SMT Solvers

- An SMT (satisfiability modulo theory) solver is a tool that decides satisfiability of formulas in first-order theories
- SMT solvers are generalizations of SAT solvers
- Can think of SMT formula as SAT formula where propositional variables are replaced with formulas in first-order theories
- Common first-order theories SMT solvers reason about:
  - Theory of equality
  - Theory of rationals
  - Theory of integers
  - Theory of bitvectors
  - Theory of arrays
  - Difference logic

Applications of SMT Solvers

- SMT solvers have gained enormous popularity over the last several years – active research topic
- Many applications: software verification, programming languages, test case generation, planning and scheduling, . . .
- Slogan: “Whatever SAT solvers can do, SMT solvers can do better!”

Existing SMT Solvers

- Many existing off-the-shelf SMT solvers:
  - Yices (SRI)
  - Z3 (Microsoft Research)
  - CVC3 (NYU, U Iowa)
  - STP (Stanford)
  - MathSAT (U Trento, Italy)
  - Barcelogic (Catalonia, Spain)
- Annual competition SMT-COMP between solvers; tools ranked in various categories
- All of these SMT solvers have many users
- For instance, at Microsoft, there are at least two dozen projects that rely on the Z3 SMT solver

Overview

- Plan for today: Learn about how SMT solvers actually work
- Already know how to decide satisfiability of several qff first-order theories (theory of equality, theory of rationals, theory of integers)
- Also already know how to combine these theories using Nelson-Oppen technique
- Big missing piece: How to handle boolean structure of SMT formulas including disjunctions

Motivation for DPLL(T)

- So far, decided satisfiability of first-order theories by converting to DNF
- In reality, this is impractical – why?
- Need a way to decide satisfiability of SMT formulas without conversion to DNF
Main Idea of DPLL(T)

- Key idea underlying SMT solvers: Combine DPLL algorithm for SAT solving with theory solvers
- Theory solver: Decision procedure for checking satisfiability in conjunctive fragment
- This architecture where we combine DPLL-based SAT solvers with theory solvers is known as DPLL(T) framework
- Called DPLL(T) because we combine DPLL algorithm with solver for theory $T$
- However, $T$ can be a combination theory, such as $T_m \cup T_Z$
- As before, solver for $T_m \cup T_Z$ is obtained by using Nelson-Oppen technique

Example

- What is the boolean abstraction of this formula?
  
  $$F : x = z \land ((y = z \land x \neq z) \lor \neg(x = z))$$

- Boolean abstraction is also called boolean skeleton
- Since $B$ is a bijective function, $B^{-1}$ also exists
- What is $B^{-1}(b_2 \lor \neg b_1)$?

SMT Formulas and Boolean Abstraction

- SMT formula in theory $T$ formed according to CFG:
  
  $$F := a_i^T \mid F_1 \land F_2 \mid F_1 \lor F_2 \mid \neg F$$

- For each SMT formula, define a bijective function $B$, called boolean abstraction function, that maps SMT formula to overapproximate SAT formula
- Function $B$ defined inductively as follows:
  
  $$B(a_i^T) = b_i$$
  $$B(F_1 \land F_2) = B(F_1) \land B(F_2)$$
  $$B(F_1 \lor F_2) = B(F_1) \lor B(F_2)$$
  $$B(\neg F) = \neg B(F_1)$$

Boolean Abstraction as Overapproximation

- Observe: The boolean abstraction constructed this way overapproximates satisfiability of the formula
- Is this formula satisfiable?
  
  $$F : x = z \land ((y = z \land x \neq z) \lor \neg(x = z))$$

- Boolean abstraction: $B(F) = b_1 \land ((b_2 \land b_3) \lor \neg b_1)$
- Is this satisfiable?
- What is a sat assignment?
- What is $B^{-1}(A)$?
SMT Solving: Simplest Version

