15-414: Bug Catching: Automated Program Verification

Lecture Notes on Theory Combination

Matt Fredrikson

Carnegie Mellon University
Lecture 17
Thursday, October 29, 2026

1 Introduction

In Lecture 16 we studied a decision procedure for a first-order theory: equality with uninterpreted functions (EUF). As you might expect, in practice we typically want to reason about more than one theory at a time. For example, we might want to reason about the theory of integers $\mathsf{T}_{\mathbb{Z}}$ and the theory of equality with uninterpreted functions T_{E}.

In this lecture we will study how to combine first-order theories, and in particular how to use independent theory solvers to decide the satisfiability of a formula that is built from symbols from multiple theories. For now, we will continue assuming that the formulas are conjunctive and quantifier-free. We will study the *Nelson-Oppen* procedure for combining theories, which is the basis for most modern SMT solvers, and see how it "glues" independent theory solvers together by exploiting the *stability* and *convexity* properties of the theories involved. At the end of the lecture, we will see how to handle disjunctions by combining a SAT solver with a theory solver.

Learning Goals

1. Combined first-order theories.
2. The Nelson-Oppen procedure for combining theories.
3. Stability and convexity properties of first-order theories.
4. Propositional abstractions and the DPLL(T) framework.

2 Review: First-Order Theories

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

- It's signature Σ 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 Σ appear.

Having defined a theory's signature and axioms, we can reason about the same type of properties related to the semantics of a formula as we have been so far, namely validity and satisfiability.

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

Definition 2 (T-satisfiable)**.** Let T be a Σ-theory. A Σ-formula P is T-satisfiable if there *exists a model* M such that $M \models A$ and $M \models P$.

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

For example, the **theory of equality with uninterpreted functions** $\mathsf{T_E}$ 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 $\mathsf{T_E}$ 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)

3 Theory Combination

Now we turn towards combining decision procedures to handle formulas that have symbols from more than one theory.

Definition 4 (Theory combination)**.** Given two theories T_1 and T_2 with signatures Σ_1 and Σ_2, respectively, the theory combination $T_1 \oplus T_2$ is a $(\Sigma_1 \cup \Sigma_2)$-theory defined by the axiom set $T_1 \cup T_2$.

Definition 5 (The theory combination problem)**.** Let P be a $\Sigma_1 \cup \Sigma_2$ formula. The theory combination problem is to decide whether P is $T_1 \oplus T_2$-valid. Equivalently, the problem is to decide whether the following holds: $T_1 \oplus T_2 \models P$.

Given a Σ-formula P in T_E and a Σ-formula ψ in $\mathsf{T}_\mathbb{Z}$ can we check the satisfiability of $P \cup \psi$ by checking the satisfiability of P and ψ independently and combining the results? **No!** This is not a sound procedure for the theory combination problem. Consider the following counterexample:

$$P = f(x) \neq f(y)$$
$$\psi = x + y = 0 \wedge x = 0$$

Both P and ψ are satisfiable but P implies that $x \neq y$ and ψ implies that $x = y$, therefore their combination is not satisfiable!

4 The Nelson-Oppen Combination Procedure

The Nelson-Oppen combination procedure solves the theory combination problem for theories T_1 and T_2, as long as those theories satisfy a few properties.

- Both theories T_1 and T_2 are quantifier-free (conjunctive) fragments.
- Equality (=) is the only symbol in the intersection of their signatures.
- Both theories have constants that are interpreted over an infinite domain.

The motivation for the first two properties should be clear by intuition. As we saw in the previous lecture, working with conjunctive quantifier-free formulas removes the possibility of having to do case analysis. The fact that $=$ is the only symbol shared between T_1 and T_2 avoids "overloading" of symbols that might introduce spurious relationships between terms, and as we will see, both theories must have equality in order for the approach to work.

The third property might not be as obvious. To make sure that we understand what this restriction means, consider the theory $T_{a,b}$ with signature $\Sigma_T : \{a, b, =\}$ where both a and b are constants. Suppose it has a single axiom:

$$\forall x. x = a \vee x = b$$

This axiom says that every model of the theory must map variables to either a or b. Thus, there is no way to interpret the theory over an infinite domain without violating this axiom. On the other hand, the theories of integers and real arithmetic are interpreted over an infinite domain.

But why would this matter for a decision procedure? This has to do with the way that the Nelson-Oppen procedure first isolates theories, and then coordinates between them by introducing new equalities. The technique follows the steps below, for a given formula P over theories $T_1, \ldots, T_n$.

