15-414: Bug Catching: Automated Program Verification

Lecture Notes on Satisfiability Modulo Theories

Matt Fredrikson Ruben Martins

Carnegie Mellon University

Lecture 14

1 Introduction

In the previous lecture 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:

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

- A combination of linear arithmetic and arrays:

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

In this lecture, we will show how we can solve formulas that combine multiple theories by using the Nelson-Oppen combination method and the DPLL(T) framework. [1]

2 Preliminaries

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

- Its signature Σ is a set of constant, function, and predicate symbols.
- Its 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.

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

Definition 1 (T-valid)**.** A Σ-formula φ 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 φ (i.e., $I \models \varphi$).

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

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

2.1 Example of Theories

Some theories that we will use throughout this lecture are:

- The theory of equality with uninterpreted functions (T_{E}).
- The theory of integers ($T_{\mathbb{R}}$).

The **theory of equality with uninterpreted functions** T_{E} is the simplest first-order theory. Its signature

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

consists of

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

The axioms of T_{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)

The **theory of reals** $T_{\mathbb{R}}$ has signature

$$\Sigma_{\mathbb{R}} : \{0, 1, +, -, \cdot, =, \geq\}$$

where

- 0 and 1 are constants;
- + (addition) and $\cdot$ (multiplication) are binary functions;

- - (negation) is a unary function;
- and = (equality) and $\geq$ (weak inequality) are binary predicates.

$T_{\mathbb{R}}$ has a complex axiomatization and we will not describe all its axioms here since they are not essential to the understanding of the Nelson-Oppen procedure and the DPLL(T) framework. We refer the interested student to [BM07] for a detailed reading on the axiomatization of the theory of reals.

2.2 Theory combination

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 φ be a $\Sigma_1 \cup \Sigma_2$ formula. The theory combination problem is to decide whether φ is $T_1 \oplus T_2$-valid. Equivalently, the problem is to decide whether the following holds: $T_1 \oplus T_2 \models \varphi$.

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

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

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

3 The Nelson-Oppen Combination Procedure

The Nelson-Oppen combination procedure solves the theory combination problem for theories T_1 and T_2 that comply with the following restrictions:

- Both theories T_1 and T_2 are quantifier-free (conjunctive) fragments.
- Equality (=) is the only symbol in the intersection of their signatures, i.e., $\Sigma_1 \cap \Sigma_2 = \{=\}$.
- Both theories are stably infinite.

Definition 6 (Stably infinite)**.** A theory T with signature Σ is stably infinite if for every satisfiable Σ_T-formula φ, there is an interpretation that satisfies φ and that has a universe of infinite cardinality

Consider the theory $T_{a,b}$ with signature $\Sigma_T : \{a, b, =\}$ where both a and b are constants and with the following axiom:

- $\forall x. x = a \lor x = b$ (two)

Because of axiom (two), every interpretation I is such that the domain of I has at most two elements. Therefore, $T_{a,b}$ is not stably infinite. Note that most of the theories of interest for program verification are stably infinite, e.g. theory of equality of uninterpreted functions and theory of integers.

The Nelson-Oppen procedure for a formula φ that combines different theories consists of:

1. **Purification**: Purify φ into $F_1, \ldots, F_n$.
2. Apply the decision procedure for T_i to F_i. If there exists i such that F_i is unsatisfiable in T_i, then φ is unsatisfiable.
3. **Equality propagation**: If there exists i, j such that F_i T_i-implies an equality between variables of φ that is not T_j-implied by F_j, add this equality to F_j and go to step 2.
4. If all equalities have been propagated then the formula is satisfiable.

3.1 Purification and equality propagation

Purification is a satisfiability-preserving transformation of the formula, after which each atom is from a specific theory. In this case, we say that all the atoms are **pure**. More specifically, given a formula φ, purification generates an equisatisfiable formula φ' as follows:

1. Let $\varphi' := \varphi$.
2. For each "alien" subexpression ϕ in φ':
   - Replace ϕ with a new auxiliary variable a_ϕ
   - Constraint φ' with $a_\phi = \phi$.

