15-414: **Bug Catching: Automated Program Verification**

# Lecture Notes on SMT Theories & Deciding Uninterpreted Functions

Ruben Martins

Carnegie Mellon University
Lecture 17
Thursday, March 21, 2024

## 1 Introduction

In previous lectures, we studied decision procedures for propositional logic. However, verification conditions that arise in practice often combine expression from different theories. Consider the following examples:

- A combination of linear arithmetic and uninterpreted functions:

$$(x_2 \geq x_1) \land (x_1 - x_3 \geq 2) \land (x_3 \geq 0) \land f(f(x_1) - f(x_2)) \neq f(x_3)$$

- A combination of linear arithmetic and arrays:

$$x = v\{i \leftarrow e\}[j] \land y = v[j] \land x > e \land x > y$$

In this lecture, we will present several examples of SMT theories. We will also take a closer look at the theory of equality with uninterpreted functions and see how we can solve it with a congruence closure algorithm.[1]

### Learning Goals

- SMT theories and specifialized procedures to solve formulas that only use a given theory.

[1] Lecture notes based on [BM07] and [KS16].

- The theory of *equality and uninterpreted functions* (EUF), and how to use the *congruence closure* algorithm for deciding conjunctive, quantifier free formulas in EUF.

## 2 SMT Theories

A first-order theory $T$ is defined by the following components.

- It's signature $\Sigma$ is a set of constant, function, and predicate symbols.
- It's set of axioms $\mathcal{A}$ is a set of closed first-order logic formulae in which only constant, function, and predicate symbols of $\Sigma$ appear.

**Definition 1** ($T$-valid)**.** A $\Sigma$-formula $\varphi$ is valid in the theory $T$ ($T$-valid), if *every interpretation* $I$ that satisfies the axioms of $T$ (i.e., $I \models A$ for every $A \in \mathcal{A}$) also satisfies $\varphi$ (i.e., $I \models \varphi$).

**Definition 2** ($T$-satisfiable)**.** Let $T$ be a $\Sigma$-theory. A $\Sigma$-formula $\varphi$ is $T$-satisfiable if there *exists an interpretation* $I$ such that $I \models A$ and $I \models \varphi$.

**Definition 3** ($T$-decidable)**.** A theory $T$ is decidable if $T \models \varphi$ is decidable for every $\Sigma$-formula. That is, there exists an algorithm that always terminate with "yes" if $\varphi$ is $T$-valid or with "no" if $\varphi$ is $T$-invalid.

SMT supports many different theories such as linear real arithmetic, linear integer arithmetic, fixed-width bitvectors, arrays, and equality with uninterpreted functions. Formulas can combine these theories and we can solve them using the DPLL(T) procedure as described in the previous lecture. To use DPLL(T), we need a decision procedure for each of these theories. However, decision procedures for these and other theories have been developed during the last decades. Even though we will not go into detail on how these procedures work, we will highlight some of the methods and their respective complexity. In this lecture, we will restrict ourselves to quantifier-free theories.

- **Linear Real Arithmetic.** Consider formulas using linear real arithmetic that are conjunctions of linear constraints over $\mathbb{R}$. These formulas can be decided in *polynomial time* but in practice is often solved with the general Simplex method which is in the worst-case exponential. It can also be decided by other exponential methods like the Fourier-Motzkin elimination. If you are interested in known more about the Simplex algorithm you can take a look at the lecture notes from "15-451 Design and Analysis of Algorithms".
- **Linear Integer Arithmetic.** Consider formulas using a conjunction of linear constraints over $\mathbb{Z}$. Deciding if a formula is satisfiable or not in this domain is NP-Complete. We refer the interested reader for the same lecture notes of "15-451 Design and Analysis of Algorithms". These formulas can be solved with techniques such as branch-and-bound (which are based on Simplex) that are commonly used

in commercial linear integer arithmetic solvers such as Gurobi or CPLEX. Other approaches include the Omega Test which is an extension of Fourier-Motkzin.