1. **Purification**: Partition the literals of P into new conjunctive formulas $P_1, \ldots, P_n$, where P_i contains only symbols from T_i.
2. **Theory solving**: Apply the decision procedure for T_i to P_i. If one of the formulas is unsatisfiable, then so is P.
3. **Equality propagation**: As illustrated in the example earlier, the fact that each P_i is independently satisfiable does not mean that their combination in P is. This step gradually adds more information to each P_i by searching for equalities that are implied by the other P_j formulas.
   a) If there exists i, j such that P_i implies an equality between variables of P that is not implied by P_j, add the equality to P_j and return to step 2.
   b) Otherwise, if there are no such equalities to add, then P is satisfiable.

Returning to the question of why the theories must be interpreted over an infinite domain, suppose that we have a formula over $T_{a,b} \cup \mathsf{T_E}$, where $T_{a,b}$ is the toy theory with two constants and equality from earlier:

$$w = x \wedge f(x) \neq f(y) \wedge f(y) \neq f(z) \wedge f(x) \neq f(z)$$

Then after purification, the $T_{a,b}$ formula will just be $w = x$, and the $\mathsf{T_E}$ formula will have the rest of the (negative) literals. Equality propogation will not add anything to either formula, because the only things that could be implied are *negated* equalities, i.e., congruence from EUF implies that $x \neq y$, $y \neq z$, and $x \neq z$. Nelson-Oppen does not propagate negated equalities, so step 3b will apply, and return sat. This is incorrect, because the axiom from $T_{a,b}$ requires w, x, y, and z to be assigned to either the constant a or b, which is not consistent with the above formula.

This example should illustrate the need for the third requirement given above. Note that researchers have explored ways of combining finite-domain theories, and it is often possible to do so in practice. Tinelli and Zarba proposed an approach that attempts to compute a lower bound on the size of the domain that a formula must be satisfied in. This bound can be shared between theories during equality propagation, and if the bound ever contradicts the axioms of a given theory, then the corresponding solver can return unsat. However, it is not always possible to compute this bound, and if it is not sufficiently tight, then the result might still be incorrect.

Example 6. Now we'll see how the technique works on an example from the theory of real arithmetic combined with EUF.

$$\phi = f(f(x) - f(y)) \neq f(z) \wedge x \leq y \wedge y + z \leq x \wedge 0 \leq z$$

For the purification step, we look at any term containing symbols from more than one theory. For example, $f(x) - f(y)$ contains subtraction from real arithmetic, and function application from EUF. To separate this term into pure components, we equate the "alien" subexpressions $f(x)$ and $f(y)$ with fresh variables, and replace their occurence in the subtraction term with the new variables:

$$v_1 = f(x) \wedge v_2 = f(y) \wedge f(v_1 - v_2) \neq f(z) \wedge x \leq y \wedge y + z \leq x \wedge 0 \leq z$$

There is still one impure term, $f(v_1 - v_2)$, so we equate $v_1 - v_2$ with the fresh variable v_3, and subustitute:

$$v_1 = f(x) \wedge v_2 = f(y) \wedge v_3 = v_1 - v_2 \wedge f(v_3) \neq f(z) \wedge x \leq y \wedge y + z \leq x \wedge 0 \leq z$$

Now the formula is pure, and can be easily separated into a formula $P_{\mathbb{R}}$ containing only real arithmetic, and a formula P_{E} containing only equality and uninterpreted functions.

$$\begin{array}{rcl} P_{\mathbb{R}} & \equiv & v_3 = v_1 - v_2 \wedge x \leq y \wedge y + z \leq x \wedge 0 \leq z \\ P_{\mathsf{E}} & \equiv & v_1 = f(x) \wedge v_2 = f(y) \wedge f(v_3) \neq f(z) \end{array}$$

Moving on, the next step is to look for implied equalities that are not already present in either formula. There are several opportunities.

- Together, $x \leq y$, $y + z \leq x$, and $0 \leq z$ imply that both $x = y$ and $z = 0$.
- On the EUF side, once $x = y$ has been added, then $f(x) = f(y)$ by congruence, so $v_1 = v_2$.
- Once $v_1 = v_2$ is added to $P_{\mathbb{R}}$, it implies that $v_3 = z$.

