Logic of Equality and Uninterpreted Functions (EUF)

Deepak D'Souza

Department of Computer Science and Automation
Indian Institute of Science, Bangalore.

17 March 2025

Outline

Logic of Equality and Uninterpreted Functions (EUF) (KS Ch 4)

Boolean combinations of equality predicates (or quantifier-free fragment of $FO(\{\}, \{f, g, \ldots, h\}, \{c, d, \ldots, e\}))$.

EUF syntax

(Formula) $\varphi ::= Atom \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi$
(Atom) $Atom ::= Term = Term$
(Term) $Term ::= Var \mid Const \mid F(Term)$

Example formula φ_1

$$(x_1 = x_2) \wedge (x_2 = x_3) \wedge (F(x_1) \neq F(x_3)).$$

Example formula φ_2

$$F(x) = F(G(y)) \vee x = y$$

Questions we want to answer

Given an EUF formula φ:

- Satisfiability: Does there exist $M = (D, I, A)$, $D = \mathbb{Z}$ (or any infinite domain), such that $M \models \varphi$.
- Validity: Does $M \models \varphi$, for *every* $M = (D, I, A)$ and $D = \mathbb{Z}$ (or any infinite domain).

Exercise:

- Give examples of satisfiable, unsatisfiable, valid EUF formulas.
- What is the relation between satisfiabililty and validity?

Importance of EUF logic

Many practical applications. Arguing correctness of:

- Assertions in programs
- Program transformation, compilation
- Pipelining in a hardware circuit.

Checking Assertions in Programs

```
main() {
  int x, y = 1;
  while (x != 0) {
    x = foo(x);
    y = foo(y);
  }
}
// assert y == 0

int foo (int) {
  // complex function
  ...
}
```

Potential loop invariant: $x = y$

Using Floyd-Hoare Logic, the assertion is true if following Verification Conditions are valid ($\forall x \forall y \forall x' \forall y' (\ldots)$ is implicit):

C1: $(x = 1 \wedge y = 1) \rightarrow x = y$
C2: $(x = y \wedge x \neq 0 \wedge x' = foo(x) \wedge y' = foo(y)) \rightarrow x' = y'$
C3: $(x = y \wedge \neg(x \neq 0)) \rightarrow y = 0.$

Question can be answered by viewing as validity of an EUF formula.

Program Transformation

Example: Are these programs equivalent?

```
S1: z := (x1 + y1) * (x2 + y2);
```

```
T1: u1 := (x1 + y1);
T2: u2 := (x2 + y2);
T3: z := u1 * u2;
```

We want to check whether (forall $x_1, x_2, y_1, y_2, z_1, z_2, u_1, u_2$)

$$
\begin{aligned}
&(z_1 = (x_1 + y_1) * (x_2 + y_2) \wedge \\
&\ u_1 = x_1 + y_1 \ \wedge u_2 = x_2 + y_2 \ \wedge z_2 = u_1 * u_2) \\
&\rightarrow z_1 = z_2.
\end{aligned}
$$

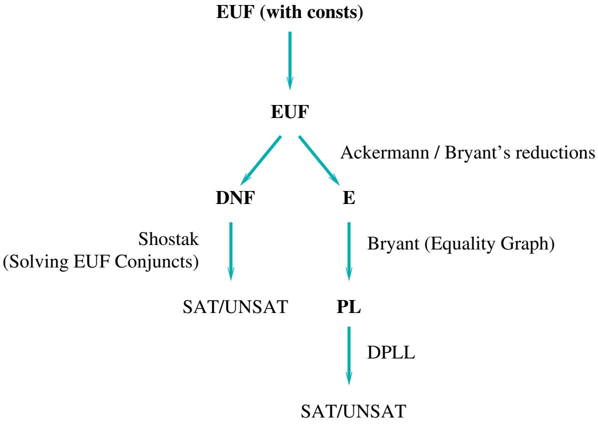
Since reasoning about 32 bit ints and addition and multiplication is difficult, we could instead check whether the EUF formula:

$$
\begin{aligned}
&(z_1 = G(F(x_1, y_1), F(x_2, y_2)) \wedge \\
&\ u_1 = F(x_1, y_1) \ \wedge u_2 = F(x_2 + y_2) \ \wedge z_2 = G(u_1, u_2)) \\
&\rightarrow z_1 = z_2.
\end{aligned}
$$

is valid. Gives a sufficient proof technique.