- **Fixed-Width Bitvectors.** Consider formulas with an arbitrary combination of constraints over bitvectors. Deciding if a formula is satisfiable or not in this domain is NP-Complete. This problem can be reduced to a SAT problem and solved using SAT solvers.
- **Arrays.** Consider formulas with constraints over read/write terms in the theory of arrays. The problem of deciding the satisfiability of these formulas can be reduced to $T_{\mathsf{E}}$ satisfiability. However, because the reduction introduces disjunctions this problem is also NP-Complete.
- **Equality with uninterpreted functions.** Consider formulas with conjunctions of equality constraints over uninterpreted functions. The satisfiability of these formulas can be decided by using the congruence closure algorithm that will be explained in detail in these lecture notes. This algorithm has polynomial time complexity.

## 3 Theory of Equality with Uninterpreted Functions (EUF)

The **theory of equality with uninterpreted functions** $T_{\mathsf{E}}$ has a signature that consists of a single binary predicate $=$, and all possible constant $(a, b, c, x, y, z, \ldots)$ and function $(f, g, h, \ldots)$ symbols:

$$\Sigma_{\mathsf{E}} : \{=, a, b, c, \ldots, f, g, h, \ldots\}$$

The axioms of $T_{\mathsf{E}}$ define the usual meaning of equality (reflexivity, symmetry, and transitivity), as well as *functional congruence*.

1. $\forall x. x = x$ (reflexivity)
2. $\forall x, y. x = y \rightarrow y = x$ (symmetry)
3. $\forall x, y, z. x = y \wedge y = z \rightarrow x = z$ (transitivity)
4. $\forall x, y. x = y \rightarrow f(\bar{x}) = f(\bar{y})$ (congruence)

Function congruence states that whenever the arguments to a function are equal, then the function's value at those arguments must be equal as well. Observe that this is equivalent to the congruence axiom from the theory of arrays, if we replace function applications with their corresponding $read$ terms. In fact, it generalizes array congruence, because it can also stated so that it applies to functions with multiple arguments. If $\bar{x}$ and $\bar{y}$ are sequences $x_0, \ldots, x_n$ and $y_0, \ldots, y_n$ of variables, then Equation 1 formalizes congruence over $n$-ary functions.

$$\forall \bar{x}, \bar{y}. (\textstyle\bigwedge_{i=1}^{n} x_i = y_i) \rightarrow f(\bar{x}) = f(\bar{y}) \tag{1}$$

For the purposes of today's lecture, we only need to consider unary functions of a single argument, because they are sufficient to capture the meaning of $read$ terms in the theory of arrays.

We note that many treatments of this theory also include predicate symbols, and have a corresponding notion of predicate congruence. The algorithm that we present later will work on formulas that include predicates as well, with minimal modifications; our use of EUF to reason about arrays does not require them, so we leave predicates out of the theory for the rest of the lecture.

*Example* 4. Consider the $\Sigma$-formula $\varphi$

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

$\varphi$ is $T_{\mathsf{E}}$-unsatisfiable. We can make the following intuitive argument: substituting $a$ for $f(f(f(a)))$ in $f(f(f(f(f(a))))) = a$ by the first equality yields $f(f(a)) = a$; substituting $a$ for $f(f(a))$ in $f(f(f(a))) = a$ according to this new equality yields $f(a) = a$, contradicting the literal $f(a) \neq a$. Formally, we can apply the axioms of $T_{\mathsf{E}}$ and derive the same contradiction:

1. $f(f(f(f(a)))) = f(a)$ — first literal of $\varphi$ (congruence)
2. $f(f(f(f(f(f(a)))))) = f(f(a))$ — step 1 (congruence)
3. $f(f(a)) = f(f(f(f(f(f(a))))))$ — step 2 (symmetry)
4. $f(f(a)) = a$ — step 3 and second literal of $\varphi$ (transitivity)

### 3.1 Deciding EUF: The Congruence Closure Algorithm

Each positive positive literal $s = t$ of a $\Sigma$-formula $\varphi$ over $T_{\mathsf{E}}$ asserts an equality between two terms $s$ and $t$. Applying the axioms of $T_{\mathsf{E}}$ produces more equalities over terms that occur in $\varphi$. Since there are only a finite number of terms in $\varphi$, only a finite number of equalities among these terms are possible. Hence, one of two situations eventually occurs: either some equality is formed that directly contradicts a negative literal $s' \neq t'$ of $\varphi$; or the propagation of equalities ends without finding a contradiction. These cases correspond to $T_{\mathsf{E}}$-unsatisfiability and $T_{\mathsf{E}}$-satisfiability, respectively, of $\varphi$. In this section, we will formally describe this procedure as forming the **congruence closure** of the equality relation over terms asserted by $\varphi$.