After adding these implied equalities, we have left with the following formulas.

$$\begin{array}{rcl} P_{\mathbb{R}} & \equiv & v_3 = v_1 - v_2 \wedge x \leq y \wedge y + z \leq x \wedge 0 \leq z \wedge x = y \wedge z = 0 \wedge v_1 = v_2 \wedge v_3 = z \\ P_{\mathsf{E}} & \equiv & v_1 = f(x) \wedge v_2 = f(y) \wedge f(v_3) \neq f(z) \wedge x = y \wedge v_1 = v_2 \wedge v_3 = z \end{array}$$

Now we see that P_{E} is not satisfiable, because $v_3 = z$ and $f(v_3) \neq f(z)$ is not consistent with the congruence axiom.

4.1 Convexity

Before moving on, we point out that the procedure described above is only valid for *convex* theories.

Definition 7 (Convex theory)**.** A theory T is convex if, whenever a conjunctive Σ-formula P implies a finite disjunction of equalities:

$$P \rightarrow \bigvee_{i=1}^{n} x_i = y_i$$

it must also imply at least one of those equalities on its own:

$$P \rightarrow x_i = y_i \text{ for some i } \in \{1, \cdots, n\}$$

An example of a nonconvex theory is the theory of integers ($\mathsf{T}_{\mathbb{Z}}$). For instance, while the following is valid:

$$x_1 = 1 \wedge x_2 = 2 \wedge 1 \leq x_3 \wedge x_3 \leq 2 \rightarrow (x_3 = x_1 \vee x_3 = x_2)$$

Neither of the isolated cases are:

$$x_1 = 1 \wedge x_2 = 2 \wedge 1 \leq x_3 \wedge x_3 \leq 2 \rightarrow x_3 = x_1$$
$$x_1 = 1 \wedge x_2 = 2 \wedge 1 \leq x_3 \wedge x_3 \leq 2 \rightarrow x_3 = x_2$$

Consider the following formula defined over $\mathsf{T}_{\mathbb{Z}}$ and T_{E}:

$$P = 1 \leq x \wedge x \leq 2 \wedge f(x) \neq f(1) \wedge f(x) \neq f(2)$$

This formula is clearly unsatisfiable, but Nelson-Oppen will return sat, for reasons very similar to the example discussed earlier with $T_{a,b}$.

In practice, SMT solvers use an extended version of Nelson-Oppen that propagates implied disjunctions of equalities. The details of this extension are beyond the scope of the lecture, but note that adding additional disjunctions to a formula requires case-splitting, which can quickly become expensive. So, while it is possible to combine non-convex theories with others, one should be aware that doing so may make the solver's job intractible, and explore other options.

5 Propositional Encodings

So far, we have assumed that the formulas given to our theory solvers are conjunctive and quantifier-free. To handle formulas with disjunction, one could always convert to disjunctive normal form. However, this conversion can be exponentially expensive. We will now see how to use a SAT solver to handle the Boolean structure of a formula, while a conjunctive theory solver decides satisfiability in the background theory. This approach is called DPLL(T), where T refers to the theory that we wish to solve.

DPLL is a procedure for propositional satisfiability that searches for a satisfying assignment by choosing truth values for propositional variables. It propagates choices forced by clauses, and backtracks when those choices lead to a contradiction. For our purposes, we can treat it as a SAT solver that either returns a satisfying assignment or reports that the propositional formula is unsatisfiable. The theory solver can be a procedure such as congruence closure from Lecture 16, or a combination of theory solvers using the procedure from earlier in this lecture.

5.1 Propositional abstractions

The first insight needed to understand how DPLL(T) works is that it is possible to "abstract" a quantifier-free theory formula P as a propositional formula $B(P)$. The way that we go about accomplishing this is motivated by the way that we ultimately want to make use of DPLL, which only understands propositional formulas. We aim for two key properties.

- If P is satisfiable in T, then $B(P)$ should be satisfiable also.
- If $B(P)$ is unsatisfiable, then P should be unsatisfiable in T as well.

Why is $B(P)$ an abstraction? Note that if P is *not* satisfiable in T, then $B(P)$ could be either satisfiable or unsatisfiable. In this sense, $B(P)$ has lost information that was present in P, so it is an abstraction. However, these properties do allow us to determine unsatisfiability in some cases by applying DPLL to $B(P)$. In the case where $B(P)$ is satisfiable, we will see that the abstraction, plus DPLL's sat decision, provides enough information to continue making progress on deciding P.

