SAT and SMT Solving

Sarah Winkler

SS 2018

Department of Computer Science
University of Innsbruck

Outline

- Summary of Last Week
- Simplex Algorithm

Summary of Last Week

Definition (DPLL(T) systems)

- basic system $\mathcal{B}$: unit propagate, decide, fail, T-backjump, T-propagate
- full system $\mathcal{F}$: $\mathcal{B}$ plus T-learn, T-forget, and restart

Theorem (Correctness)

For derivation with final state S_n:

$$\parallel F \quad \Longrightarrow_{\mathcal{F}} \quad S_1 \quad \Longrightarrow_{\mathcal{F}} \quad S_2 \quad \Longrightarrow_{\mathcal{F}} \quad \dots \quad \Longrightarrow_{\mathcal{F}} \quad S_n$$

- *if* $S_n = \text{FailState}$ *then* F *is T-unsatisfiable*
- *if* $S_n = M \parallel F'$ *and* M *is T-consistent then* F *is T-satisfiable and* $M \vDash_T F$

Theorem (Termination)

Γ: $\parallel F \Longrightarrow^*_{\mathcal{F}} S_0 \Longrightarrow^*_{\mathcal{F}} S_1 \Longrightarrow^*_{\mathcal{F}} \dots$ *is finite if*

- *there is no infinite sub-derivation of only T-learn and T-forget steps, and*
- *for every sub-derivation* $S_i \overset{\text{restart}}{\Longrightarrow}_{\mathcal{F}} S_{i+1} \Longrightarrow^*_{\mathcal{F}} S_j \overset{\text{restart}}{\Longrightarrow}_{\mathcal{F}} S_{j+1} \Longrightarrow^*_{\mathcal{F}} S_k$ *with no restart steps in* $S_{i+1} \Longrightarrow^*_{\mathcal{F}} S_j$ *and* $S_{j+1} \Longrightarrow^*_{\mathcal{F}} S_k$*:*
  - *there are more $\mathcal{B}$-steps in* $S_j \Longrightarrow^*_{\mathcal{F}} S_k$ *than in* $S_i \Longrightarrow^*_{\mathcal{F}} S_j$*, or*
  - *a clause is learned in* $S_j \Longrightarrow^*_{\mathcal{F}} S_k$ *that is never forgotten in* Γ

Congruence Closure

Input: set of equations E and equation $s \approx t$, both without variables

Output: *valid* ($E \vDash_T s \approx t$) or *invalid* ($E \nvDash_T s \approx t$)

1. build congruence classes
   (a) put different subterms of terms in $E \cup \{s \approx t\}$ in separate sets
   (b) merge sets $\{\dots, t_1, \dots\}$ and $\{\dots, t_2, \dots\}$ for all $t_1 \approx t_2$ in E
   (c) merge sets $\{\dots, f(t_1, \dots, t_n), \dots\}$ and $\{\dots, f(u_1, \dots, u_n), \dots\}$ if t_i and u_i belong to same set for all $1 \leqslant i \leqslant n$, repeatedly
1. if s and t belong to same set then return *valid* else return *invalid*

Deciding Satisfiability of EUF Conjunctions

- consider EUF conjunction φ with free variables $x_1, \ldots, x_n$
- split φ into positive and negative literals

$$\varphi = \left(\bigwedge P\right) \wedge \left(\bigwedge \neg N\right)$$

- determine satisfiability

$\varphi = (\bigwedge P) \wedge (\bigwedge \neg N)$	unsatisfiable	
$\iff \exists x_1 \ldots x_n . (\bigwedge P) \wedge (\bigwedge \neg N)$	unsatisfiable	
$\iff (\bigwedge \widehat{P}) \wedge (\bigwedge \neg \widehat{N})$	unsatisfiable	skolemization
$\iff \neg \left((\bigwedge \widehat{P}) \wedge (\bigwedge \neg \widehat{N})\right)$	valid	φ unsat iff $\neg\varphi$ valid
$\iff \bigwedge \widehat{P} \to \bigvee \widehat{N}$	valid	deMorgan
$\iff \exists\, s \approx t$ in $\widehat{N}$ such that $\bigwedge \widehat{P} \to s \approx t$ valid		semantics of $\vee$
$\iff \exists\, s \approx t$ in $\widehat{N}$ such that $\bigwedge \widehat{P} \vDash_T s \approx t$		semantics of $\vDash$