- In simplest version of SMT solvers, construct boolean abstraction \( B(F) \) of SMT formula \( F \)
- If \( B(F) \) is unsat, return unsat
- If \( B(F) \) is sat, get sat assignment \( A \) (conjunction of propositional literals)
- Construct \( B^{-1}(A) \); this is conjunction of atomic \( T \)-formulas
- Query \( T \)-solver for satisfiability of \( B^{-1}(A) \)

SMT Solving: Simplest Version, cont

- If \( T \)-solver decides \( B^{-1}(A) \) is sat, return SAT
- If \( B^{-1}(A) \) is unsat, does this mean original formula is UNSAT?
- Yes because since \( B^{-1}(A) \) is unsat \( B^{-1}(\neg A) \) is valid; thus, \( F \land B^{-1}(\neg A) \) is equivalent to \( F \)
- Formulas such as \( \neg A \) that are \( T \)-valid are called theory conflict clauses

SMT Solving, Simplest Version: Correctness

- If \( B^{-1}(A) \) is unsat, construct new abstraction as \( B(F) \land \neg A \)
- Does \( B(F) \land \neg A \) still overapproximate satisfiability?
- Yes because since \( B^{-1}(A) \) is unsat \( B^{-1}(\neg A) \) is valid; thus, \( F \land B^{-1}(\neg A) \) is equivalent to \( F \)
- Formulas such as \( \neg A \) that are \( T \)-valid are called theory conflict clauses

Example

- Consider example from before:
  \[
  F: \quad x = z \land ((y = z \land x \neq z) \lor \neg(x = z))
  \]
  \[
  B(F): \quad b_1 \land (b_2 \land b_3) \lor \neg b_1
  \]
- Sat assignment to \( B(F) \)
  \[
  A: \quad b_1 \land b_2 \land b_1
  \]
- \( B^{-1}(A) \) is unsat
- What is new boolean abstraction?
- Is this formula SAT?

Shortcoming of This Approach

- So far, we just add negation of current assignment as theory conflict clause, but such conflict clauses are too weak
- Suppose \( A \) is a conjunction of 100 literals such that
  \[
  B^{-1}(A) = x = y \land x \neq y \land a_1 \land a_2 \land \ldots \land a_{100}
  \]
- Theory conflict clause \( \neg A \) prevents exact same assignment
- But it doesn’t prevent many other bad assignments involving \( x = y \land x \neq y \) such as:
  \[
  B^{-1}(A) = x = y \land x \neq y \land a_1 \land a_2 \land \ldots \land \neg a_{100}
  \]
- In fact, there are \( 2^{100} \) unsat assignments containing \( x = y \land x \neq y \) but \( \neg A \) prevents only one of them!
SMT solving. Improvement #1

- Suppose SAT solver makes assignment $A$ s.t. $B^{-1}(A)$ is unsat
- Rather than adding $\neg A$ as a conflict clause, better idea is to find an unsatisfiable core of $B^{-1}(A)$
- An unsatisfiable core $C$ of $A$ contains a subset of atoms in $A$ and $B^{-1}(C)$ is still unsatisfiable.
- Ideally, we would like to find the minimal unsatisfiable core
- Minimal unsatisfiable core $C^*$ has property that if you drop any single atom of $C^*$, result is satisfiable
- What is a minimal unsat core of $x = y \land x \neq y$?

Discussion

- This strategy is much better than simplest strategy where we add $B^{-1}(A)$ as theory conflict clause.
- Using simple strategy, we block just one assignment
- Using minimal unsat cores, we block many assignments using one theory conflict clause
- However, our strategy still not ideal because it waits for full assignment to boolean abstraction to generate conflict clause

Computing Minimal Unsat Core

- How can we compute minimal unsat core of conjunctive $T$ formula without modifying theory solver?
- Let $\phi$ be original unsatisfiable conjunct
- Drop one atom from $\phi$, call this $\phi'$
- If $\phi'$ is still unsat, $\phi := \phi'$
- Repeat this for every atom in $\phi$

