Satisfiability Modulo Theories

Viktor Kunčak

Satisfiability Modulo Theories (SMT) Solvers

SMT solvers are the most widely used tools for automation of program verification
Examples of solvers:

- CVC5 (successor of CVC4): `https://github.com/cvc5/cvc5`
- Z3: `https://github.com/Z3Prover/z3`
- Princess: `http://www.philipp.ruemmer.org/princess.shtml`
- Yices: `https://yices.csl.sri.com/`

Competitions of solvers: `https://smt-comp.github.io/2023/`
Tools using them:

- Stainless: `https://github.com/epfl-lara/stainless/`
- Dafny: `https://github.com/dafny-lang/dafny`
- Fstar: `https://www.fstar-lang.org/`
- Viper: `https://www.pm.inf.ethz.ch/research/viper.html`

Satisfiability Modulo Theories

SAT = Satisfiability for Propositional Logic

- formula: $p \wedge (\neg q \vee r) \wedge s$
- Do there exist truth values p, q, r, s that make formula true?

Satisfiability Modulo Theories

SAT = Satisfiability for Propositional Logic

- formula: $p \wedge (\neg q \vee r) \wedge s$
- Do there exist truth values p, q, r, s that make formula true? yes, e.g., $p \mapsto 1, q \mapsto 0, s \mapsto 1$ (r is any)

Satisfiability Modulo Theories

SAT = Satisfiability for Propositional Logic

- formula: $p \wedge (\neg q \vee r) \wedge s$
- Do there exist truth values p, q, r, s that make formula true? yes, e.g., $p \mapsto 1, q \mapsto 0, s \mapsto 1$ (r is any)

SMT = Satisfiability Modulo Theories (e.g. Z3 solver)

- formula: $a = b \wedge (\neg(f(a) = f(b)) \vee b = c) \wedge \neg(f(a) = f(c))$
- Do there exist values of a, b, c that makes formula true?

Satisfiability Modulo Theories

SAT = Satisfiability for Propositional Logic

- formula: $p \land (\neg q \lor r) \land s$
- Do there exist truth values p, q, r, s that make formula true? yes, e.g., $p \mapsto 1, q \mapsto 0, s \mapsto 1$ (r is any)

SMT = Satisfiability Modulo Theories (e.g. Z3 solver)

- formula: $a = b \land (\neg(f(a) = f(b)) \lor b = c) \land \neg(f(a) = f(c))$
- Do there exist values of a, b, c that makes formula true? No. We have: $a = b$, $f(a) = f(b)$, $b = c$, $a = c$, $f(a) = f(c)$, $\neg(f(a) = f(c))$ - unsat

Satisfiability Modulo Theories

SAT = Satisfiability for Propositional Logic

- formula: $p \land (\neg q \lor r) \land s$
- Do there exist truth values p, q, r, s that make formula true? yes, e.g., $p \mapsto 1, q \mapsto 0, s \mapsto 1$ (r is any)

SMT = Satisfiability Modulo Theories (e.g. Z3 solver)

- formula: $a = b \land (\neg(f(a) = f(b)) \lor b = c) \land \neg(f(a) = f(c))$
- Do there exist values of a, b, c that makes formula true? No. We have: $a = b$, $f(a) = f(b)$, $b = c$, $a = c$, $f(a) = f(c)$, $\neg(f(a) = f(c))$ - unsat
- Another example: $x = y \land f(x) < f(y)$

Satisfiability Modulo Theories

SAT = Satisfiability for Propositional Logic

- formula: $p \wedge (\neg q \vee r) \wedge s$
- Do there exist truth values p, q, r, s that make formula true? yes, e.g., $p \mapsto 1, q \mapsto 0, s \mapsto 1$ (r is any)

SMT = Satisfiability Modulo Theories (e.g. Z3 solver)

- formula: $a = b \wedge (\neg(f(a) = f(b)) \vee b = c) \wedge \neg(f(a) = f(c))$
- Do there exist values of a, b, c that makes formula true? No. We have: $a = b$, $f(a) = f(b)$, $b = c$, $a = c$, $f(a) = f(c)$, $\neg(f(a) = f(c))$ - unsat
- Another example: $x = y \wedge f(x) < f(y)$ - unsat

Satisfiability Modulo Theories

SAT = Satisfiability for Propositional Logic

- formula: $p \wedge (\neg q \vee r) \wedge s$
- Do there exist truth values p, q, r, s that make formula true? yes, e.g., $p \mapsto 1, q \mapsto 0, s \mapsto 1$ (r is any)

SMT = Satisfiability Modulo Theories (e.g. Z3 solver)