Consider the following formula:

$$\varphi = f(x + g(y)) \leq g(a) + f(b)$$

This formula combines the theories T_{E} and $T_{\mathbb{R}}$. Below we show the purification of φ into φ' defined over $T_{\mathbb{R}}$ and φ'' defined over T_{E}

Purification	
φ' ($T_{\mathbb{R}}$)	φ'' (T_{E})
$u_4 = x + u_1 \land$	$u_1 = g(y) \land$
$u_5 \leq u_2 + u_3$	$u_2 = g(a) \land$
	$u_3 = f(b) \land$
	$u_5 = f(u_4)$

Observe that φ' only contains atoms from $T_{\mathbb{R}}$ and φ'' only contains atoms from T_{E}. A variable is shared if it occurs in both formulas and local otherwise. For example, $\{u_1, u_2, u_3, u_4, u_5\}$ are shared variables since they appear in both φ' and φ'' and variables $\{x, y, a, b\}$ are local to either φ' ($\{x\}$) or φ'' ($\{y, a, b\}$).

Consider another formula:

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

We will show how determine the satisfiability of ϕ with the Nelson-Oppen procedure. We start by doing purification and then perform equality propagation over the shared variables.

Purification	
ϕ' ($T_{\mathbb{R}}$)	ϕ'' (T_{E})
$x \leq y \wedge$	$f(w) \neq f(z) \wedge$
$y + z \leq x \wedge$	$u = f(x) \wedge$
$0 \leq z \wedge$	$v = f(y)$
$w = u - v$	
Equality propagation	
$x = y \wedge$	$x = y \wedge$
$u = v \wedge$	$u = v \wedge$
$w = z$	$w = z \wedge$
	unsat

Observe that $x \leq y$, $y + z \leq x$ and $0 \leq z$ implies that $x = y$ and $z = 0$. Therefore, we add $x = y$ to both formulas. Since $x = y$ this implies that $f(x) = f(y)$ and therefore $u = v$. Since $u = v$ and $w = u - v$ than this implies that $w = 0$ which means that $w = z$. However, if $w = z$ than $f(w) = f(z)$ but ϕ'' contains $f(w) \neq f(z)$. Hence, ϕ is unsatisfiable.

3.2 Convex theories

The Nelson-Oppen procedure described in the previous section is only valid for convex theories. Note that this procedure can be modified to handle nonconvex theories but for simplification purposes we omit that version.

Definition 7 (Convex theory). A Σ-theory T is convex if for every conjunctive Σ-formula φ:

$$(\varphi \rightarrow \bigvee_{i=1}^{n} x_i = y_i) \text{ is } T\text{-valid for some finite } n > 1 \rightarrow$$
$$(\varphi \rightarrow x_i = y_i) \text{ is } T\text{-valid for some i } \in \{1, \cdots, n\}$$

where x_i, y_i, for $i \in \{1, \cdots, n\}$, are some variables.

In other words, in a convex theory T, if a formula T-implies a disjunction of equalities, it also T-implies at least one of these equalities separately.

An example of a noncovex theory is the theory of integers ($T_{\mathbb{Z}}$). For instance, while

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

holds, neither

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

nor

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

holds.

Consider the following formula defined over the theory of integers ($T_{\mathbb{Z}}$) and the theory of uninterpreted functions with equality (T_{E}):

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

We can see that this formula is unsatisfiable since x is either 1 or 2 but $f(x) \neq 1 \wedge f(x) \neq 2$ which means that x has to be different than 1 and 2. However, if we apply the Nelson-Oppera procedure described in the previous section we will **incorrectly** conclude that φ is satisfiable:

Purification	
φ' ($T_{\mathbb{Z}}$)	φ'' (T_{E})
$1 \leq x \wedge$ $x \leq 2 \wedge$ $z = 1$ $w = 2$	$f(x) \neq f(z)$ $f(x) \neq f(w)$
Equality propagation	
sat	sat

4 DPLL(T) framework

The Nelson-Oppera procedure allows us to solve conjunctive first-order theories. To handle disjunction, we could convert the formula to Disjunctive Normal Form (DNF). However, this conversion is usually too expensive and is not the most efficient way of solving disjunctive first-order theories. In Lecture 13 we covered SAT Solvers & DPLL and one of the strengths of the DPLL algorithm is its ability to handle disjunctions. DPLL can be extended into a DPLL(T) framework which allows Satisfiability Modulo Theory (SMT) solvers to handle disjunctions of first-order theories and forms the baseline of modern SMT solvers.

