Satisfiability Modulo Integer Difference Logic

MTAT.03.238, Advanced Algorithmics

Simmo Saan
Institute of Computer Science, University of Tartu

Study IT in .ee
sponsored by Skype™

Introduction

Boolean satisfiability (SAT) problems, which consist of purely boolean algebra expressions, can be solved by SAT solvers. While many complicated problems can be described as SAT problems, describing the problem purely with booleans and constructing the respective conjunctive normal form (CNF) can be difficult and inefficient.

Being able to use integer variables and conditions makes modelling of many problems much easier and more intuitive. **Satisfiability modulo theories (SMT)** is a generalization of SAT, where the logical expressions may also contain, for example, integer variables and conditions between them. Such problems are solved by SMT solvers.

The goal of the project was to understand how SMT solvers work and implement a simple solver.

DPLL algorithm [1]

Davis–Putnam–Logemann–Loveland (DPLL) algorithm is a *complete* SAT/SMT solving algorithm based on backtracking search, where a call consists of three steps:

1. **Conflict checking:** if there is a conflict, backtrack.
2. **Unit propagation:** if a clause contains only one literal, then assume it to be true.
3. **Splitting:** choose a variable and split into two recursive searches, in the first one assume the variable to be true, in the second one false.

DPLL allows for significant improvements like conflict-driven clause learning (CDCL), which is used by state of the art SAT/SMT solvers.

Integer difference logic [2]

Integer difference logic (IDL) uses integer variables and constraints in the form $x - y \leq n$, where x, y are integer variables and n an integer constant. Systems of such difference constraints can, for example, encode various scheduling problems.

Furthermore, many other integer constraints can be converted to IDL by applying transformations like:

$$\begin{aligned}
x - y < n &\quad\mapsto\quad x - y \leq n - 1,\\
x - y \geq n &\quad\mapsto\quad y - x \leq -n,\\
x - y = n &\quad\mapsto\quad (x - y \leq n) \wedge (x - y \geq n),\\
x - y \neq n &\quad\mapsto\quad (x - y < n) \vee (x - y > n),\\
x \bowtie y + n &\quad\mapsto\quad x - y \bowtie n,\\
x \bowtie n &\quad\mapsto\quad x - \mathbf{Z} \bowtie n,
\end{aligned}$$

where $\bowtie \in \{\leq, <, \geq, >, =, \neq\}$ and $\mathbf{Z}$ is an artifical zero variable.

Bellman-Ford algorithm [3]

Systems of difference constraints (a) can be transformed into constraint graphs (b) containing an extra vertex. The Bellman-Ford single-source shortest paths algorithm can be used to find a solution from the distances (c), or detect a conflict from a negative weight cycle.

(a) Difference constraints

$$\begin{aligned}
x_1 - x_2 &\leq 2\\
x_2 - x_3 &\leq 1\\
x_3 - x_1 &\leq -1
\end{aligned}$$

(b) Constraint graph

$$\begin{tikzcd}
 & & x_2 \arrow[ld, "2"'] \\
x_0 \arrow[rru, bend left, "0"] \arrow[r, "0"] \arrow[rrd, bend right, "0"'] & x_1 \arrow[rd, "-1"'] & \\
 & & x_3 \arrow[uu, "1"']
\end{tikzcd}$$

(c) Solution

$$\begin{aligned}
x_1 &= \delta(x_0, x_1) = 0\\
x_2 &= \delta(x_0, x_2) = 0\\
x_3 &= \delta(x_0, x_3) = -1
\end{aligned}$$

Notable optimizations are:

- Avoiding the use of the extra vertex by initializing all distances to 0.
- Returning early if no edge is relaxed in an iteration.
- Reusing previous distances for initialization if edges are added.

SMT solvers [2]

The modern standard architecture of SMT solvers is the following:

- A **SAT solver** (e.g. DPLL algorithm) handles solving of the boolean expressions, without having to deal with the logic constraints within.
- A **logic solver** (e.g. Bellman-Ford algorithm for IDL) handles solving of the logic itself, without having to deal with boolean satisfiability.

The two independent solving procedures are combined by having the SAT solver call the logic solver when it needs to, for example, to check for conflicts.

Dinesman's multiple-dwelling problem

Problem definition in English and standard SMT-LIB language:

0. Baker, Cooper, Fletcher, Miller, and Smith live on different floors of an apartment house that contains only five floors.
```
(assert (distinct baker cooper fletcher miller smith))
(assert (<= 1 baker 5))
(assert (<= 1 cooper 5))
(assert (<= 1 fletcher 5))
(assert (<= 1 miller 5))
(assert (<= 1 smith 5))
```
1. Baker does not live on the top floor.
```
(assert (distinct baker 5))
```
2. Cooper does not live on the bottom floor.
```
(assert (distinct cooper 1))
```
3. Fletcher does not live on either the top or the bottom floor.
```
(assert (and (distinct fletcher 1)
             (distinct fletcher 5)))
```
4. Miller lives on a higher floor than does Cooper.
```
(assert (> miller cooper))
```
5. Smith does not live on a floor adjacent to Fletcher's.
```
(assert (and (distinct smith (+ fletcher 1))
             (distinct smith (- fletcher 1))))
```
6. Fletcher does not live on a floor adjacent to Cooper's.
```
(assert (and (distinct fletcher (+ cooper 1))
             (distinct fletcher (- cooper 1))))
```

SMT-LIB commands for checking satisfiability and respective variable values, and their outputs from the implemented SMT solver for the given problem:

```
(check-sat)
sat
(get-model)
((cooper 2) (fletcher 4) (baker 3) (miller 5) (smith 1))
```

Conclusion

The implemented SMT solver:

1. takes problem definition in SMT-LIB language,
2. converts constraints to difference constraints (if possible),
3. converts boolean expression to CNF,
4. uses DPLL to solve the boolean parts of the problem,
5. uses Bellman-Ford to solve IDL parts of the problem for DPLL.

Its correctness has been tested on logic puzzles like Dinesman's multiple-dwelling puzzle, Philips's liars puzzle, *n*-queens puzzle and zebra puzzle (Einstein's puzzle).

Project code repository: `https://github.com/sim642/algosmt`.

References

[1] A. Farinelli. *DPLL method.* Automated Reasoning. University of Verona. 2010–2011.

[2] A. Oliveras and E. Rodríguez-Carbonell. *Satisfiability Modulo Difference Logic.* May 25, 2016.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms.* 3rd. The MIT Press, 2009.