**Models of equality.** We begin by introducing the notion of a *congruence relation* in Definition 5.

**Definition 5** (Congruence relation, congruence class)**.** Consider a set $S$ and functions $F = \{f_1, \ldots, f_n\}$. A relation $R$ over $S$ is a *congruence relation* if for every function $f \in F$, it satisfies the following:

1. Reflexive: $\forall_s \in S. s\, R\, s$

2. Symmetric: $\forall_{s_1,s_2} \in S.s_1\, R\, s_2 \rightarrow s_2\, R\, s_1$
3. Transitive: $\forall_{s_1,s_2,s_3} \in S.s_1\, R\, s_2 \land s_2\, R\, s_3 \rightarrow s_1\, R\, s_3$
4. Congruent: $\forall_{s,t} s\, R\, t \rightarrow f(s)\, R\, f(t)$

We say that two elements $x, y \in S$ are in the same *congruence class* of $R$ whenever $x\, R\, y$, and write $[x]_R$ to denote the set of elements in $x$'s congruence class.

You may have noticed that the requirements of a congruence relation mirror the axioms of our present theory. Suppose that we are shown a congruence relation $R$ over the set $S = \{a, b, f(a), f(b)\}$. The properties of congruence relations make it possible for us to construct a satisfiable EUF formula from $R$. For example, if $R$ relates the pairs $\{(a, b), (f(a), f(b))\}$[2] from $S$, then we could derive:

$$a = b \land f(a) = f(b) \land a \neq f(a) \land a \neq f(b) \land b \neq f(a) \land b \neq f(b)$$

In other words, any pair related by $R$ appears in an equality literal, and any possible pair *not* in $R$ in a negative equality literal. We know that this formula will be satisfiable, because everything that is equated came from $R$, which is reflexive, symmetric, transitive, and congruent.

If we could "reverse" this reasoning, and derive a congruence relation for a given formula, then perhaps we could decide that the formula is satisfiable. For example, given the formula $P \equiv a = f(x) \land a = g(y) \land x \neq y$, then $R = \{(a, (f(x)), (a, g(y)), \ldots)\}$ would be such a relation. Note that the ellipses refer to an infinite set of pairs that follow from nested applications of $f$ and $g$ via congruence. For example, because $a\, R\, f(x)$, congruence says that $f(a)\, R\, f(f(x))$, and that $f(f(a))\, R\, f(f(f(x)))$, and $\ldots$, must also be true. Congruence relations will always be impossible to write down for this reason, and we will instead use the convention of denoting them by their congruence classes only over the terms that appear in the formula. We would thus denote $R$ in this way as $\{\{a, f(x), g(y)\}, \{x\}, \{y\}\}$.

We can say that $R$ *models* $P$, written $R \models P$, as it demonstrates the satisfiability of $P$. In whatever domain the terms of $P$ range over, we could assign a unique element for each congruence class of $R$. Then any assignment where variables and function applications map to the element for their congruence class will satisfy $P$.

To see this concretely, let us assume that $a, b, x, y, f$, and $g$ range over integers. The current relation $R$ has three equivalence classes: one containing $a, f(x), g(y)$, another containing $x$, and one containing $y$. If we let 0 be the element for the first class, 1 be for the second, and 2 for the third, then a satisfying assignment $M$ would be:

$$M(a) = 0, M(x) = 1, M(y) = 2, M(f) = M(g) = [0 \mapsto 0, 1 \mapsto 0, 2 \mapsto 0]$$

To conclude, given a congruence relation over the terms appearing in a formula, we can construct an assignment to the variables and function values appearing in that formula. Moreover, this assignment will be consistent with the axioms of equality, as well as with function congruence.