Simplex Algorithm

Effects and Side Effects

- guaranteed to solve all your real arithmetic problems
- encountering Simplex can cause initial dizzyness
- in rare cases solving systems of linear inequalities can become addictive

Definition (Theory of Linear Arithmetic over C)

- for variables $x_1, \dots, x_n$, formulas built according to grammar

$$\varphi ::= \varphi \land \varphi \mid t = t \mid t < t \mid t \leqslant t$$

$$t ::= a_1 x_1 + \dots + a_n x_n + b \qquad \text{for } a_1, \dots, a_n, b \in \text{in carrier } C$$

- axioms are equality axioms plus calculation rules of arithmetic over C
- solution assigns values in C to $x_1, \dots, x_n$

Definitions

- Linear Real Arithmetic (LRA) uses carrier $C = \mathbb{R}$
- Linear Integer Arithmetic (LIA) uses carrier $C = \mathbb{Z}$

Example

- $x + y + z = 2 \land z > y \land y > -1$
  is satisfiable in LRA and LIA, e.g. with $v(x) = v(y) = 0$ and $v(z) = 2$
- $x < 3 \land 2x > 4$
  is unsatisfiable in LIA but satisfiable in LRA, e.g. with $v(x) = 2.5$

Satisfiability Problem for Linear Arithmetic

- integers (LIA): NP-complete
- reals (LRA) or rationals: polynomial Simplex algorithm

Some History

exponential worst-case complexity

1947 Danzig proposed Simplex algorithm to solve optimization problem:

$$\text{maximize } c(\vec{x}) \qquad \text{such that} \qquad A\vec{x} \leqslant b \text{ and } \vec{x} \geqslant 0$$

for linear objective function c, matrix A, vector b, and vector of variables $\vec{x}$

- also known as linear programming

1979 Khachiyan proposed polynomial version based on ellipsoid method

1984 Karmakar proposed polynomial version based on interior points method

2000- SMT solvers use DPLL(T) version to solve satisfiability problem

$$A\vec{x} \leqslant b$$

Problem Input (General Form)

- m equalities

$$a_1x_1 + \ldots a_nx_n = 0$$

- (optional) lower and upper bounds on variables

$$l_i \leqslant x_i \leqslant u_i$$

Lemma

any LRA problem without $<$ can be turned into equisatisfiable general form

Example

$$\begin{aligned} x - y &\geqslant -1 \\ y &\leqslant 4 \\ x + y &\geqslant 6 \\ 3x - y &\leqslant 7 \end{aligned} \quad \Longrightarrow \quad \begin{aligned} -x + y - s_1 &= 0 & s_1 &\leqslant 1 \\ y - s_2 &= 0 & s_2 &\leqslant 4 \\ -x - y - s_3 &= 0 & s_3 &\leqslant -6 \\ 3x - y - s_4 &= 0 & s_4 &\leqslant 7 \end{aligned} \qquad \text{slack variables}$$

- s_1, s_2, s_3, s_4 are slack variables
- x, y are problem variables

Representation

- represent equalities using $m \times (n+m)$ matrix A

$$\begin{aligned} -x + y - s_1 &= 0 & s_1 &\leqslant 1 \\ y - s_2 &= 0 & s_2 &\leqslant 4 \\ -x - y - s_3 &= 0 & s_3 &\leqslant -6 \\ 3x - y - s_4 &= 0 & s_4 &\leqslant 7 \end{aligned} \quad \Longrightarrow \quad \begin{pmatrix} -1 & 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 \\ -1 & -1 & 0 & 0 & -1 & 0 \\ 3 & -1 & 0 & 0 & 0 & -1 \end{pmatrix} \begin{array}{l} s_1 \leqslant 1 \\ s_2 \leqslant 4 \\ s_3 \leqslant -6 \\ s_4 \leqslant 7 \end{array}$$

