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

# Lecture Notes on SMT Theories

Ruben Martins

Carnegie Mellon University
Lecture 17
Tuesday, April 6, 2021

## 1 Introduction

In the previous lecture, we studied the Nelson-Oppen procedure and the DPLL(T) framework that allows us to build decision procedures for formulas that use multiple theories. 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]

## 2 SMT Theories

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 and will talk about quantified SMT in Lecture 19.

- **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

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

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

We start by reviewing the signature and axioms of the theory of equality with uninterpreted functions.

## 3.1 Preliminaries

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

consists of

- = (equality), a binary predicate;
- and all constant, function and predicate symbols.

The axioms of $T_{\mathsf{E}}$ are the following:

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. $\forall \bar{x}, \bar{y}.(\bigwedge_{i=1}^{n} x_i = y_i) \rightarrow f(\bar{x}) = f(\bar{y})$ (congruence)
5. $\forall \bar{x}, \bar{y}.(\bigwedge_{i=1}^{n} x_i = y_i) \rightarrow (p(\bar{x}) \leftrightarrow p(\bar{y}))$ (equivalence)

Consider the $\Sigma$-formula $\varphi$

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

$\varphi$ 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 $f(a) \neq a$. More 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)

Note that even though we have the equivalence axiom, we can transform an instance of this axiom to an instance of the congruence axiom. This transformation allows us to disregard the equivalence axiom. For example, given $\Sigma$-formula:

$$x = y \rightarrow (p(x) \leftrightarrow p(y))$$

introduce a fresh constant $c$ and a fresh function $f_p$, and write

$$x = y \rightarrow ((f_p(x) = c)) \leftrightarrow (f_p(y) = c))$$

In the rest of this lecture, we will consider $\Sigma$-formulae without predicates other than $=$.

## 3.2 Congruence closure

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 $\varphi$.

**Definition 1** (Equivalence relation)**.** A binary relation $R$ over a set $S$ is an equivalence relation if:

1. Reflexive: $\forall_s \in S.sRs$;
2. Symmetric: $\forall_{s_1,s_2} \in S.s_1Rs_2 \rightarrow s_2Rs_1$;
3. Transitive: $\forall_{s_1,s_2,s_3} \in S.s_1Rs_2 \wedge s_2Rs_3 \rightarrow s_1Rs_3$.

For example, the relation $=$ is an equivalence relation over real numbers and $\equiv_2$ is an equivalence relation over $\mathbb{Z}$.

**Definition 2** (Congruence relation)**.** Consider a set $S$ equipped with functions $F = \{f_1, \ldots, f_n\}$. A relation $R$ over $S$ is a congruence relation if it is an equivalence relation and for every $n$-ary function $f \in F$:

$$\forall_{\bar{s},\bar{t}} \bigwedge_{i=1}^{n} s_iRt_i \rightarrow f(\bar{s})Rf(\bar{t})$$

**Definition 3** (Equivalence and congruence classes)**.** For a given equivalence relation over $S$, every member of $S$ belongs to an equivalence class. The equivalence class of $s \in S$ under R is the set:

$$[s]_R \stackrel{\text{def}}{=} \{s' \in S : sRs'\}$$

If R is a congruence relation then this set is called a congruence class.

For example, the equivalence class of 1 under $\equiv_2$ are the odd numbers, and the equivalence class f 6 under $\equiv_3$ the multiples of 3.

**Definition 4** (Equivalence closure)**.** The equivalence closure $R^E$ of the binary relation $R$ over $S$ is the equivalence relation such that:

- $R \subseteq R^E$;
- for all other equivalence relations $R'$ s.t. $R \subseteq R'$, $R^E \subseteq R'$.

Thus, $R^E$ is the smallest equivalence relation that includes $R$.

Let $S = \{a, b, c, d\}$ and $R = \{aRb, bRc, dRd\}$ then

- $aRb$, $bRc$, $dRd \in R^E$ since $R \subseteq R^E$;
- $aRa$, $bRb$, $cRc \in R^E$ by reflexivity;
- $bRa$, $cRb \in R^E$ by symmetry;
- $aRc \in R^E$ by transitivity;
- $cRa \in R^E$ by symmetry;

Hence,

$$R^E = \{aRb, bRa, aRz, bRb, bRc, cRb, cRc, aRc, cRa, dRd\}.$$

**Definition 5** (Subterm set)**.** The subterm set $S_\varphi$ of $\Sigma$-formula $\varphi$ is the set that contains precisely the subterms of $\varphi$.

For example, the subterm set of $\varphi$:

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

is

$$S_\varphi = \{a, b, f(a,b), f(f(a,b),b)\}.$$

Now we relate the congruence closure of a $\Sigma$-formula's subterm set with its $T_{\mathsf{E}}$-satisfiability. Given $\Sigma$-formula $\varphi$

$$\varphi : s_1 = t_1 \wedge \ldots \wedge s_m = t_m \wedge s_{m+1} \neq t_{m+1} \wedge \ldots \wedge s_n \neq t_n$$

with subterm set $S_\varphi$, $\varphi$ is $T_{\mathsf{E}}$-satisfiable iff there exists a congruence relation $\sim$ over $S_\varphi$ such that

- for each $i \in \{1, \ldots, m\}, s_i \sim t_i$;
- for each $i \in \{m+1, \ldots, n\}, s_i \not\sim t_i$.