[2] We do not include the symmetric pairs $(b, a)$ and $(f(a), f(b))$ explicitly to save space, but they must be in $R$ for it to be a congruence relation.

**Minimal models.** Observe that not all congruence relations over $\{a, f(x), g(y), x, y\}$ from the example in the previous paragraph work as models of $P$. For any set $S$ of terms in a formula $P$, the relation containing one congruence class is always trivially a congruence relation. This corresponds to the maximal congruence relation $R^{\max}$ over $S$, and if there is a negative equality literal in $P$, then $R^{\max}$ will not model $P$. In the example from the previous paragraph, this relation would allow $x$ and $y$ to be assigned to the same element 0, because $x \, R^{\max} \, y$.

In general, a congruence relation $R$ does not model a formula $P$ whenever there exist a set of terms $s, t$ where $s \, R \, t$ and a negative equality $s \neq t$ in $P$. Thus, when searching for a relation that models a formula, we want to find the *minimal* congruence relation in order to avoid relating terms that conflict with a negative equality in $P$. This motivates the definition of *congruence closure*, detailed in Definition 6.

**Definition 6** (Congruence closure)**.** The *congruence closure* $R^{\text{cong}}$ of the binary relation $R$ over $S$ is the unique relation which satisfies:

- $R^{\text{cong}}$ relates everything that $R$ does: $R \subseteq R^{\text{cong}}$.
- $R^{\text{cong}}$ is the smallest congruence relation satisfying (1). If $R'$ is a congruence relation that satisfies (1), then $R^{\text{cong}} \subseteq R'$.

Note that the congruence closure of a given relation always exists, because $R^{\max}$ is a congruence relation; in the "worst" case, it may also be the smallest congruence relation containing $R$.

A bit of thought should convince you that if we begin with a relation $R$ that captures the equality literals in $P$, and compute its congruence closure, then whenever $P$ is satisfiable, $R^{\text{cong}}$ will model it. Returning to the previous example,

$$P \equiv a = f(x) \land a = g(y) \land x \neq y$$

The relation that captures the equality literals in $P$ is given by $R = \{(a, f(x)), (a, g(y))\}$ (omitting the necessary reflexive and symmetric pairs for clarity). The congruence closure of $R$ is,

$$R = \{(a, f(x)), (a, g(y)), (f(x), g(y)), (f(a), f(f(x))), (g(a), g(g(y)), (f(f(x)), f(g(y))), \ldots\}$$

For both relations, the congruence classes restricted to $\{a, x, y, f(x), f(y)\}$ (i.e., the terms appearing in the formula), are $\{\{a, f(x), f(y)\}, \{x\}, \{y\}\}$. In other words, in this case we can find a model of $P$ just by processing the equality literals that appear in it because the classes of $R$ are identical to those of $R^{\text{cong}}$.

As you might expect, this isn't always the case. Consider the example from earlier in the notes.

$$\varphi : f(f(f(a))) = a \land f(f(f(f(f(a))))) = a \land f(a) \neq a$$

The initial relation is $R = \{(f^3(a), a), (f^5(a), a)\}$. The set of terms appearing in the formula are $S = \{a, f(a), f^2(a), f^3(a), f^4(a), f^5(a)\}$, so the initial relation gives classes $\{\{a, f^3(a), f^5(a)\}, \{f(a)\}, \{f^2(a)\}, \{f^4(a)\}\}$. If we assign, for example, $f(a)$ and $f^4(a)$

to different elements, then congruence is violated because $a\,R\,f^3(a)$. So in this case we do in fact need to compute the congruence closure, which has just one class:

$$\{\{a, f(a), f^2(a), f^3(a), f^4(a), f^5(a)\}\}$$

In other words, everything in the formula must be related.