Example

- Let’s compute minimal unsat core of
  $$\phi: x = y \land f(x) + z = 5 \land f(x) \neq f(y) \land y \leq 3$$
- Drop $x = y$ from $\phi$. Is result unsat?
- Drop $f(x) + z = 5$. Is result unsat?
- New formula: $\phi: x = y \land f(x) \neq f(y) \land y \leq 3$
- Drop $f(x) \neq f(y)$. Is result unsat?
- Finally, drop $y \leq 3$. Is result unsat?
- Minimal unsat core: $x = y \land f(x) \neq f(y)$

SMT Solving Using Unsat Cores

- Given formula $F$, construct boolean abstraction $B(F)$
- Use SAT solver to decide if $B(F)$ is unsat; if so $F$ also unsat
- Otherwise, get satisfying assignment $A$ to $B(F)$
- Query theory solver if $B^{-1}(A)$ is sat; if so $F$ is sat
- Otherwise, compute minimal unsat core $C$ of $B^{-1}(A)$
- Use $\neg C$ as theory conflict clause
- i.e., construct new boolean abstraction as $B(F \land \neg C)$
- Repeat until we decide sat or unsat

Motivation for Integration with DPLL

- Consider very large formula $F$ containing $x = y$ and $x \neq y$ with corresponding boolean variables $b_1$ and $b_2$
- Also, suppose $B(F)$ contains hundreds of boolean variables
- As soon as sat solver makes assignment $b_1 = T, b_2 = T$, we are doomed because this is unsatisfiable in theory
- Thus, no need to continue with SAT solving after this bad partial assignment
- Idea: Don’t use SAT solver as “blackbox”
- Instead, integrate theory solver right into the DPLL algorithm
DPLL-Based SAT Solver Architecture

- **Decide**
  - SAT

- **Analyze Conflict**
  - UNSAT

- **BCP**
  - no conflict
  - backtrack if \( d > 0 \)
  - conflict

**Idea:** Integrate theory solver right into this SAT solving loop!

DPLL(\( T \)) Framework

- **Decide**
  - SAT
  - backtrack if \( d > 0 \)

- **BCP**
  - no conflict, theory propagation lemma(s)

- **Analyze Conflict**
  - UNSAT

**Theory Solve**

- **C(A)**

**Combination of DPLL-based SAT solver and decision procedure for conjunctive \( T \) formula called DPLL(\( T \)) framework**

Theory Propagation Lemmas

- **Idea:** Theory solver can communicate which literals are implied by current partial assignment.

- **In our example,** \( \neg x \neq z \) implied by current partial assignment \( x = y \land y = z \)

- **Thus,** can safely add \( b_1 \land b_2 \rightarrow b_3 \) to clause database

- **These kinds of clauses implied by theory are called theory propagation lemmas**

Inferring Theory Propagation Lemmas

- **How do we obtain theory propagation lemmas?**

- **Option #1:** Treat theory solver as blackbox, query whether particular literal \( a \) is implied by current partial assignment?

- **Option #2:** Modify theory solver so that it can figure out implied literals

- Second option is considered more efficient, but have to figure out how to do this for each different theory

Which Theory Propagation Lemmas to Add

- **Which theory propagation lemmas do we add?**

- **Option #1:** Figure out and add all literals implied by current partial assignment; called exhaustive theory propagation

- **Option #2:** Only figure out literals “obviously” implied by current partial assignment

- Exhaustive theory propagation can be very expensive; second option considered preferable

- Solvers use different strategies to obtain “obvious” implications
Summary

▶ SMT solvers decide satisfiability in boolean combinations of different theories

▶ Instead of converting to DNF, they handle boolean structure using SAT solving techniques

▶ Most common approach is to construct boolean abstraction and lazily infer theory conflict clauses

▶ To do this, can either consider SAT solver as blackbox or can integrate with it

▶ Latter strategy considered superior and known as DPLL(\(T\)) framework