A Survey of Satisfiability Modulo Theory

David Monniaux

VERIMAG

GdR GPL Grenoble, 2018-06-14

SMT = SAT + theories

SAT = say whether a formula over Booleans is satisfiable (and give a model if so)

SMT = say whether a formula over Booleans **and other types** is satisfiable (and give a model if so)

$(x \leq 0 \lor x + y \leq 0) \land y \geq 1 \land x \geq 1$ unsatisfiable for $x, y \in \mathbb{R}$

$(x \leq 0 \lor x + y \leq 0) \land y \geq 1$ satisfiable for $x, y \in \mathbb{Z}$

Here theory = **linear real arithmetic** (LRA) or **linear integer arithmetic** (LIA)

Contents

DPLL and CDCL

DPLL(T)

Natural domain SMT

- Exponential behaviour of DPLL(T)
- Abstract CDCL (ACDCL)
- Model-construction satisfiability calculus (MCSAT)

Other topics

Conclusion

Propositional satisfiability (SAT)

Input: formula with $\wedge$, $\vee$
(possibly "if then else", "exclusive-or" etc.)

$$\left((a \wedge \bar{b} \wedge \bar{c}) \vee (b \wedge c \wedge \bar{d})\right) \wedge (\bar{b} \vee \bar{c}) .$$

Output: "unsat" or a model (satisfying assignment)

Conjonction normal form (CNF)

View the SAT formula as a system of constraints
= **clauses** (disjunctions of literals a or $\bar{a}$)

convert from arbitrary formula to CNF

cannot be done efficiently keeping only original variables
(exponential blowup, per distributivity)

$$(a \vee b) \wedge (c \vee d) \longrightarrow (a \wedge c) \vee (a \wedge d) \vee (b \wedge c) \vee (b \wedge d)$$

Tseitin encoding

Add extra variables

$$\left((a \land \bar{b} \land \bar{c}) \lor (b \land c \land \bar{d})\right) \land (\bar{b} \lor \bar{c})\,.$$

Assign propositional variables to sub-formulas:

$$\begin{array}{ccc} e \equiv a \land \bar{b} \land \bar{c} & f \equiv b \land c \land \bar{d} & g \equiv e \lor f \\ h \equiv \bar{b} \lor \bar{c} & \phi \equiv g \land h\,; & \end{array}$$

Tseitin encoding

$$e \equiv a \wedge \bar{b} \wedge \bar{c} \qquad f \equiv b \wedge c \wedge \bar{d} \qquad g \equiv e \vee f$$
$$h \equiv \bar{b} \vee \bar{c} \qquad \phi \equiv g \wedge h;$$

turned into clauses

$$\begin{array}{cccc}
\bar{e} \vee a & \bar{e} \vee \bar{b} & \bar{e} \vee \bar{c} & \bar{a} \vee b \vee c \vee e \\
\bar{f} \vee b & \bar{f} \vee c & \bar{f} \vee d & \bar{b} \vee \bar{c} \vee d \vee f \\
\bar{e} \vee g & \bar{f} \vee g & \bar{g} \vee e \vee f & \\
b \vee h & c \vee h & \bar{h} \vee \bar{b} \vee \bar{c} & \\
\bar{\phi} \vee g & \bar{\phi} \vee h & \bar{g} \vee \bar{h} \vee \phi & \phi
\end{array}$$

DPLL

Each clause acts as **propagator** e.g.
assuming a and $\bar{b}$, clause $\bar{a} \vee b \vee c$ yields c

Boolean constraint propagation aka **unit propagation**:
propagate as much as possible
once the value of a variable is known, use it elsewhere

	7	2	3	8	5	4		
	3	9		1	6			
1			2	7		3		6
7	8					6	4	
5								7
	9	4					3	1
4		1		6	3			8
			9	2		1	6	
		8	5	4	1	2	7	

DPLL: Branching

If unit propagation insufficient to

- either find a satisfying assignment
- either find an unsatisfiable clause (all literals forced to false)

Then:

- pick a variable
- do a search subtree for both polarities of the variable

Example

$$\begin{array}{cccc}
\bar{e} \vee a & \bar{e} \vee \bar{b} & \bar{e} \vee \bar{c} & \bar{a} \vee b \vee c \vee e \\
\bar{f} \vee b & \bar{f} \vee c & \bar{f} \vee d & \bar{b} \vee \bar{c} \vee d \vee f \\
\bar{e} \vee g & \bar{f} \vee g & \bar{g} \vee e \vee f & \\
b \vee h & c \vee h & \bar{h} \vee \bar{b} \vee \bar{c} & \\
\bar{\phi} \vee g & \bar{\phi} \vee h & \bar{g} \vee \bar{h} \vee \phi & \phi
\end{array}$$

