Decision Procedures and Verification

Martin Blicha

Charles University

16.4.2018

Theory of Linear Arithmetic

Theory of linear arithmetic

- quantifier-free conjunctive fragment

Definition

A quantifier-free formula in the language of the theory of linear arithmetic is defined by the following grammar:

$$\begin{aligned} \mathit{fla} &: \mathit{fla} \wedge \mathit{fla} \mid \mathit{atom} \\ \mathit{atom} &: \mathit{sum}\ \mathit{op}\ \mathit{sum} \\ \mathit{op} &: \ = \ \mid \ \leq \ \mid \ < \\ \mathit{sum} &: \mathit{term} \mid \mathit{sum} + \mathit{term} \\ \mathit{term} &: \mathit{identifier} \mid \mathit{constant} \mid \mathit{constant}\ \mathit{identifier} \end{aligned}$$

where identifiers are variables defined over single ininite domain.

Domains

- Reals (LRA)
- Integers (LIA)
- $x > 0 \wedge x < 1$
  - Satisfiable in LRA, unsatisfiable in LIA
- Deciding satisfiability of conjunction of linear constraints over reals is polynomial, while it is NP-complete over integers.

Algorithms for solving linear constraints

- General Simplex (R)
- Fourier-Motzkin variable elimination (R)
- Branch-and-bound (I)
- Omega test (I)

Simplex algorithm

- Proposed by Dantzig in 1947.
- Optimizes an objective function given a set of linear constraints.
  - linear program (LP)
- Worst-case exponential, but efficient in practice.
  - Polynomial algorithms exist, e.g. *ellipsoid* method.
- Traverses vertices of a convex polytope defined by the constraints.

General simplex

- Input are constraints of the following form:
  - Equalities $a_1 x_1 + \cdots + a_n x_n = 0$
  - Lower and upper bounds on the variables $l_i \leq x_i \leq u_i$, where l_i and u_i are constants.
    - Bounds are optional
- This form is called *general* form
- Transformation of any linear constraint $L \bowtie R$ (with $\bowtie \in \{=, \leq, \geq\}$) to general form:
  - Move every term with variable from R to L to obtain $L' \bowtie b$ where b is a constant.
  - Introduce new variable s_i. Add constraints $L' - s_i = 0$ and $s_i \bowtie b$
  - Replace equalities with two inequalities (both $\leq$ and $\geq$)
- Original and transformed problem are equisatisfiable
- New variables are called *additional* or *slack* variables, the variables in the original constrains are referred to as *problem* variables.

General simplex - problem representation

- With n problem and m additional variables, the problem is represented as $m \times (n + m)$-matrix $\mathbf{A}$, together with bounds on the variables.
- The decision problem can be written as $\mathbf{A}\mathbf{v} = 0$ and $\bigwedge_i l_i \leq s_i \leq u_i$, with $\mathbf{v}$ a vector of problem and additional variables.
- The submatrix corresponding to columns of additional variables is a diagonal matrix with -1 on the diagonal.
  - There is always such submatrix during the run of the algorithm.
- Variables of the columns of this diagonal matrix are *basic* (also *dependent*) variables. The others are nonbasic variables.
- *Tableau* is a representation of $\mathbf{A}$ as $m \times n$ matrix ($\mathbf{A}$ without the diagonal submatrix) with rows labeled by basic variables and columns labeled by nonbasic variables.

General simplex - the algorithm (1)

- Data structures:
  - Set of basic variables - $\mathcal{B}$
  - Set of nonbasic variables - $\mathcal{N}$
  - Tableau
  - Assignment $\alpha : \mathcal{B} \cup \mathcal{N} \rightarrow \mathbb{Q}$
- Initially: Additional variables are basic, program variables are nonbasic, assignment assigns 0 to all variables.
- Algorithm maintains two invariants:
  1. $\mathbf{A}\alpha(\mathbf{v}) = \mathbf{0}$ ($\mathbf{v}$ is the vector of all variables)
  2. The values of the nonbasic variables are within their bounds: $\forall v_j \in \mathcal{N} . l_j \leq \alpha(v_j) \leq u_j$.

General simplex - the algorithm (2)

Algorithm GENERAL-SIMPLEX

1. Transform input to general form $\mathbf{Av} = 0$ and $\bigwedge_i l_i \leq s_i \leq u_i$.
2. Initialize the data structures.
3. Determine a fixed order on the variables.
4. If no basic variable violates its bounds, return SAT. Otherwise take the first basic variable v_i violating its bounds.
5. Find the first suitable nonbasic variable v_j for pivoting with v_i. If there is no such variable, return UNSAT.
6. Update α so that v_i satisfies its bounds. Perform the pivot operation on v_i and v_j.
7. Go to step 4.

General simplex - pivoting

- *Pivot* operation (or pivoting) = update of the tableau corresponding to swapping one basic and one nonbasic variable.
- Given a basic variable v_i and nonbasic variable v_j, the coefficient a_{ij} is the pivot element. The column of x_j is the pivot column, the row of x_i is the pivot row.
- Steps:
  1. Solve row i for x_j
  2. For all rows $l \neq i$, eliminate x_j by using the equality for x_j obtained from row i.
- Fixed ordering of variables ensures that no set of basic variables is ever repeated and hence guarantees termination.
  - *Bland's rule*

General simplex - final notes

- Strict inequalities:
  - Set of constraints containing strict inequalities $\{s_1 > 0, \dots s_n > 0$ is satisfiable iff there exists a rational number $\delta > 0$ such that the same set of constraints with replaced inequalities $s_1 \geq \delta, \dots s_n \geq \delta$ is satisfiable.
- DPLL(T) setting:
  - Addition of a constraint:
    1. If it is a bound on nonbasic variable, update α to restore the second invariant.
    2. Run General-Simplex from step 4.
  - Removal of a constraint: Disable a bound on the corresponding variable.
  - For backtracking, only α needs to be updated, the tableau need not change.