In this case, the congruence closure conflicts with $f(a) \neq a$. Can we conclude that the formula is not satisfiable based on just that evidence? Thanks to the work of Shostak [Sho78] in the 1970's, we can answer this question affirmatively. The proof of this result is beyond the scope of today's lecture, but your intuition should serve you well in believing the claim. If the *minimal* relation that satisfies the reflexive, symmetric, transitive, and congruence axioms also conflicts with a negative equality in $P$, then how could one ever find a way to assign these terms to values that did not contradict the negative equality literal? Any such "satisfying" assignment would fail to account for one of the axioms, and thus fail as a model of the EUF formula.

**The algorithm.** To summarize what we have learned so far, we have that a congruence relation over the terms of a formula corresponds to an assignment that is consistent with the axioms of EUF. For a given relation, the congruence closure is the smallest congruence relation that contains $R$. If we begin with a relation that reflects the positive equality literals in a formula, and find its congruence closure, then the result will also give us a satisfying assignment if one exists.

We now turn to computing the congruence closure. The algorithm works explicitly with a representation of the congruence classes, rather than the relation itself. In the following, we will use the infix operator $\cong$ to refer to the congruence closure that is computed by the algorithm, and $P$ to the formula being processed.

1. Let $S_P$ be the set of all terms, and their subterms (recursively), in $P$.
2. Initialize $\cong$ by placing each element of $S_P$ in its own congruence class.
3. For every positive literal $s = t$ in $P$, merge the congruence classes of $s$ and $t$.
4. While $\cong$ changes, repeat the following:
   a) Propagate the congruence axiom, to account for any merged congruence classes from the previous step. For any $s \cong t$, if $f(\ldots, s, \ldots)$ and $f(\ldots, t, \ldots)$ are currently in different congruence classes, then merge them.
5. Check the negative equality literals in $P$ against the computed $\cong$.
   - For any $s \neq t$ appearing in $P$, if $s \cong t$, then return that $P$ is $unsat$.
   - Otherwise, $s \ncong t$ for all $s \neq t$ appearing in $P$, so return that $P$ is $sat$.

Recall the assumptions that we have made about the formula $P$: it is in the conjunctive, quantifier-free frament of EUF. This is why it is possible to return $unsat$ after finding just a single conflict with a negative equality literal. If there were a disjunction in $P$, then this conclusion would not be possible. For the conjunctive quantifier-free fragment, the algorithm is sound and complete.

Soundness means that whenever this procedure terminates, it produces the correct answer, and as we discussed earlier, Shostak [Sho78] proved this. It is also complete, which means that it will always terminate, because the cardinality of the initial set of congruence classes is finite: each time a pair of congruence classes is merged, the procedure makes progress towards termination, which at the very least must occur when there is only one congruence class under $\cong$.

To efficiently implement the procedure, a popular approach is to take advantage of a union-find data structure. This is an acyclic graphical data structure where each node represents a term in $S_P$. Directed edges encode the subterm structure of $P$, i.e., the node for term $f(a)$ would have an edge to the node representing $a$. Congruence classes are also represented by directed edges, by arbitrarily picking a representative element from each congruence class, and drawing edges towards its node from all other members of its class.

Bradley and Manna describe such an implementation strategy [BM07, Chapter 9] that yields $O(e^2)$ runtime, where $e$ is the number of positive equality literals in $P$, with $O(|S_P|)$ merge operations. Downey, Sethi, and Tarjan gave an algorithm with better average-case complexity, $O(e \log e)$ and $O(|S_P|)$ merges [DST80].

We'll conclude this section with a few examples to illustrate the procedure.

*Example* 7. Consider the formula:

$$P : f(a, b) = a \wedge f(f(a, b), b) \neq a$$

The subterm set $S_P$ is $\{a, b, f(a, b), f(f(a, b), b)\}$, so we construct the initial relation by giving each element its own congruence class:

$$\cong_0: \{\{a\}, \{b\}, \{f(a, b)\}, \{f(f(a, b), b)\}\}$$

There is one equality in $P$, $f(a, b) = a$, so we merge the first and third congruence classes:

$$\cong_1: \{\{a, f(a, b)\}, \{b\}, \{f(f(a, b), b)\}\}$$

Now we must check to see if there are congruences to propagate. Now that $a$ and $f(a, b)$ are in the same class, we must determine whether any applications of $f(\cdot, b)$ to either of these terms resides in a different class. We see that $f(\cdot, b)$ applied to $a$, i.e. $f(a, b)$, is in a different class than $f(\cdot, b)$ applied to $f(a, b)$, i.e., $f(f(a, b), b)$. So we merge them, giving the relation:

$$\cong_2: \{\{a, f(a, b), f(f(a, b), b)\}, \{b\}\}.$$

As there are no further applications of $f$ in any but the first equivalence class, there are no further opportunities to propagate congruence, so $\cong_2$ is the congruence closure of $\cong_0$. The last step of the procedure scans the negative literals in $P$ to determine whether $\cong_2$ is a model. In this case, it is not, because there is one negative literal, $f(f(a, b), b) \neq a$, but these terms are in the same congruence class of $\cong_2$. Thus, the formula is $unsat$.

*Example* 8. Now we'll return to the example from earlier, but derive the congruence closure via the algorithm.

$$P : f(f(f(a))) = a \land f(f(f(f(f(a))))) = a \land f(a) \neq a$$

As we said before, the subterm set is $S_P = \{a, f(a), f^2(a), f^3(a), f^4(a), f^5(a)\}$, so the initial relation is:

$$\cong_0 \colon \{\{a\}, \{f(a)\}, \{f^2(a)\}, \{f^3(a)\}, \{f^4(a)\}, \{f^5(a)\}\}$$

There are two positive equality literals in $P$, so we merge $f^3(a)$ and $a$, as well as $f^5(a)$ and $a$:

$$\cong_1 \colon \{\{a, f^3(a), f^5(a)\}, \{f(a)\}, \{f^2(a)\}, \{f^4(a)\}\}$$

We now look for congruences in need of merging. Looking at $a$ and $f^5(a)$, there are no terms of $f^6(a)$ in any classes, so no congruences need to be accounted for. But $a$ and $f^3(a)$ are also related under $\cong_1$, and because $f(a)$ and $f^4(a)$ are in different classes, we merge them.

$$\cong_2 \colon \{\{a, f^3(a), f^5(a)\}, \{f(a), f^4(a)\}, \{f^2(a)\}\}$$

The most recent merge implies that $f^2(a)$ and $f^5(a)$ should also be merged:

$$\cong_3 \colon \{\{a, f^2(a), f^3(a), f^5(a)\}, \{f(a), f^4(a)\}\}$$

And now, because $f^2(a) \cong_3 f^3(a)$, we must merge the two remaining classes:

$$\cong_4 \colon \{\{a, f(a), f^2(a), f^3(a), f^4(a), f^5(a)\}\}$$

This latest $\cong_4$ *must* be the congruence closure, because there are no further opportunities to merge distinct classes. Moving on to the final step, there is one negative literal $f(a) \neq a$ in $P$, and $f(a) \cong_4 a$, so $P$ is $unsat$.

*Example* 9. Consider another formula $P$:

$$P : f(x) = f(y) \land x \neq y$$

The subterm set $S_\varphi$ induces the following initial partition:

$$\cong_0 \colon \{\{x\}, \{y\}, \{f(x)\}, \{f(y)\}\}$$

Since $f(x) = f(y)$ we can merge those two congruence classes:

$$\cong_1 \colon \{\{x\}, \{y\}, \{f(x), f(y)\}\}$$

The union $\{f(x), f(y)\}$ does not yield any new congruences, so we reached a congruence closure. This formula is *satisfiable* since $x \neq y$ but $x$ and $y$ belong to different congruence classes.

## References

[BM07] Aaron R. Bradley and Zohar Manna. *The Calculus of Computation: Decision Procedures with Applications to Verification*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[DST80] Peter J. Downey, Ravi Sethi, and Robert Endre Tarjan. Variations on the common subexpression problem. *Journal of the ACM*, 27(4):758–771, oct 1980.

[KS16] Daniel Kroening and Ofer Strichman. *Decision Procedures - An Algorithmic Point of View*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2016.

[Sho78] Robert E. Shostak. An algorithm for reasoning about equality. *Communications of the ACM*, 21(7):583–585, jul 1978.