How do we decide satisfiability of EUF formulas?

Strategies we will look at:



Getting rid of constants (KS Sec 4.1.3)

Consider EUF formulas intrepreted in given domain like $\mathbb{R}$ or $\mathbb{Z}$, with interpreted constants like 1.2 or 3.

Given φ in EUF with intepreted consts, replace each constant k in φ by a new variable c_k and add conjuncts saying that $c_k \neq c_{k'}$ for each pair of distinct constants k, k'.

Example: Replace φ:

$$(y = z \wedge z \neq 1) \vee ((x \neq z) \wedge x = 2)$$

by equisatisfiable φ':

$$[(y = z \wedge z \neq c_1) \vee ((x \neq z) \wedge x = c_2)] \wedge (c_1 \neq c_2).$$

Claim: φ is satisfiable iff φ' is.

A Brute-Force Algorithm

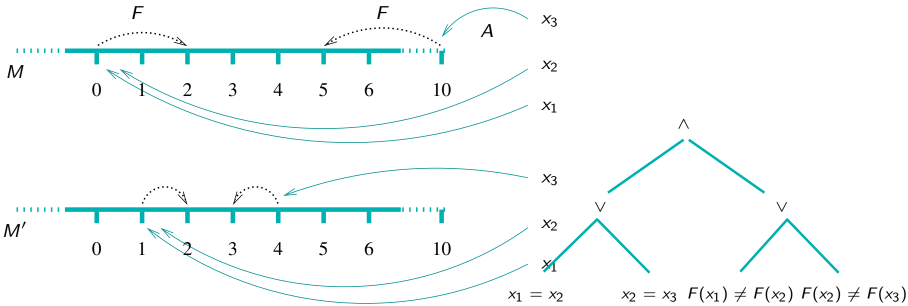
Given EUF formula φ:

1. Let V be the variables in φ, and let k be the number of distinct subterms in φ.
2. Let $U = \{1, \ldots, k\}$
3. For each possible model M with domain U, check if $M \models \varphi$.
4. If some M satisfies φ, output SAT; else output UNSAT.

Example

Example

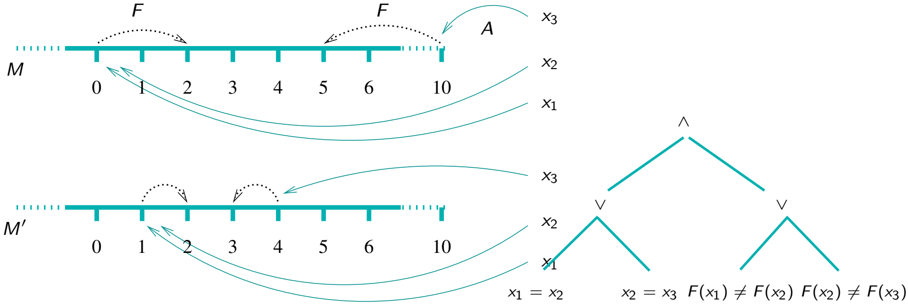
$$(x_1 = x_2 \vee x_2 = x_3) \wedge (F(x_1) \neq F(x_2) \vee F(x_2) \neq F(x_3))$$



Example

Example

$$(x_1 = x_2 \vee x_2 = x_3) \wedge (F(x_1) \neq F(x_2) \vee F(x_2) \neq F(x_3))$$



Claim: If M satisfies φ, then we can construct M' such that $M' \models \varphi$, and M' has domain U.

Congruence Closure Algorithm (Shostak 1978) (KS Sec 4.3)

Given EUF formula φ as conjunction of literals:

1. Consider all subterms t of φ.
2. If $(t_1 = t_2)$ is a predicate in φ, put t_1, t_2 in the same equivalence class. All other terms in their own singleton equivalence classes.
3. If two classes share a term, merge them.
4. Apply congruence closure: If t_1 and t_2 are in the same equivalence class, then merge the equivalence classes of $F(t_1)$ and $F(t_2)$.
5. If there is a disequality $t_1 \neq t_2$ in φ, with t_1 and t_2 in the same equivalence class, return UNSAT. Else return SAT.