Branch and bound method (1)

- Developed for solving integer linear programs as optimization problems.
  - Here modified version for deciding feasibility.
- Idea:
  - Solve *relaxed* problem
  - If satisfiable but satisfying assignment is not integral, add constraints forbidding this non-integer assignment but preserving all potential integral ones.

Relaxed problem

Given an integer linear system S, its relaxation is S without the integrality requirement (i.e., the variables are not required to be integer).

- If relaxed problem is unsatisfiable, so is the original problem.

Branch and Bound method (2)

- Assume the existence of a procedure $LP_{feasible}$ which receives an linear system and returns satisfying assignment or $UNSAT$ of no satisfying assignment exists.
  - Easy modification of GENERAL-SIMPLEX

---

procedure SEARCH-INTEGRAL-SOLUTION(S)
    $res = LP_{feasible}(relaxed(S))$
    if $res == UNSAT$ **then return**
    if res is integral **then exit**(SAT)
    Select a variable v with a nonintegral value r
    SEARCH-INTEGRAL-SOLUTION($S \cup (v \leq \lfloor r \rfloor)$)
    SEARCH-INTEGRAL-SOLUTION($S \cup (v \geq \lceil r \rceil)$)
                ▷ No integer solution in this branch
procedure FEASIBILITY-BRANCH-AND-BOUND(S)
    SEARCH-INTEGRAL-SOLUTION(S)
    return(UNSAT)

Completeness of Branch and Bound

- As presented, FEASIBILITY-BRANCH-AND-BOUND is not complete.
  - $1 \leq 3x - 3y \leq 2$ has no integer solution but unbounded real solutions.
- Completeness can be achieved using the *small-model* property.
  - A bound on each variable can be computed such that if a solution exists, there also exists one within these bounds.
  - If added explicitly as constraints, it makes Branch and Bound complete.

Cutting-planes

- *Cutting-planes* are constraints that are added to a system that remove only noninteger solutions.
- They improve the tightness of the relaxation, hence can make branch-and-bound faster
  - $\Rightarrow$ *branch-and-cut*
- *Gomory cuts*
  - Can be generated from assignment returned by general Simplex method and current state of the tableau.

Fourier-Motzkin Variable Elimination (1)

- Decides satisfiability of a conjunction of linear constraints.
- In practice not as efficient as Simplex method.
  - But competitive on small formulas.
- Used for eliminating existential quantifiers from quantified formulas of linear arithmetic.
- Eliminates variables from the system one by one while preserving satisfiability.

Fourier-Motzkin Variable Elimination (2)

1. Eliminate all equalities from the system
   - Express one variable as a linear combination of others and substitute in all other constraints.
2. Repeatedly choose variable and remove it from the system by *projecting* its constraints onto the rest of the system.
3. Deciding satisfiability of a system with single variable is trivial.

Projection of a variable (1)

- Assumptions:
  - x_n is picked to be eliminated next
  - All constraints have the following form (with i ranging over constraints):

$$\sum_{j=1}^{n} a_{i,j} x_j \leq b_i$$

- Gather all constraints containing x_n with non-zero coefficient and use them to derive *bounds* on x_n.
  - $\beta = \frac{b_i}{a_{i,n}} - \sum_{j=1}^{n-1} \frac{a_{i,j}}{a_{i,n}} x_j$
  - If $a_{i,n} > 0 \Rightarrow$ upper bound, if $a_{i,n} < 0 \Rightarrow$ lower bound.
- If x_n is not bounded both ways, i.e. it has only lower bounds or only upper bounds, the variable is *unbounded*.
- Unbounded variable can be removed from the system together with all constraints where it occurs.

Projection of a variable (2)

- If a variable has both kind of bounds it is *bounded*.
- Enumerate pairs of lower and upper bounds derived in the previous step.
- For each pair of bounds $\beta_l \leq x_n \leq \beta_u$ the following constraint is added:

$$\beta_l \leq \beta_u$$

- This may simplify to constraint $0 \leq b$ where b is a negative constant, which means the problem is *unsatisfiable*.
- Otherwise the constraints containing x_n are removed and next variable to eliminate is picked.

Fourier-Motzkin Variable Elimination - final notes

- The algorithm can be naturally extended to handle strict inequalities.
  - If either lower or upper bound is strict, so is the resulting constraint.
- Complexity:
  - Increase in number of constraints in one step in worst case is from m to $\frac{m^2}{4}$
  - Overall increase in worst case is from m to $\frac{m^{2^n}}{4^n}$.

Preprocessing

- Techniques to modify the input system of constraints independent of the decision procedure used.

1. Constraints of form $\sum_{j=0}^{n} a_j x_j \leq b$ are redundant if
$$\sum_{j|a_j>0} a_j u_j + \sum_{j|a_j<0} a_j l_j \leq b$$
2. Bounds on individual variables can be derived (and possibly tighten). If $a_0 > 0$ then
$$x_0 \leq (b - \sum_{j|a_j>0, j>0} a_j l_j - \sum_{j|a_j<0} a_j u_j)/a_0$$
and if $a_0 < 0$ then
$$x_0 \geq (b - \sum_{j|a_j>0} a_j l_j - \sum_{j|a_j<0, j>0} a_j u_j)/a_0$$

Preprocessing for integers

1. Multiply all constraints to make all constants and coefficients integral.
2. Weak inequalities can be used instead of strong ones.

$$\sum_{i=1}^{n} a_i x_i < b \Rightarrow \sum_{i=1}^{n} a_i x_i \leq b - 1$$