The key idea behind this framework is to decompose the SMT problem into parts we can deal with efficiently:

- Use SAT solver to cope with the **Boolean structure** of the formula;
- Use dedicate conjunctive **theory solver** to decide satisfiability in the background theory.

4.1 Boolean abstraction

We define the Boolean abstraction of a Σ-formula φ recursively:

- <literal> ::= <atom>$_T$ | $\neg$ <atom>$_T$
- <formula> ::= <literal> $\mathcal{B}\,(l_T)\overset{\text{def}}{=}P_i$, where P_i is a fresh variable
- <formula> ::= $\neg$ <formula> $\mathcal{B}\,(\neg F)\overset{\text{def}}{=}\neg\mathcal{B}(F)$
- <formula> ::= <formula> $\wedge$ <formula> $\mathcal{B}\,(F_1 \wedge F_2)\overset{\text{def}}{=}\mathcal{B}(F_1) \wedge \mathcal{B}(F_2)$
- <formula> ::= <formula> $\vee$ <formula> $\mathcal{B}\,(F_1 \vee F_2)\overset{\text{def}}{=}\mathcal{B}(F_1) \vee \mathcal{B}(F_2)$
- <formula> ::= <formula>$\rightarrow$ <formula> $\mathcal{B}\,(F_1 \rightarrow F_2)\overset{\text{def}}{=}\mathcal{B}(F_1) \rightarrow \mathcal{B}(F_2)$
- <formula> ::= <formula>$\leftrightarrow$ <formula> $\mathcal{B}\,(F_1 \leftrightarrow F_2)\overset{\text{def}}{=}\mathcal{B}(F_1) \leftrightarrow \mathcal{B}(F_2)$

Given a Σ-formula φ:

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

The Boolean abstraction of φ is the following:

$$\begin{aligned}
\mathcal{B}(F) &= \mathcal{B}(g(a) = c) \wedge \mathcal{B}(f(g(a)) \neq f(c) \vee g(a) = d) \wedge c \neq d) \\
&= \mathcal{B}(g(a) = c) \wedge \mathcal{B}(f(g(a)) \neq f(c) \vee g(a) = d)) \wedge \mathcal{B}(c \neq d) \\
&= \mathcal{B}(g(a) = c) \wedge \mathcal{B}(f(g(a)) \neq f(c)) \vee \mathcal{B}(g(a) = d) \wedge \mathcal{B}(c \neq d) \\
&= P_1 \wedge (\neg P_2 \vee P_3) \wedge \neg P_4
\end{aligned}$$

Note that we can also define $\mathcal{B}^{-1}$ which maps from the Boolean variables back to the atoms in the original formula. For example $\mathcal{B}^-1(P_1 \wedge P_3 \wedge P_4)$ corresponds to the formula $g(a) = c \wedge g(a) = d \wedge c = d$.

We call $\mathcal{B}(\varphi)$ an abstraction of φ since it is an over-approximation of φ with respect to satisfiability. Observe the following properties of this over-approximation:

- If φ is satisfiable then $\mathcal{B}(\varphi)$ is also satisfiable;
- If $\mathcal{B}(\varphi)$ is satisfiable then φ is not necessarily satisfiable:

  $$\varphi : 1 \leq x \wedge x \leq 2 \wedge f(x) \neq f(1) \wedge f(x) \neq f(2)$$

  φ is unsatisfiable in the theory of integers ($T_{\mathbb{Z}}$) since x is either 1 or 2 but $f(x) \neq f(1) \wedge f(x) \neq f(2)$ implies that x must be different than 1 and 2. However, the Boolean abstraction $\mathcal{B}(\varphi) = P_1 \wedge P_2 \wedge P_3 \wedge P_4$ is satisfiable.

- If φ is unsatisfiable then $\mathcal{B}(\varphi)$ is not necessarily unsatisfiable:

  $$\varphi : 1 \leq x \land x \leq 2 \land f(x) \neq f(1) \land f(x) \neq f(2)$$

  The same example as for the previous case holds for this case as well. φ is unsatisfiable in the theory of integers ($T_{\mathbb{Z}}$) but $\mathcal{B}(\varphi)$ is satisfiable.