- simplified matrix presentation

$$\begin{array}{cc} & \begin{array}{cc} x & y \end{array} \\ \begin{array}{c} s_1 \\ s_2 \\ s_3 \\ s_4 \end{array} & \begin{pmatrix} -1 & 1 \\ 0 & 1 \\ -1 & -1 \\ 3 & -1 \end{pmatrix} \end{array}$$

$\leftarrow$ nonbasic variables

basic variables $\rightarrow$

Notation

- simplified matrix is tableau
- B is set of basic variables (in tableau listed vertically)
- N is set of non-basic variables (in tableau listed horizontally)

DPLL(T) Simplex Algorithm

Input: conjunction of LRA literals φ without $<$

Output: satisfiable or unsatisfiable

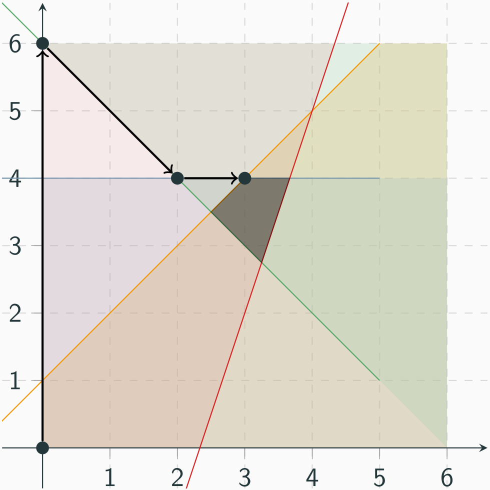
1. transform φ into general form and construct tableau
2. fix order on variables and assign 0 to each variable
3. if all basic variables satisfy their bounds then return satisfiable
4. let $x_i \in B$ be variable that violates its bounds
5. search for suitable variable $x_j \in N$ for pivoting with x_i
6. return unsatisfiable if search unsuccessful
7. perform pivot operation on x_i and x_j
9. update assignment
10. go to step 3

Simplex, Visually

- constraints

$$x - y \geqslant -1$$
$$y \leqslant 4$$
$$x + y \geqslant 6$$
$$3x - y \leqslant 7$$

- solution space
- Simplex solution search



Example

tableau

$$\begin{array}{c} \\ s_3 \\ x \\ y \\ s_4 \end{array} \begin{array}{c} \begin{array}{cc} s_2 & s_1 \end{array} \\ \begin{pmatrix} -2 & 1 \\ 1 & -1 \\ 1 & 0 \\ 2 & -1 \end{pmatrix} \end{array}$$

constraints

$s_1 \leqslant 1$
$s_2 \leqslant 4$
$s_3 \leqslant -6$
$s_4 \leqslant 7$

assignment

x	y	s_1	s_2	s_3	s_4
3	4	1	4	-7	7

1. Iteration 1
   - s_3 violates its bounds
   - decreasing s_3 requires to increase x or y (both suitable since they have no upper bound)
   - pivot s_3 with y:

     $$y = -x - s_3 \qquad s_1 = -2x - s_3$$
     $$s_2 = -x - s_3 \qquad s_4 = 4x + s_3$$

   - update assignment

     $$s_3 = s_3 - 6 = -6 \qquad y = 6$$
     $$s_1 = 6 \qquad s_2 = 6 \qquad s_4 = -6$$