From unit clause ϕ

$\bar{\phi} \vee g \to g$ $\quad$ $\bar{\phi} \vee h \to h$ $\quad$ $\bar{g} \vee \bar{h} \vee \phi$ removed

Now g and h are **t**,

$\bar{e} \vee g$ removed $\quad$ $\bar{f} \vee g$ removed $\quad$ $b \vee h$ removed

$c \vee h$ removed $\quad$ $\bar{g} \vee e \vee f \to e \vee f$ $\quad$ $\bar{h} \vee \bar{b} \vee \bar{c} \to \bar{b} \vee \bar{c}$

CDCL: clause learning

A DPLL branch gets closed by **contradiction**: a literal gets forced to both **t** and **f**.

Both **t** and **f** inferred from hypotheses H by unit propagation.
Trace back to a subset of hypotheses, sufficient for contradiction.
e.g. $a \wedge \overline{b} \wedge \overline{c} \wedge d \wedge H \implies \mathbf{f}$

Learn clause = negation of bad hypotheses, implies by H:

$$\overline{a} \vee b \vee c \vee \overline{d}$$

Add this clause (maybe garbage-collected later) to H
Used by unit propagation

Proof systems

DPLL Tree resolution

CDCL DAG resolution (shared proof subtrees)
= linear resolution

Some problems have **exponentially smaller proofs** in DAG than tree resolution.

(Independent of search strategy.)

Implementation wise

Clause simplification etc. implemented as
two watched literals per clause

Pointers to clauses used for deduction

Highly optimized proof engines

- Minisat
- Glucose

Preprocessing

Contents

DPLL and CDCL

DPLL(T)

Natural domain SMT
- Exponential behaviour of DPLL(T)
- Abstract CDCL (ACDCL)
- Model-construction satisfiability calculus (MCSAT)

Other topics

Conclusion

DPLL(T)

(Improper terminology, should be CDCL(T))

$(x \leq 0 \vee x + y \leq 0) \wedge y \geq 1 \wedge x \geq 1$
$\downarrow$ dictionary of theory literals
$(a \vee b) \wedge c \wedge d$

Solve, get $(a, b, c, d) = (\mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{t})$.
But $x \leq 0 \wedge x \geq 1$ is a contradiction!
Add **theory lemma** $\bar{a} \vee \bar{d}$

Solve, get $(a, b, c, d) = (\mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{t})$.
But $x + y \leq 0 \wedge \geq 1 \wedge x \geq 1$ is a contradiction!
Add **theory lemma** $\bar{b} \vee \bar{c} \vee \bar{d}$.

The problem is **unsatisfiable**.

DPLL(T)

In practice, do not wait for the CDCL solver to provide a full assignment.
Check partial assignments for theory feasibility.

If during theory processing, a literal becomes known to be **t** or **f**, propagate it to CDCL.
e.g. $x \geq 0$, $x \geq 1$ assigned, propagate $x + y \geq 0$

Boolean relaxation of the original problem.
Lazy expansion of theory.

Linear real arithmetic

Usually decided by exact precision **simplex**.
Extract from the tableau the contradictory subset of assignments.

LRA Example

$$\begin{cases} 2 & \leq 2x + y \\ -6 & \leq 2x - 3y \\ -1000 & \leq 2x + 3y & \leq 18 \\ -2 & \leq -2x + 5y \\ 20 & \leq x + y\,. \end{cases} \tag{1}$$

LRA Example

$$\left\{\begin{array}{llrrrll} a = & 2x & +y & 2 & \leq a & & \\ b = & 2x & -3y & -6 & \leq b & & \\ c = & 2x & 3y & -1000 & \leq c & \leq 18 & \\ d = & -2x & +5y & -2 & \leq d & & \\ e = & x & +y & 20 & \leq e\,. & & \end{array}\right. \tag{2}$$

LRA Example

Gauss-like pivoting until:

$$\begin{cases} e = & 7/16c & -1/16d \\ a = & 3/4c & -1/4d \\ b = & 1/4c & -3/4d \\ x = & 5/16c & -3/16d \\ y = & 1/8c & +1/8d\,. \end{cases} \tag{3}$$

LRA Example

$e = 7/16c - 1/16d$
But: $c \leq 18$ and $d \geq -2$, so $-7/16c - 1/16d \leq 8$.
But we have $e \geq 20$, thus **no solution**.