- If $\mathcal{B}(\varphi)$ is unsatisfiable then φ is also unsatisfiable.

4.2 Combining theory and SAT solvers

The Boolean abstraction provides us with a **lazy** way to solve SMT. Given a Σ-formula φ, we can determine its satisfiability by performing the following procedure:

1. Construct the Boolean abstraction $\mathcal{B}(\varphi)$;
2. If $\mathcal{B}(\varphi)$ is unsatisfiable then φ is unsatisfiable;
3. Otherwise, get an interpretation I for $\mathcal{B}(\varphi)$;
4. Construct $\omega = \bigwedge_{i=1}^{n} P_i \leftrightarrow I(P_i)$;
5. Send $B^{-1}(\omega)$ to the T-solver;
6. If T-solver reports that $\varphi \cup B^{-1}(\omega)$ is satisfiable then φ is satisfiable;
7. Otherwise, update $\mathcal{B}(\varphi) := \mathcal{B}(\varphi) \land \neg\omega$ and return to step 2.

This procedure terminates when: (i) $\mathcal{B}(\varphi)$ becomes unsatisfiable which implies that φ is also unsatisfiable or (ii) T-solver reports that $\varphi \cup B^{-1}(\omega)$ is satisfiable which implies that $\mathcal{B}(\varphi)$ is satisfiable and that there exists an interpretation I that satisfies all axioms in the theory T. Note that if $\varphi \cup B^{-1}(\omega)$ is unsatisfiable we cannot terminate since there may be another interpretation to $\mathcal{B}(\varphi)$ that would make $\varphi \cup B^{-1}(\omega)$ satisfiable. Therefore, we need to exhaust all interpretations for $\mathcal{B}(\varphi)$ before deciding that φ is unsatisfiable. On step 7 we add $\neg\omega$ to $\mathcal{B}(\varphi)$ since if we did not, we would get the same interpretation I for $\mathcal{B}(\varphi)$. We denote $\neg\omega$ as a **theory conflict clause** that prevents the SAT solver from going down the same path in future iterations.

Suppose we want to find if the Σ-formula φ is satisfiable:

$$\varphi : g(a) = c \land (f(g(a)) \neq f(c) \lor g(a) = d) \land c \neq d$$

We start by building its Boolean abstraction $\mathcal{B}(\varphi)$:

$$\mathcal{B}(\varphi) : P_1 \land (\neg P_2 \lor P_3) \land \neg P_4$$

Table 1 shows the step 1 of the procedure with φ and the corresponding Boolean abstraction $\mathcal{B}(\varphi)$. Next, we query the SAT solver for an interpretation to $\mathcal{B}(\varphi)$. Assume that the SAT solver returns the following interpretation $I = \{P_1, \neg P_2, P_3, \neg P_4\}$. We

Theory solver	SAT solver
$g(a) = c \wedge$ $(f(g(a)) \neq f(c) \vee g(a) = d) \wedge$ $c \neq d$	$P_1 \wedge (\neg P_2 \vee P_3) \wedge \neg P_4$

Table 1: φ and $\mathcal{B}(\varphi)$.

Theory solver	SAT solver
$g(a) = c \wedge$ $(f(g(a)) \neq f(c) \vee g(a) = d) \wedge$ $c \neq d$	$P_1 \wedge (\neg P_2 \vee P_3) \wedge \neg P_4$ $(\neg P_1 \vee P_2 \vee \neg P_3 \vee P_4)$

Table 2: Updated $\mathcal{B}(\varphi)$ after checking that the interpretation $I = \{P_1, \neg P_2, P_3, \neg P_4\}$ does not satisfy φ

construct $\omega = (P_1 \wedge \neg P_2 \wedge P_3 \wedge \neg P_4)$ and send $\mathcal{B}^{-1}(\omega)$ to T-solver. Note that $\mathcal{B}^{-1}(\omega)$ corresponds to:

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

$\mathcal{B}^{-1}(\omega) \cup \varphi$ is unsatisfiable since if $g(a) = d$ and $g(a) = c$ then $c = d$ but φ states that $c \neq d$. Therefore, we know that this interpretation is not satisfiable but there may exist another interpretation I that satisfies φ. We update $\mathcal{B}(\varphi)$ with $\neg\omega$ as shown in Table 2 and query the SAT solver for another interpretation.