- formula: $a = b \wedge (\neg(f(a) = f(b)) \vee b = c) \wedge \neg(f(a) = f(c))$
- Do there exist values of a, b, c that makes formula true? No. We have: $a = b$, $f(a) = f(b)$, $b = c$, $a = c$, $f(a) = f(c)$, $\neg(f(a) = f(c))$ - unsat
- Another example: $x = y \wedge f(x) < f(y)$ - unsat
- Another example: $1 \leq x \wedge x \leq 2 \wedge f(x) \neq f(1)$

Satisfiability Modulo Theories

SAT = Satisfiability for Propositional Logic

- formula: $p \land (\neg q \lor r) \land s$
- Do there exist truth values p, q, r, s that make formula true? yes, e.g., $p \mapsto 1, q \mapsto 0, s \mapsto 1$ (r is any)

SMT = Satisfiability Modulo Theories (e.g. Z3 solver)

- formula: $a = b \land (\neg(f(a) = f(b)) \lor b = c) \land \neg(f(a) = f(c))$
- Do there exist values of a, b, c that makes formula true? No. We have: $a = b$, $f(a) = f(b)$, $b = c$, $a = c$, $f(a) = f(c)$, $\neg(f(a) = f(c))$ - unsat
- Another example: $x = y \land f(x) < f(y)$ - unsat
- Another example: $1 \leq x \land x \leq 2 \land f(x) \neq f(1)$: $x \mapsto 2$

Satisfiability Modulo Theories

SAT = Satisfiability for Propositional Logic

- formula: $p \wedge (\neg q \vee r) \wedge s$
- Do there exist truth values p, q, r, s that make formula true? yes, e.g., $p \mapsto 1, q \mapsto 0, s \mapsto 1$ (r is any)

SMT = Satisfiability Modulo Theories (e.g. Z3 solver)

- formula: $a = b \wedge (\neg(f(a) = f(b)) \vee b = c) \wedge \neg(f(a) = f(c))$
- Do there exist values of a, b, c that makes formula true? No. We have: $a = b$, $f(a) = f(b)$, $b = c$, $a = c$, $f(a) = f(c)$, $\neg(f(a) = f(c))$ - unsat
- Another example: $x = y \wedge f(x) < f(y)$ - unsat
- Another example: $1 \leq x \wedge x \leq 2 \wedge f(x) \neq f(1)$: $x \mapsto 2$

Large formulas with few no or few quantifiers (unlike pure FOL provers)

- propositional structure explored using SAT solver
- function and relation symbols come from decidable theories (quantifier-free linear arithmetic, algebraic data types)
- atomic formulas solved using decision procedures (theory solvers)
- quantifiers handled mostly by instantiation

Flattening and Extracting Propositional Structure

$$a = b \wedge (f(a) \neq f(b) \vee b = c) \wedge f(a) \neq f(c)$$

for each atomic formula introduce propositional variable:

Flattening and Extracting Propositional Structure

$$a = b \wedge (f(a) \neq f(b) \vee b = c) \wedge f(a) \neq f(c)$$

for each atomic formula introduce propositional variable:

$$\begin{array}{l} p \wedge (\neg q \vee r) \wedge \neg s \\ p \Leftrightarrow a = b \\ q \Leftrightarrow f(a) = f(b) \\ r \Leftrightarrow b = c \\ s \Leftrightarrow f(a) = f(c) \end{array}$$

flatten: give name to each subterm, e.g. f_a denotes $f(a)$:

Flattening and Extracting Propositional Structure

$$a = b \wedge (f(a) \neq f(b) \vee b = c) \wedge f(a) \neq f(c)$$

for each atomic formula introduce propositional variable:

$$\begin{array}{l} p \wedge (\neg q \vee r) \wedge \neg s \\ p \Leftrightarrow a = b \\ q \Leftrightarrow f(a) = f(b) \\ r \Leftrightarrow b = c \\ s \Leftrightarrow f(a) = f(c) \end{array}$$

flatten: give name to each subterm, e.g. f_a denotes $f(a)$:

$p \wedge (\neg q \vee r) \wedge \neg s$ give to SAT solver, who returns e.g. $p \wedge \neg q \wedge \neg s$

Flattening and Extracting Propositional Structure

$$a = b \wedge (f(a) \neq f(b) \vee b = c) \wedge f(a) \neq f(c)$$

for each atomic formula introduce propositional variable:

$$\begin{array}{l} p \wedge (\neg q \vee r) \wedge \neg s \\ p \Leftrightarrow a = b \\ q \Leftrightarrow f(a) = f(b) \\ r \Leftrightarrow b = c \\ s \Leftrightarrow f(a) = f(c) \end{array}$$

flatten: give name to each subterm, e.g. f_a denotes $f(a)$:

$p \wedge (\neg q \vee r) \wedge \neg s$ give to SAT solver, who returns e.g. $p \wedge \neg q \wedge \neg s$

$$\left.\begin{array}{l} p \Leftrightarrow a = b \\ q \Leftrightarrow f_a = f_b \\ r \Leftrightarrow b = c \\ s \Leftrightarrow f_a = f_c \end{array}\right\} \quad \begin{array}{l} \text{maps prop. assignment to conjunction of literals} \\ a = b \wedge f_a \neq f_b \wedge f_a \neq f_c \end{array}$$

Flattening and Extracting Propositional Structure

$$a = b \wedge (f(a) \neq f(b) \vee b = c) \wedge f(a) \neq f(c)$$

for each atomic formula introduce propositional variable:

$$\begin{array}{l} p \wedge (\neg q \vee r) \wedge \neg s \\ p \Leftrightarrow a = b \\ q \Leftrightarrow f(a) = f(b) \\ r \Leftrightarrow b = c \\ s \Leftrightarrow f(a) = f(c) \end{array}$$

flatten: give name to each subterm, e.g. f_a denotes $f(a)$:

$p \wedge (\neg q \vee r) \wedge \neg s$ give to SAT solver, who returns e.g. $p \wedge \neg q \wedge \neg s$

$$\left.\begin{array}{l} p \Leftrightarrow a = b \\ q \Leftrightarrow f_a = f_b \\ r \Leftrightarrow b = c \\ s \Leftrightarrow f_a = f_c \end{array}\right\} \quad \begin{array}{l} \text{maps prop. assignment to conjunction of literals} \\ a = b \wedge f_a \neq f_b \wedge f_a \neq f_c \end{array}$$

$$\left.\begin{array}{l} f_a = f(a) \\ f_b = f(b) \\ f_c = f(c) \end{array}\right\} \quad \begin{array}{l} \text{each theory uses its conjuncts and definitions} \\ \ldots \wedge a = b \wedge f_a \neq f_b \wedge f_a \neq f_c \\ \text{UNSAT, give to SAT solver new clause to conjoin} : \neg(p \wedge \neg q \wedge \neg s) \end{array}$$

Formula containing function symbols and arithmetic

- f is **uninterpreted symbol** (as in FOL)
- $+, <, \leq, 1, 3, 5$ are as in linear integer arithmetic; x is of type integer

$$\underbrace{1 \leq x}_{p} \wedge \underbrace{x < 3}_{q} \wedge \Big(\big(\underbrace{f(1) + 1 \leq f(x)}_{r} \wedge \underbrace{f(x) < f(2)}_{s} \big) \vee \underbrace{4 = 2x}_{t} \Big)$$

Formula containing function symbols and arithmetic

- f is **uninterpreted symbol** (as in FOL)
- $+, <, \leq, 1, 3, 5$ are as in linear integer arithmetic; x is of type integer

$$\underbrace{1 \leq x}_{p} \wedge \underbrace{x < 3}_{q} \wedge \Big(\big(\underbrace{f(1) + 1 \leq f(x)}_{r} \wedge \underbrace{f(x) < f(2)}_{s} \big) \vee \underbrace{4 = 2x}_{t} \Big)$$

$$\begin{aligned}
&p \wedge q \wedge ((r \wedge s) \vee t) \wedge \\
&\quad p \Leftrightarrow 1 \leq x \ \wedge \\
&\quad q \Leftrightarrow x < 3 \ \wedge \\
&\quad r \Leftrightarrow u_1 \leq u_2 \ \wedge \quad u_1 = u_3 + 1 \wedge u_3 = f(u_4) \wedge u_2 = f(x) \wedge u_4 = 1 \\
&\quad s \Leftrightarrow u_2 < u_5 \ \wedge \quad u_5 = f(u_6) \wedge u_6 = 2 \\
&\quad t \Leftrightarrow u_7 = u_8 \ \wedge \quad u_7 = 4 \wedge u_8 = 2x
\end{aligned}$$