Relevant original inequalities can be combined into an unsatisfiable one (thus the **theory lemma**)

$$\begin{array}{llllll} 7/16 & (-2x & -3y) & \geq & -7/16 & \times 18 \\ 1/16 & (-2x & +5y) & \geq & -1/16 & \times 2 \\ 1 & x & +y & \geq & 20 & \\ \hline & 0 & 0 & \geq & 12 & \end{array} \tag{4}$$

Linear integer arithmetic

Linear real arithmetic +

- branching: if LRA model $x = 4.3$, then $x \leq 4 \vee x \geq 5$
- (sometimes) Gomory cuts

Uninterpreted functions

$f(x) \neq f(y) \wedge x = z + 1 \wedge z = y - 1$
$\downarrow$
$f_x \neq f_y \wedge x = z + 1 \wedge z = y - 1$

Get $(x, y, z, f_x, f_y) = (1, 1, 0, 0, 1)$.
But if $x = y$ then $f_x = f_y$! Add $x = y \implies f_x = f_y$.

The problem over (x, y, z, f_x, f_y) becomes **unsatisfiable**.

Arrays

$update(f, x_0, y_0)$ the function mapping

- $x \neq x_0$ to $f[x]$
- x_0 to y_0.

Quantifiers

Show this formula is true:

$$(\forall i\, 0 \le i < j \implies t[i] = 42) \implies (\forall i\, 0 \le i \le j \implies update(t, j, 0)[i] = 42) \quad (5)$$

Equivalently, unsatisfiable:

$$0 \le i_0 \le j \land update(t, j, 0)[i_0] = 0 \land (\forall i\, 0 \le i < j \implies t[i] = 0)$$

Instantiation

Prove unsatisfiable:

$$0 \leq i_0 \leq j \wedge update(t, j, 0)[i_0] = 0 \wedge (\forall i\, 0 \leq i < j \implies t[i] = 0)$$

By **instantiation** $i = i_0$:

$$0 \leq i_0 \leq j \wedge update(t, j, 0)[i_0] = 0 \wedge (0 \leq i_0 < j \implies t[i_0] = 0)$$

Unsatisfiable

Contents

DPLL and CDCL

DPLL(T)

Natural domain SMT

- Exponential behaviour of DPLL(T)
- Abstract CDCL (ACDCL)
- Model-construction satisfiability calculus (MCSAT)

Other topics

Conclusion

DPLL(T) versus natural domain

DPLL(T) Boolean abstraction of the formula
Assign only to Boolean variables
Then refine abstraction by cubes unsatisfiable wrt theory

Natural domain Assign to Boolean and arithmetic variables

Diamonds

$D(n)$ the unsatisfiable formula:

$$\text{for } 0 \leq i < n \begin{cases} x_i - t_i \leq 2 \\ y_i - t_i \leq 3 \\ (t_{i+1} - x_i \leq 3) \vee (t_{i+1} - y_i \leq 2) \end{cases}$$
$$t_n - t_0 > 5n$$

DPLL(T) on diamonds

Will enumerate each combination of disjuncts =
All terms in disjunctive normal form

Fundamental limitation: can only use **atoms from original formula**.

Abstract CDCL

DPLL / CDCL assign truth values to Booleans

$\downarrow$ *generalization*

ACDCL assigns truth values to Booleans and intervals to reals
(or elements from an abstract domain)

e.g. if current assignment $x \in [1, +\infty)$ and $y = [4, 10]$
constraint $z = x - y \rightsquigarrow x \in [-9, +\infty)$

If too coarse, **split** intervals.
Akin to **constraint programming**.

Learning in ACDCL

Constraints $x \wedge z = x \cdot y \wedge z \leq -1$
Search context $x \leq -4$, **contradiction**.

Contradiction ensured by $x < 0$ **weaker** than search context.

Learn $x < 0$. Predicate **not in original formula**.

(CDCL-style learning would only learn $x > -4$.)

MCSAT

In DPLL(T), assign only to Booleans and atoms from original formula.
In MCSAT, assign to propositional atoms *and* numeric variables $x_1, \dots, x_n, \dots$

When finding an impossibility when trying to assign to x_{n+1}, derive a general impossibility on $x_1, \dots, x_n$ (**partial projection**).