Assume that the SAT solver returns a new interpretation $I = \{P_1, P_2, P_3, \neg P_4\}$. We construct $\omega = (P_1 \wedge P_2 \wedge P_3 \wedge \neg P_4)$ and send $\mathcal{B}^{-1}(\omega)$ to T-solver. Note that in this case $\mathcal{B}^{-1}$ corresponds to:

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

We can see that $\mathcal{B}^{-1}(\omega) \cup \varphi$ is unsatisfiable for the same reason as before. We update $\mathcal{B}(\varphi)$ with $\neg\omega$ as shown in Table 3 and perform another query to the SAT solver.

Assume that the SAT solver returns a new interpretation $I = \{P_1, \neg P_2, \neg P_3, \neg P_4\}$. We construct $\omega = (P_1 \wedge \neg P_2 \wedge \neg P_3 \wedge \neg P_4)$ and send $\mathcal{B}^{-1}(\omega)$ to T-solver. Note that in this case $\mathcal{B}^{-1}$ corresponds to:

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

Theory solver	SAT solver
$g(a) = c \wedge$ $(f(g(a)) \neq f(c) \vee g(a) = d) \wedge$ $c \neq d$	$P_1 \wedge (\neg P_2 \vee P_3) \wedge \neg P_4$ $(\neg P_1 \vee P_2 \vee \neg P_3 \vee P_4)$ $(\neg P_1 \vee \neg P_2 \vee \neg P_3 \vee P_4)$

Table 3: Updated $\mathcal{B}(\varphi)$ after checking that the interpretation $I = \{P_1, P_2, P_3, \neg P_4\}$ does not satisfy φ.

Theory solver	SAT solver
$g(a) = c \wedge$	$P_1 \wedge (\neg P_2 \vee P_3) \wedge \neg P_4$
$(f(g(a)) \neq f(c) \vee g(a) = d) \wedge$	$(\neg P_1 \vee P_2 \vee \neg P_3 \vee P_4)$
$c \neq d$	$(\neg P_1 \vee \neg P_2 \vee \neg P_3 \vee P_4)$
	$(\neg P_1 \vee P_2 \vee P_3 \vee P_4)$
	unsat

Table 4: Updated $\mathcal{B}(\varphi)$ after checking that the interpretation $I = \{P_1, \neg P_2, \neg P_3, \neg P_4\}$ does not satisfy φ. $\mathcal{B}(\varphi)$ becomes unsatisfiable after adding the negation of I.

We can see that $\mathcal{B}^{-1}(\omega) \cup \varphi$ is unsatisfiable since $g(a) = c$ but $f(g(a)) \neq f(c)$. We update $\mathcal{B}(\varphi)$ with $\neg\omega$ as shown in Table 4 and observe that $\mathcal{B}(\varphi)$ becomes unsatisfiable after adding $\neg\omega$. Since $\mathcal{B}(\varphi)$ is unsatisfiable, we can conclude that φ is also unsatisfiable.

4.3 Improving DPLL(T) framework

Consider the Σ-formula φ defined over $T_{\mathbb{Z}}$:

$$\varphi : 0 < x \wedge x < 1 \wedge x < 2 \wedge \ldots x < 99$$

The Boolean abstraction $\mathcal{B}(\varphi)$ is the following:

$$\mathcal{B}(\varphi) : P_0 \wedge P_1 \wedge \ldots \wedge P_{99}$$

Note that $\mathcal{B}(\varphi)$ has 2^{98} interpretations containing $P_0 \wedge P_1$ and none of them satisfies φ. The procedure described in the previous section will enumerate all of them one by one and add a blocking conflict clause that only covers a single assignment! A potential solution to this issue is to not treat the SAT solver as a black box but instead incrementally query the theory solver as interpretations are made in the SAT solver. If we would perform this integration then we would be able to stop after adding $\{0 < x, x < 1\}$ and would not need to explore the 2^{98} infeasible interpretations. This can be done by pushing the T-solver into the DPLL algorithm as follows:

1. After Boolean Constraint Propagation (BCP), invoke the T-solver on the partial interpretation;
2. If the T-solver returns unsatisfiable then we can stop the search of the SAT solver and immediately add $\neg\omega$ to $\mathcal{B}\varphi$;
3. Otherwise, continue as usual until we have a new partial interpretation.