DPLL(T) Simplex Algorithm

non-basic $\vec{x}_N$

basic $\vec{x}_B$

$$\begin{array}{cc} & x_j \\ x_i & \begin{pmatrix} \cdots & & \cdots \\ & A_{ij} & \\ \cdots & & \cdots \end{pmatrix} \end{array}$$

$$A\vec{x}_N = \vec{x}_B \tag{1}$$

$$-\infty \leqslant l_i \leqslant x_i \leqslant u_i \leqslant +\infty \tag{2}$$

Invariant

- (1) is satisfied and (2) holds for all nonbasic variables

Pivoting

- swap basic x_i and non-basic x_j, so $i \in B$ and $j \in N$

$$x_i = \sum_{k \in N} A_{ik} x_k \quad \Longrightarrow \quad x_j = \frac{1}{A_{ij}} \left(x_i - \sum_{k \in N - \{j\}} A_{ik} x_k \right) \tag{$\star$}$$

- new tableau A' consists of ($\star$) and $A_{B-\{i\}}\vec{x}_N = \vec{x}_{B-\{i\}}$ with ($\star$) substituted

Update

- assignment of x_i is updated to previously violated bound l_i or u_i,
- assignment of x_k is recomputed using ($\star$) and A' for all $k \in B - \{i\} \cup \{j\}$

DPLL(T) Simplex Algorithm

$$A\vec{x}_N = \vec{x}_B \qquad (1)$$

$$-\infty \leqslant l_i \leqslant x_i \leqslant u_i \leqslant +\infty \qquad (2)$$

Suitability

- basic variable x_i violates lower and/or upper bound
- pick nonbasic variable x_j such that
  - if $x_i < l_i$: $A_{ij} > 0$ and $x_j < u_j$ or $A_{ij} < 0$ and $x_j > l_j$
  - if $x_i > u_i$: $A_{ij} > 0$ and $x_j > l_j$ or $A_{ij} < 0$ and $x_j < u_j$

Observation

- problem is unsatisfiable if no suitable pivot exists

Bland's Rule

- pick lexicographically smallest (i, j) that is suitable pivot
- guarantees termination

How to Treat Strict Inequalities

replace in LRA formula φ every strict inequality

$$a_1x_1 + \cdots + a_nx_n < b$$

by non-strict inequality

$$a_1x_1 + \cdots + a_nx_n \leqslant b - \delta$$

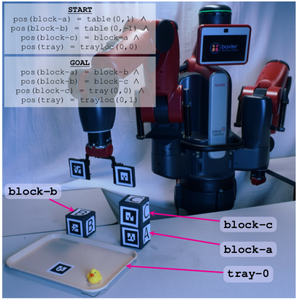
to obtain formula φ_δ in LRA without $<$, and treat δ symbolically during Simplex algorithm

Lemma

$$\varphi \textit{ is satisfiable} \quad\iff\quad \exists \textit{ rational number } \delta > 0 \textit{ such that } \varphi_\delta \textit{ is satisfiable}$$

Application: Motion Planning for Robots

- robots needs to plan motions
  to place objects correctly
- instance of *constraint based planning*
- encoding
  - fix number of time slots $t_1, \ldots, t_n$
  - action variable a_i for time t_i encodes which action performed at time t_i (one action per time)
  - actions require precondition and imply postcondition
  - use arithmetic to minimize path



Neil T. Dantam, Zachary K. Kingston, Swarat Chaudhuri, and Lydia E. Kavraki.
Incremental Task and Motion Planning: A Constraint-Based Approach.
In: The International Journal of Robotics Research, 2018.

(Almost) Everything is Better With Arithmetic

LRA and LIA admit more efficient encodings of

- n-queens
- Sudoku
- graph coloring
- Minesweeper
- travelling salesperson
- rabbit problem
- planning problems
- scheduling problems
- component configuration problems
- everything with cardinality constraints
- ...

Bibliography

Bruno Dutertre and Leonardo de Moura.
A Fast Linear-Arithmetic Solver for DPLL(T).
In Proc. of International Conference on Computer Aided Verification, pp. 81–94, 2006.

Bruno Dutertre and Leonardo de Moura
Integrating Simplex with DPLL(T)
Technical Report SRI–CSL–06–01, SRI International, 2006