Example: diamonds

$$\text{for } 0 \leq i \leq 2 \begin{cases} x_i - t_i \leq 2 \\ y_i - t_i \leq 3 \\ t_{i+1} - x_i \leq 3 \vee t_{i+1} - y_i \leq 2 \end{cases}$$

$$t_0 = 0$$

$$t_3 \geq 16$$

Pick $t_0 \mapsto 0$, $t_1 - x_0 \leq 3 \mapsto \mathbf{t}$, $x_0 \mapsto 0$,
$t_1 \mapsto 0$, $t_2 - x_1 \leq 3 \mapsto \mathbf{t}$, $x_1 \mapsto 0$,
$t_2 \mapsto 0$, $t_3 - x_2 \leq 3 \mapsto \mathbf{t}$, $x_2 \mapsto 0$.

No way to assign to x_3!
Because $x_2 \mapsto 0$ and $t_3 - x_2 \leq 3$ and $t_3 \geq 16$.

Analyze the failure

$x_2 \mapsto 0$ fails due to a **more general reason** (Fourier-Motzkin)

$$\begin{cases} t_3 - x_2 \leq 3 \\ t_3 \geq 16 \end{cases} \implies x_2 \geq 13$$

Possible to learn

$$t_3 - x_2 > 3 \lor x_2 \geq 13$$

Retract $x_2 \mapsto 0$.

Backtracking

We have learnt $t_3 - x_2 > 3 \vee x_2 \geq 13$.
$t_3 - x_2 \leq 3$ still assigned.

$$\{ \; x_2 \geq 13 x_2 - t_2 \leq 2 \quad \Longrightarrow \quad t_2 \geq 11$$

Thus learn

$$t_3 - x_2 > 3 \vee t_2 \geq 11$$

$t_3 - x_2 \leq 3 \mapsto \mathbf{t}$ retracted.

Continuation

Same reasoning for $t_3 - x_2 \leq 3 \mapsto \mathbf{f}$ yields by learning

$$t_3 - x_2 \leq 3 \vee t_2 \geq 11$$

Thus

$$\begin{cases} t_3 - x_2 > 3 \vee t_2 \geq 11 \\ t_3 - x_2 \leq 3 \vee t_2 \geq 11 \end{cases} \implies t_2 \geq 11$$

One learns $t_2 \geq 11$.
Then $t_1 \geq 6$ similarly.

But then no satisfying assignment to t_0!

NLSAT

(Dejan Jovanović, Leonardo De Moura)
MCSAT for **non-linear arithmetic**

Partial projection: Fourier-Motzkin replaced by partial **cylindrical algebraic decomposition**.

Contents

DPLL and CDCL

DPLL(T)

Natural domain SMT
- Exponential behaviour of DPLL(T)
- Abstract CDCL (ACDCL)
- Model-construction satisfiability calculus (MCSAT)

Other topics

Conclusion

Optimization

Basic SMT: “no solution” vs “here is a solution”

Optimization: here is a solution **maximizing** f

- binary search
- local optimization: $\bigwedge l_i \implies \phi$ linear programming in $\bigwedge l_i$

and even done for nonlinear arithmetic!

Quantifier elimination

F with quantifiers $\equiv$ G without quantifiers

Use SMT to prune out / simplify inside quantifier elimination
(do not generate partial solutions already covered etc.)

Formula simplification

F complicated $\equiv G$ "simpler"

Use SMT to prune out / simplify

Craig interpolation

From $F(\vec{x}, \vec{y}) \implies G(\vec{y}, \vec{z})$ get

$$F(\vec{x}, \vec{y}) \implies I(\vec{y}) \implies G(\vec{y}, \vec{z})$$

I can be obtained by quantifier elimination
but may be much simpler!

In fact often needs I “simple”.

Contents

DPLL and CDCL

DPLL(T)

Natural domain SMT
- Exponential behaviour of DPLL(T)
- Abstract CDCL (ACDCL)
- Model-construction satisfiability calculus (MCSAT)

Other topics

Conclusion

Basic idea

- **relax** the problem
- solve relaxed problem
- if spurious solution, **refine** the problem

Nonexhaustive list of SMT-solvers

See also http://smtlib.cs.uiowa.edu/
http://smtlib.cs.uiowa.edu/solvers.shtml

Free

- Z3 (Microsoft Research) https://github.com/Z3Prover
- Yices (SRI International) http://yices.csl.sri.com/
- CVC4 http://cvc4.cs.nyu.edu/web/

Non-free

- MathSAT (Fundazione Bruno Kessler) http://mathsat.fbk.eu/

VERIMAG

Joint research unit between

- Université Grenoble-Alpes
- Grenoble institute of technology (Grenoble-INP)
- **CNRS**