The goal of the congruence closure algorithm is to construct the congruence relation of a formula's subterm set, or to prove that no congruence relation exists. The algorithm performs the following steps:

1. Construct the congruence closure $\sim$ of

   $$\{s_1 = t_1, \ldots, s_m = t_m\}$$

   over the subterm set $S_\varphi$. Then

   $$\sim \models s_1 = t_1 \wedge \ldots \wedge s_m = t_m$$

2. If $s_i \sim t_i$ for any $i \in \{m+1, \ldots, n\}$ then $\varphi$ is unsatisfiable;
3. Otherwise, $\sim \models \varphi$ and $\varphi$ is satisfiable.

How do we actually construct the congruence closure in Step 1? Initially, begin with the finest congruence relation $\sim_0$ given by the partition

$$\{\{s\} : s \in S_\varphi\}$$

in which each term of $S_\varphi$ is its own congruence class. Then, for each $i \in \{1, \ldots, m\}$, impose $s_i = t_i$ by merging the congruence classes

$$[s_i] \sim_{i-1} \text{ and } [t_i] \sim_{i-1}$$

to form a new congruence relation $\sim_i$. To accomplish this merging, first form the union of $[s_i] \sim_{i-1}$ and $[t_i] \sim_{i-1}$. Then propagate any new congruence that arise within this union.

*Example* 6. Consider the $\Sigma$-formula $\varphi$

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

Construct the following initial partition by letting each member of the subterm set $S_\varphi$ be its own class:

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

According to the first literal $f(a,b) = a$, merge

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

to form partition

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

According to the congruence axiom,

$$f(a,b) \sim a, b \sim b \text{ implies } f(f(a,b),b) \sim f(a,b),$$

resulting in the new partition

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

This partition represents the congruence closure of $S_\varphi$. Now, it is the case that

$$\{\{a,f(a,b),f(f(a,b),b)\},\{b\}\} \models \varphi \ ?$$

No! Since $f(f(a,b),b) \sim a$ but $\varphi$ asserts that $f(f(a,b),b) \neq a$. Therefore, $\varphi$ is $T_{\mathsf{E}}$-unsatisfiable.

In summary, the ***congruence closure algorithm*** for a formula $\varphi$ in $T_{\mathsf{E}}$ work as follows.

1. Construct the subterm set $S_\varphi$ and put each term in its own congruence class.
2. Process the equalities $t_i = t_j \in \varphi$ and merge the congruence classes classes $\{t_i\}$ and $\{t_j\}$.
3. Compute the congruence closure: given two terms $t_i, t_j$ that are in the same class and that $f(t_i)$ and $f(t_j)$ are terms in $S_\varphi$ for some uninterpreted function $f$, merge these two classes together. Repeat until fixpoint.
4. If there exists $t_i \neq t_j \in \varphi$ such that $t_i$ and $t_j$ are in the same class, return *unsatisfiable*. Otherwise, return *satisfiable*.

This algorithm can be implemented efficiently using a directed acyclic graph and has polynomial time complexity.

Let's apply the congruence closure algorithm to a few other examples.

*Example 7.* Let's consider the same formula presented on page 3 of these lecture notes but now let's use the congruence closure algorithm to determine its satisfiability.

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

From the subterm set $S_\varphi$, the initial partition is

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

where, for example $f^3(a)$ abbreviates $f(f(f(a)))$. According to the literal $f^3(a) = a$ we can merge these two terms. Similarly, we can also merge $f^5(a)$ with $a$.

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

Using congruence propagation we can infer that:

$$f^3(a) \sim a \text{ implies } f^4(a) \sim f(a)$$

and

$$f^4(a) \sim f(a) \text{ implies } f^5(a) \sim f^2(a)$$

Which gives the partition:

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

If we now propagate the congruence

$$f^3(a) \sim f^2(a) \text{ implies } f^4(a) \sim f^3(a)$$

yields the partition

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

which represents the congruence closure in which all of $S_\varphi$ are equal. Now since $f(a) \sim a$ but $\varphi$ says that $f(a) \neq a$ then $\varphi$ is $T_{\mathsf{E}}$-unsatisfiable.

*Example 8.* Consider another formula $\varphi$:

$$\varphi : f(x) = f(y) \wedge x \neq y$$

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

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

Since $f(x) = f(y)$ we can merge those two partitions

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

The union $\{f(x), f(y)\}$ does not yield any new congruences, so this is the final partition. This formula is satisfiable since $x \neq y$ but $x$ and $y$ belong to different congruence classes.

## 4 Summary

- SMT solvers support different SMT theories and each of these have specialized procedures to solve formulas that only use a given theory.
- Most decision procedures are solving NP-Complete problems.
- "Almost all proofs require reasoning about equalities" (Nelson and Oppen). The theory of equality with uninterpreted functions is one of the most important theories of SMT solvers.
- Congruence closure can be used to check the satisfiability of a formula with equality and uninterpreted functions.
- Congruence closure algorithm has polynomial complexity.

## 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.

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