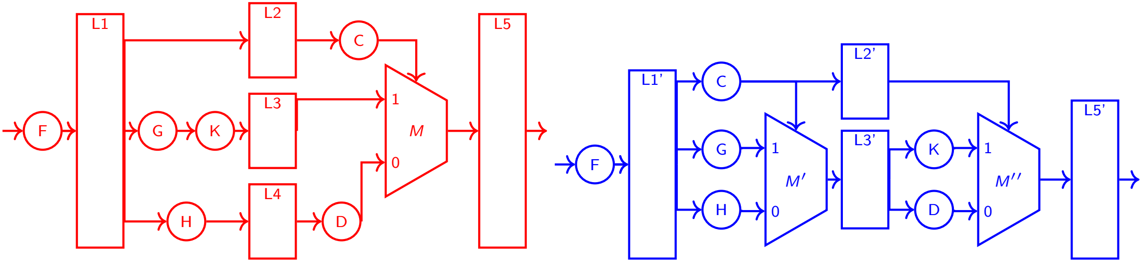
Characterize tuple (n, m, i, j) *such that the following formula is unsat.*

$$f^n(x) = f^m(x) \wedge f^i(x) \neq f^j(x)$$

Exercise: translation validation

Exercise 11.13

Show that the following two circuits are equivalent.



Ls are latches, circles are Boolean circuts, and Ms are multiplexers.

Source: http://www.decision-procedures.org/slides/uf.pdf

End of Lecture 11