Congruence Closure Example I

Example I

Is the following formula satisfiable?

$$(x_1 = x_2) \wedge (x_2 = x_3) \wedge (x_4 = x_5) \wedge (x_5 \neq x_1) \wedge (F(x_1) \neq F(x_3)).$$

Applying congruence closure algorithm:

1. Initial classes: $\{x_1, x_2\}$ $\{x_2, x_3\}$ $\{x_4, x_5\}$ $\{F(x_1)\}$ $\{F(x_3)\}$.
2. Merge classes with shared terms: $\{x_1, x_2, x_3\}$ $\{x_4, x_5\}$ $\{F(x_1)\}$ $\{F(x_3)\}$.
3. Apply congruence closure: $\{x_1, x_2, x_3\}$ $\{x_4, x_5\}$ $\{F(x_1), F(x_3)\}$.
4. Check for disequalities within a class: Yes, so return UNSAT.

Congruence Closure Example II

Example II

Is the following formula satisfiable?

$$(x_1 = x_2) \wedge (x_2 = x_3) \wedge (x_4 = x_5) \wedge (x_5 \neq x_1) \wedge (F(F(x_2)) \neq F(x_4)).$$

Applying congruence closure algorithm:

1. Initial classes:
   $\{x_1, x_2\}$ $\{x_2, x_3\}$ $\{x_4, x_5\}$ $\{F(x_2)\}$ $\{F(F(x_2))\}$ $\{F(x_4)\}$.
2. Merge classes with shared terms:
   $\{x_1, x_2, x_3\}$ $\{x_4, x_5\}$ $\{F(x_2)\}$ $\{F(F(x_2))\}$ $\{F(x_4)\}$.
3. Apply congruence closure:
   $\{x_1, x_2, x_3\}$ $\{x_4, x_5\}$ $\{F(x_2)\}$ $\{F(F(x_2))\}$ $\{F(x_4)\}$.
4. Check for disequalities within a class: No, so return SAT.

Exercise

Exercise

Apply Shostak's congruence closure algorithm to check satisfiability of the following EUF formula:

$$x = f(f(f(f(f(x))))) \land x = f(f(f(x))) \land x \neq f(x)$$

Congruence Closure Correctness

Correctness?

Congruence Closure Correctness

Correctness?

- Argue by induction on number of applications of Step 3, that in any satisfying model M, all terms in one equiv class must be mapped to the same element of domain: that is $M(t) = M(t')$ for each t, t' in an equiv class. Hence if we return UNSAT we are correct.
- Conversely, define a model M comprising equivalence classes:
  - $M = (D, I, A)$ where D is set of equiv classes $\{e_1, e_2, \ldots, e_k\}$,
  - I: $f(e) \mapsto e'$ if there is a term t in e with $f(t)$ in e'. If no such term, $f(e) \mapsto e$. Argue that this interpretation is well-defined.

  Argue that $M(t)$ coincides with the class of t. Hence $M \models \varphi$.

Running time of Shostak's algo is $O(n \log n)$ using a union-find data-structure. Brute-force algo is $O(n^n)$ time. Note that number of subterms in φ is at most $|\varphi|$.

Ackermann's Reduction (KS Sec 11.2.1)

Given EUF formula φ output E formula:

$$\varphi_{flat} \wedge FC_{\varphi},$$

where φ_{flat} replaces function application terms like $F(G(x))$ by new variable fgx etc, and FC encodes functional consistency:

$$(x = y \rightarrow fx \implies fy) \wedge (fx = gfy \rightarrow ffx = fgfy) \wedge \cdots$$

Claim: φ is sat iff $\varphi_{flat} \wedge FC_{\varphi}$ is sat.