We define the propositional abstraction of a quantifier-free Σ-formula recursively. For each distinct theory atom A_i, choose a fresh propositional variable p_i, and use the same variable at every occurrence of that atom. Negated atoms are handled by the rule for negation.

$$\begin{array}{lcl} B(A_i) & = & p_i \\ B(\neg P) & = & \neg B(P) \\ B(P \wedge Q) & = & B(P) \wedge B(Q) \\ B(P \vee Q) & = & B(P) \vee B(Q) \\ B(P \rightarrow Q) & = & B(P) \rightarrow B(Q) \end{array}$$

For example, given the formula:

$$P : g(a) = c \wedge (f(g(a)) \neq f(c) \vee g(a) = d) \wedge c \neq d$$

we use the following correspondence between theory atoms and propositional variables:

p_1	$g(a) = c$
p_2	$f(g(a)) = f(c)$
p_3	$g(a) = d$
p_4	$c = d$

The propositional abstraction of P is then:

$$B(P) = p_1 \wedge (\neg p_2 \vee p_3) \wedge \neg p_4$$

Note that we can also define B^{-1}, which maps the Boolean variables back to the atoms in the original formula and preserves the logical connectives. For example, $B^{-1}(p_1 \wedge p_3 \wedge \neg p_4)$ corresponds to the formula $g(a) = c \wedge g(a) = d \wedge c \neq d$.

5.2 Combining theory solvers with DPLL

The propositional abstraction provides us with a "lazy" way to solve SMT. Given a quantifier-free Σ-formula P, we can determine its satisfiability in T by performing the following procedure:

1. Construct the propositional abstraction, and let $F = B(P)$.
2. Use the SAT solver to decide F. If F is unsatisfiable, then P is unsatisfiable in T.
3. Otherwise, get a satisfying assignment M for F, assigning every propositional variable introduced by B.

4. Construct a conjunction R that contains p_i whenever M assigns p_i true, and $\neg p_i$ whenever M assigns p_i false.
5. Send $B^{-1}(R)$ to the T-solver.
6. If the T-solver reports that $B^{-1}(R)$ is satisfiable, then P is satisfiable in T.
7. Otherwise, update $F := F \wedge \neg R$ and return to step 2.

The formula sent to the theory solver is a conjunction of theory literals, even when P contains disjunctions. Because M satisfies $B(P)$, any theory model of $B^{-1}(R)$ also satisfies P.

Note that if $B^{-1}(R)$ is unsatisfiable, we cannot terminate, since there may be another assignment to $B(P)$ that is consistent with the theory. On step 7 we add $\neg R$ to F since, if we did not, we could get the same assignment M again. We call $\neg R$ a **theory conflict clause** that prevents the SAT solver from going down the same path in future iterations. There are only finitely many Boolean assignments, and each unsuccessful iteration rules out one of them, so the procedure terminates as long as the SAT solver and theory solver do.

Example 8. Returning to the formula from the previous section, suppose that the SAT solver first returns the assignment:

$$M = \{p_1, \neg p_2, p_3, \neg p_4\}$$

We construct $R = p_1 \wedge \neg p_2 \wedge p_3 \wedge \neg p_4$ and send the following conjunction to the EUF solver:

$$B^{-1}(R) : g(a) = c \wedge f(g(a)) \neq f(c) \wedge g(a) = d \wedge c \neq d$$

This is unsatisfiable, since $g(a) = c$ and $g(a) = d$ imply $c = d$, but the conjunction also requires $c \neq d$. We add the conflict clause:

$$\neg p_1 \vee p_2 \vee \neg p_3 \vee p_4$$

Assume that the SAT solver next returns $M = \{p_1, p_2, p_3, \neg p_4\}$. The corresponding theory conjunction is:

$$g(a) = c \wedge f(g(a)) = f(c) \wedge g(a) = d \wedge c \neq d$$

This is unsatisfiable for the same reason as before, so we add another conflict clause:

$$\neg p_1 \vee \neg p_2 \vee \neg p_3 \vee p_4$$

The only remaining satisfying assignment to F is $M = \{p_1, \neg p_2, \neg p_3, \neg p_4\}$. This time, the theory conjunction is:

$$g(a) = c \wedge f(g(a)) \neq f(c) \wedge g(a) \neq d \wedge c \neq d$$

We can see that this is unsatisfiable since $g(a) = c$ but $f(g(a)) \neq f(c)$, contrary to congruence. We add the conflict clause:

$$\neg p_1 \vee p_2 \vee p_3 \vee p_4$$

Now F is unsatisfiable, so we can conclude that the original formula P is unsatisfiable in EUF.