Satisfiability Modulo Theories

IN5110

Gaute Berge   Sondre Lunde

8. November 2019

Contents

Motivation

What can SMT solvers be used for?

- Optimization
- Bounded model checking
- Brute force

Boolean satisfiability problem

- The problem of determining if there exists an interpretation that satisfies a given Boolean formula.
- Well known NP-complete problem

Conjunctive Normal Form

Generally, the formulas are on Conjunctive Normal Form (CNF)

- $p \wedge q \wedge r$
- $p \wedge (q \vee p) \wedge r$
- $p \wedge (q \rightarrow p) \wedge r$ Not CNF, but easily fixed
- $p \wedge (\neg q \vee p) \wedge r$

Clausal form

Formulas as sets of clauses

- $p \wedge (\neg q \vee p) \wedge r$
- $\{\{p\}, \{\neg q, p\}, \{r\}\}$

SAT is fast

- Brute force is slow, but ...
- we often don't have to brute force
- Making SAT faster is heavily researched
- Conflict-Driven Clause Learning
- DPLL
- Utilizing the *shape* of the clauses

Encoding problems in propositional logic is difficult and annoying.

Encoding constraints

- $x < y$
- $x < 5 \land x > 10$
- $f(y) = f(x) \land y \neq x$

Looks a lot like first-order logic

First-order logic

Satisfiability for general FOL is undecidable, however ...

- There are versions of FOL that *are* decidable
  - The set of first-order logical validities in the signature with only equality, established by Leopold Löwenheim in 1915.
  - The first-order theory of the natural numbers in the signature with equality and addition, also called Presburger arithmetic.
- We are often interested in the *expected* interpretation of common symbols
  - $x < y \land \neg(x < y + 0)$

Theories

What is a theory?

- A limited First-Order Logic
- with equality

Syntax

The same as First-Order Logic, but with a single addition:
We can have the equality-symbol between **terms**.

Semantics

Similar to first-order logic, but with restrictions on sorted variables.

$$
\begin{aligned}
&\dots \\
\mathcal{A} \vDash t_1 = t_2 &\text{ iff } t_1^{\mathcal{A}} = t_2^{\mathcal{A}} \\
\mathcal{A} \vDash \exists x : \sigma \phi &\text{ iff } \mathcal{A}[x \rightarrow a] \vDash \phi \text{for some a} \in \mathcal{A}_\sigma \\
&\dots
\end{aligned}
$$

Equality with Uninterpreted Function Symbols

Also known as the **empty theory**, due to imposing no restrictions on its models

Axioms:

1. $\forall x. x = x$ (Reflexivity)
2. $\forall x, y. (x = y) \rightarrow (y = x)$ (Symmetry)
3. $\forall x, y, z. (x = y) \land (y = z) \rightarrow (x = z)$ (Transitivity)

Example

1. $\Phi = \{f(f(a)) = a, f(f(f(a))) = a, g(a) \neq g(f(a))\}$
2. $\{\{a, f(f(a)), f(f(f(a)))\}, \{f(a)\}, \{g(a)\}, \{g(f(a))\}\}$
3. $a = f(f(a)) \implies f(a) = f(f(f(a)))$
4. $\{\{a, f(f(a)), f(f(f(a))), f(a)\}, \{g(a)\}, \{g(f(a))\}\}$
5. $\{\{a, f(f(a)), f(f(f(a))), f(a)\}, \{g(a), g(f(a))\}\}$
6. (unsat)

Example: Power of equality

1. $\{a * (f(b) + f(c)) = d, b * (f(a) + f(c)) \neq d, a = b\}$
2. Change $*$ to h, and $+$ to g
3. $\{h(a, g(f(b), f(c))) = d, h(b, g(f(a), f(c))) \neq d, a = b\}$
4. This can be proved unsat with Congruence Closure

Example theories

- Equality
- Integer arithmetic
- Real arithmetic
- Bitvectors
- Arrays
- Algebraic Data Types

Arithmetic

Signature for Integer Arithmetic:

- A single sort **Z**, the integer number constants
- Function symbols $\Sigma^F = \{+, -, *\}$
- Predicate symbol $\Sigma^P = \{\leq\}$

The signature is paired with the standard model of the integers, the Σ-model that interprets **Z** as the set $\mathbb{Z}$, and the constants and operators in the usual way.