Example

Ackermann's reduction

$$(x_1 \neq x_2) \vee (F(x_1) = F(x_2)) \vee (F(x_1) \neq F(x_3))$$

Equality logic formula φ^E is:

$$\begin{array}{ll}
[(x_1 \neq x_2) \vee (fx_1 = fx_2) \vee (fx_1 \neq fx_3)] & (\varphi_{flat}) \\
\wedge\ (x_1 = x_2 \rightarrow fx_1 = fx_2) & (FC_\varphi) \\
\wedge\ (x_2 = x_3 \rightarrow fx_2 = fx_3) & \\
\wedge\ (x_1 = x_3 \rightarrow fx_1 = fx_3). &
\end{array}$$

Exercise

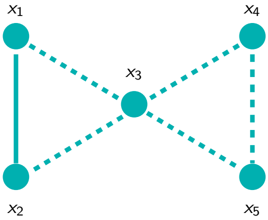
Exercise

Give Ackermann's reduction for this formula

$$(x_1 = x_2) \wedge F(F(G(x_1))) \neq F(F(G(x_2))).$$

Equality graph induced by an E formula (KS Sec 11.3)

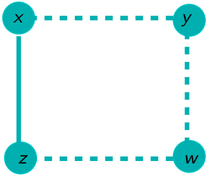
Let φ be a E formula. Then the equality graph G_φ induced by φ is an undirected graph with nodes as variables and "- - -" edge (x_i, x_j) iff the literal $x_i = x_j$ occurs in φ, and "—" edge (x_i, x_j) iff the literal $x_i \neq x_j$ occurs in φ.

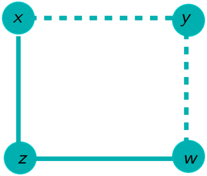


- Contradictory cycles (cycle with exactly one disequality edge).
- Is an abstraction of the original formula.
- Can be used to simplify an E formula.

Contradictory Cycles

A conjunction φ of equality constraints is satisfiable iff the equality graph G_φ induced by φ has no contradictory cycles.





Proof (Exercise).

Solving E-formulas: DNF approach

Example φ

$$x = y \wedge [(y = z \wedge \neg(x = z)) \vee (y = z \wedge \neg(x = w))].$$

$e(\varphi)$: $\quad e_{xy} \wedge [(e_{yz} \wedge \neg e_{xz}) \vee (e_{yz} \wedge \neg e_{xw})].$

Convert $e(\varphi)$ to DNF and check each disjunct using equality graph.

Solving E-formulas: DPLL(T) approach (KS Sec 3.4)

Example φ

$$x = y \wedge [(y = z \wedge \neg(x = z)) \vee (y = z \wedge \neg(x = w))].$$

$e(\varphi)$: $$e_{xy} \wedge [(e_{yz} \wedge \neg e_{xz}) \vee (e_{yz} \wedge \neg e_{xw})].$$

Check satisfiabilty of $e(\varphi)$ using DPLL:

1. If $e(\varphi)$ is SAT, check if satisfying assignment is T-valid;
   - If T-valid, then return SAT;
   - If not T-valid, add negation of the assignment as conflicting clause to $e(\varphi)$, and go back Step 1.
2. If UNSAT, report UNSAT.

Bryant's Graph-Based reduction of E to PL (KS Sec 11.5)

Given E formula φ output PL formula:

$$e(\varphi) \wedge \mathcal{B}_{trans},$$

where $e(\varphi)$ replaces each literal $x_i = x_j$ by a propositional symbol p_{ij}, and $\mathcal{B}_{trans}$ encodes transitivity constraints based on the non-polar graph induced by φ.

Bryant's Graph-Based reduction of E to PL (KS Sec 11.5)

Given E formula φ output PL formula:

$$e(\varphi) \wedge \mathcal{B}_{trans},$$

where $e(\varphi)$ replaces each literal $x_i = x_j$ by a propositional symbol p_{ij}, and $\mathcal{B}_{trans}$ encodes transitivity constraints based on the non-polar graph induced by φ.
Claim: φ is equisatisfiable with $e(\varphi) \wedge \mathcal{B}_{trans}$.