Formula containing function symbols and arithmetic

- f is **uninterpreted symbol** (as in FOL)
- $+, <, \leq, 1, 3, 5$ are as in linear integer arithmetic; x is of type integer

$$\underbrace{1 \leq x}_{p} \wedge \underbrace{x < 3}_{q} \wedge \Big(\big(\underbrace{f(1) + 1 \leq f(x)}_{r} \wedge \underbrace{f(x) < f(2)}_{s} \big) \vee \underbrace{4 = 2x}_{t} \Big)$$

$$\begin{aligned}
&p \wedge q \wedge ((r \wedge s) \vee t) \wedge \\
&\quad p \Leftrightarrow 1 \leq x \ \wedge \\
&\quad q \Leftrightarrow x < 3 \ \wedge \\
&\quad r \Leftrightarrow u_1 \leq u_2 \ \wedge \quad u_1 = u_3 + 1 \wedge u_3 = f(u_4) \wedge u_2 = f(x) \wedge u_4 = 1 \\
&\quad s \Leftrightarrow u_2 < u_5 \ \wedge \quad u_5 = f(u_6) \wedge u_6 = 2 \\
&\quad t \Leftrightarrow u_7 = u_8 \ \wedge \quad u_7 = 4 \wedge u_8 = 2x
\end{aligned}$$

Who handles which part in this example:

propositional formula	SAT solver
pure equalities ($u_7 = u_8$)	both theory solvers
highlighted formulas	solver for theory of uninterpreted functions
remaining ones	solver for theory of integer linear arithmetic

Theory of Uninterpreted Function Symbols

Quantifier-free first-order logic with equality
Assume it is interpreted over an infinite domain
Assume no relation symbols: replace $R(t_1,\ldots,t_n)$ with $f_R(t_1,\ldots,t_n) = T$ for some fresh constant T
SAT solver handles disjunctions: assume conjunction of equalities and disequalities
Key inference rule, for each function symbol f of n arguments:

$$\frac{t_1 = t_1' \quad \ldots \quad t_n = t_n'}{f(t_1,\ldots,t_n) = f(t_1',\ldots,t_n')}$$

Also: "$=$" is equivalence relation and $t \neq t$ is contradictory
Apply these rules only to those terms that occur in the formula
Implementation: E-graph stores congruence relation computed so far.
Applying rules: merging nodes in this graph

Example of Running the Algorithm

Let $f^k(a)$ denote $f(\dots f(a)\dots)$ with k-fold application of f. Consider

$$f^3(a) = a \wedge f^5(a) = a \wedge f^2(a) \neq a$$

Apply the congruence closure algorithm to check its satisfiability.

Initial graph of all ground terms and the given equalities:

$$\begin{tikzcd}
a \arrow[r, "f"] \arrow[rrr, bend right, dashed, no head] \arrow[rrrrr, bend right, dashed, no head] & f(a) \arrow[r, "f"] & f^2(a)) \arrow[r, "f"] & f^3(a)) \arrow[r, "f"] & f^4(a) \arrow[r, "f"] & f^5(a)
\end{tikzcd}$$

Congruence rule

in this case: $x = y \longrightarrow f(x) = f(y)$

Equivalence maintained using union-find algorithm

Conjunction is unsatisfiable $\Leftrightarrow$ there is literal $t_1 \neq t_2$ where t_1, t_2 are merged

$\Leftarrow$): by properties of equality, conclusions are sound

$\Rightarrow$): contrapositive: congruence extends to congruence on the Herbrand model

Decision Procedures are Building Blocks of SMT Solvers

Decision procedures:

- uninterpreted functions
- algebraic data types
- rational linear arithmetic
- integer linear arithmetic

Many other decidable theories exist (e.g. sets with cardinality bounds)
Tarski has shown that there exists a quantifier elimination algorithm for conjunction of polynomials over reals and over complex numbers.
Later method: Cylindrical Algebraic Decomposition (CUD), used in computer algebra systems

Versions of Z3 and CVC4 have implementations of complete procedures for reals:
Deciding the Consistency of Non-Linear Real Arithmetic Constraints with a Conflict Driven Search Using Cylindrical Algebraic Coverings

Over integers, the problem is undecidable: 10th Hilbert's problem (Y. Matiyasevich)

Quantifier Instantiation During SMT Solving Process

$$G \land \forall x. F(x) \quad \rightsquigarrow \quad G \land F(t) \land \forall x. F(x)$$

where t is a term occurring in G

- this can go on forever
- in general this is incomplete: may need to invent terms that do not occur
- even in the limit it is not complete with respect to the ideal semantics of e.g. integers (theory of quantified integers is not even enumerable)

Controlling the instantiation process using **triggers**

- for each quantified formula $\forall \bar{x}. F(\bar{x})$ require a pattern $P(\bar{x})$ that contains all free variables in $F(\bar{x})$
- instantiate $F(\bar{x})$ only if the the pattern $P(x)$ occurs in the ground formula so far
- introduced in Simplify: a theorem prover for program checking

More information in these papers

- Solving Quantified Verification Conditions using Satisfiability Modulo Theories
- Efficient E-matching for SMT solvers