- Integers are decidable if multiplication is restricted. (LIA)
- (Reals are decidable even with multiplication)

Satisfiability Modulo Theories

Example

$$b + 2 = c \wedge f(read(write(a, b, 3), c - 2)) \neq f(c - b + 1)$$

Arithmetic Array theory Uninterpreted Function Symbols By arithmetic, this is equivalent to

$$b + 2 = c \wedge f(read(write(a, b, 3), b)) \neq f(3)$$

then, by the array theory axiom: $read(write(v, i, x), i) = x$

$$b + 2 = c \wedge f(3) \neq f(3)$$

then, the formula is unsatisfiable

Eager

Just convert to SAT
Can lead to an explosion in terms of size.

Lazy offline

Given a formula ϕ

- let ϕ^p be the boolean abstraction of ϕ
- If ϕ^p is unsat, then ϕ is unsat.
- otherwise we have a satisfying assignment μ^p, which is sent to the T-solver
- If μ is T-consistent, then ϕ is T-consistent
- otherwise, $\neg\mu^p$ is added as a clause to ϕ^p and the SAT solver is restarted.

Lazy online

Hybrid approaches of lazy and eager

When to use which? When to combine?

- Best result from combining eager and lazy
- "Mostly Horn-clauses"
- Machine Learning-based heuristics

Applications

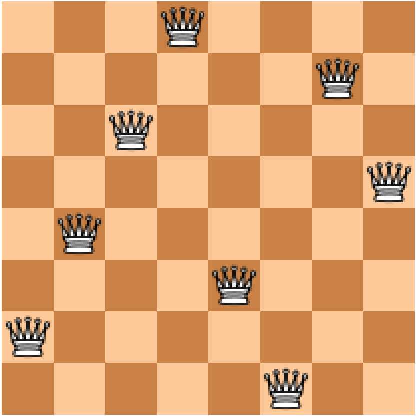
- Security
- Scheduling
- Optimization
- Model Checking
  - Bounded Model Checking

Z3

SMT solver created by Microsoft

- Very fast
- Wide variety of theories
- Bindings to many popular languages (C, Python, Haskell, ++)

Eight Queens



Eight Queens

```
# We know each queen must be in a different row.
# So, we represent each queen by a single integer: the column position
Q = [ Int('Q_%i' % (i + 1)) for i in range(8) ]

# Each queen is in a column {1, ... 8 }
val_c = [ And(1 <= Q[i], Q[i] <= 8) for i in range(8) ]

# At most one queen per column
col_c = [ Distinct(Q) ]

# Diagonal constraint
diag_c = [ If(i == j,
              True,
              And(Q[i] - Q[j] != i - j, Q[i] - Q[j] != j - i))
           for i in range(8) for j in range(i) ]

solve(val_c + col_c + diag_c)
```

1 million USD prize if you can create an AI that solves the 1000x1000 problem, or prove that it is not possible.

Some references

- Barrett, C., et.al. *Satisfiability Modulo Theories* in
  - Biere, A., et.al. (eds.) (2009) *The Handbook of Satisfiability*, IOS Press: Amsterdam
- Barrett, C., et.al. *Satisfiability Modulo Theories* in
  - Clarke, E.M., et.al. (eds.) (2018) *The Handook of Model Checking*, Springer International Publishing
- Kanig, J., (2010) *An Introduction to SMT solvers*, http://www.open-do.org/wp-content/uploads/2010/06/SMT_provers.pdf [Accessed Nov. 7. 2019]
- Griggio, A., (2014) *Introduction to SMT*, http://satsmt2014.forsyte.at/files/2014/07/SMT-introduction.pdf [Accessed Nov. 7. 2019]