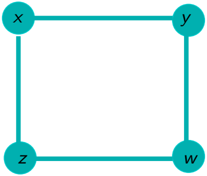
Bryant's Graph-Based reduction: Example

Example φ

$$x = y \wedge [(y = w \wedge w = z \wedge \neg(x = z)) \vee (y = w \wedge \neg(w = z))].$$

$e(\varphi)$: $\quad e_{xy} \wedge [(e_{wy} \wedge e_{wz} \wedge \neg e_{xz}) \vee (e_{wy} \wedge \neg e_{wz})].$

G^{NP}_{φ}:

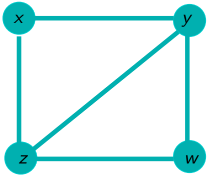


$\mathcal{B}_{trans}$:

$$\begin{array}{lcl} (e_{xy} \wedge e_{wy} \wedge e_{wz}) & \Longrightarrow & e_{xz} \wedge \\ (e_{wy} \wedge e_{wz} \wedge e_{xz}) & \Longrightarrow & e_{xy} \wedge \\ (e_{wz} \wedge e_{xz} \wedge e_{xy}) & \Longrightarrow & e_{wy} \wedge \\ (e_{xz} \wedge e_{xy} \wedge e_{wy}) & \Longrightarrow & e_{wz} \end{array}$$

Bryant's Graph-Based reduction: Using chordal graph

Make G_φ^{NP} chordal (every simple cycle of ≥ 4 vertices has a chord).



Sufficient to check no contradictary triangles [Bryant-Velev CAV 2000):

$$\mathcal{B}_{trans}: \quad \begin{array}{rcl} (e_{xy} \wedge e_{yz}) & \implies & e_{xz} \wedge \\ (e_{yz} \wedge e_{xz}) & \implies & e_{xy} \wedge \\ (e_{xz} \wedge e_{xy}) & \implies & e_{yz} \wedge \\ (e_{wy} \wedge e_{wz}) & \implies & e_{yz} \wedge \\ (e_{wz} \wedge e_{yz}) & \implies & e_{wy} \wedge \\ (e_{yz} \wedge e_{wy}) & \implies & e_{wz}. \end{array}$$

Using equality graph to simplify E formulas

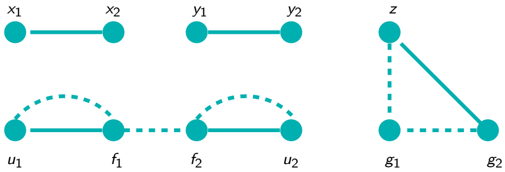
Given E formula φ:

1. Construct equality graph G_φ for φ.
2. If a literal does not occur as part of a contradictory cycle in G_φ, set it to *true*. Obtain φ' in this way.
3. Simplify φ' and go back to Step 2.
4. Ouput φ' as equisatisfiable to φ.

Equality graph: example

Example

$$(u_1 \neq f_1 \vee y_1 \neq y_2 \vee f_1 = f_2) \wedge$$
$$(x_1 \neq x_2 \vee u_2 \neq f_2 \vee g_1 = g_2) \wedge$$
$$(u_1 = f_1 \wedge u_2 = f_2 \wedge z = g_1 \wedge z \neq g_2).$$



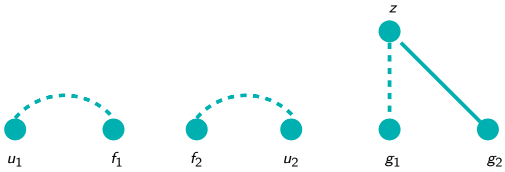
$$(u_1 \neq f_1 \vee true \vee true) \wedge$$
$$(true \vee u_2 \neq f_2 \vee g_1 = g_2) \wedge$$
$$(u_1 = f_1 \wedge u_2 = f_2 \wedge z = g_1 \wedge z \neq g_2).$$

Simplifies to:

$$u_1 = f_1 \wedge u_2 = f_2 \wedge z = g_1 \wedge z \neq g_2.$$

Equality graph: example contd.

$$u_1 = f_1 \wedge u_2 = f_2 \wedge z = g_1 \wedge z \neq g_2.$$



$$true \wedge true \wedge true \wedge true.$$

Simplifies to:

$$true.$$