Recall the example:

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

And its Boolean abstraction $\mathcal{B}(\varphi)$:

$$\mathcal{B}(\varphi) : P_1 \wedge (\neg P_2 \vee P_3) \wedge \neg P_4$$

DPLL with being by propagating P_1 and $\neg P_4$ since they are unit clauses. At this point the theory axioms imply more propagations:

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

Deciding $\neg P_2$ or P_3 would be wasteful, so we can add the **theory lemmas**:

$$(P_1 \to P_2)$$
$$(P_1 \wedge \neg P_3) \to \neg P_3$$

This procedure is called **theory propagation** and can guarantee that every Boolean interpretation is T-satisfiable. However, in practice doing this at every step can be expensive and theory propagation is only applied when it is "likely" (using heuristics) to derive useful implications.

Another optimization that can be performed is to minimize the conflict clause ω that we add to $\mathcal{B}(\varphi)$ to contain only the root cause of the issue. Consider again the Σ-formula φ:

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

Notice that the interpretations $I = \{P_1, \neg P_2, P_3, \neg P_4\}$ and $I' = \{P_1, \neg P_2, P_3, \neg P_4\}$ had the same root cause that lead to φ being unsatisfiable under that interpretation, i.e. $g(a) = d$ and $g(a)$ which implies that $c = d$ but we know that $c \neq d$ which is a contradiction. Can we find the root cause of this issue and learn something stronger than $\omega = (\neg P_1 \vee P_2 \vee \neg P_3 \vee P_4)$? **Yes**, we can minimize ω using unsatisfiable cores!

Definition 8 (Minimal unsatisfiable core)**.** Let φ be an unsatisfiable formula and $\varphi_c \subseteq \varphi$. φ_c is a minimal unsatisfiable core if and only if:

- φ_c is unsatisfiable;
- Removing any element from φ_c makes φ_c satisfiable.

For $I = \{P_1, \neg P_2, P_3, \neg P_4\}$ we have the following $\mathcal{B}^{-1}(\varphi)$:

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

We can compute the minimal unsatisfiable core of $\mathcal{B}^{-1}(\varphi)$ as follows.

1. Drop $g(a) = c$. Is the formula still unsatisfiable? **No!** Then it means this constraint will be part of the minimal unsatisfiable core.
2. Drop $f(g(a)) \neq f(c)$. Is the formula still unsatisfiable? **Yes!** Then it means that we can remove this constraint from the minimal unsatisfiable core.
3. Now we have $g(a) = c \wedge g(a) = d \wedge c \neq d$.
4. Drop $g(a) = d$. Is the formula still unsatisfiable? No, then keep this constraint.
5. Drop $c \neq d$. Is the formula still unsatisfiable? No, then keep this constraint.

We can conclude that our minimal unsatisfiable core is $g(a) = c \wedge g(a) = d \wedge c \neq d$. Therefore, we can learn the clause $\omega' = (\neg P_1 \vee \neg P_3 \vee P_4)$ instead of $\omega = (\neg P_1 \vee P_2 \vee \neg P_3 \vee P_4)$ which would have save one query to the SAT solver in the previous section.

5 Summary

- Nelson-Oppera procedure allow us to decide the satisfiability of a formula that consists of a combination of conjunctive first-order theories.
- Nelson-Oppera procedure is based on two phases:
  1. Purification;
  2. Equality propagation of shared variables.
- The DPLL(T) framework can be used to decide the satisfiability of a formula that consists of a combination of disjunctive first-order theories.
- We can over-approximate a formula using its Boolean abstraction.
- The key ideas behind the DPLL(T) framework is to:
  - Use SAT solver to cope with the Boolean structure of the formula;
  - Use dedicate conjunctive theory solver to decide satisfiability in the background theory.
- The basic DPLL(T) framework can be further improved with:
  - Theory propagation;
  - Minimal unsatisfiable cores.

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.