universität innsbruck

SS 2024 lecture 6

Constraint Solving

René Thiemann and Fabian Mitterwallner

based on a previous course by Aart Middeldorp

Outline

Definition (Equality Logic)

terms are restricted to variables, no quantifiers:

$$\varphi \ ::= \ \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi \mid \neg\varphi \mid t = t \qquad\qquad t \ ::= \ x$$

T-solver for conjunction φ of equality logic literals over set of variables V

Definition

equality graph is undirected graph $G_=(\varphi) = (V, E_=, E_{\neq})$ with

- $E_=$ edges corresponding to positive (equality) literals in φ
- $E_{\neq}$ edges corresponding to negative (inequality) literals in φ

Lemma

φ is satisfiable $\iff$
$G_=(\varphi)$ contains no simple contradictory cycles (with exactly one $E_{\neq}$ edge)

Quantifier-Free Fragment of Equality Logic with Uninterpreted Functions (EUF)

$$\varphi \ ::= \ \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi \mid \neg\varphi \mid t = t \qquad\qquad t \ ::= \ f(t, \dots, t)$$

T-Solver for Conjunctive Fragment via Congruence Closure

input: set E of ground equations and ground equation $s \approx t$

output: valid ($E \vDash_T s = t$) or invalid ($E \nvDash_T s = t$)

1. build congruence classes
   - (a) put different subterms of terms in $E \cup \{s = t\}$ in separate sets
   - (b) merge sets $\{\dots, t_1, \dots\}$ and $\{\dots, t_2, \dots\}$ for all $t_1 = t_2$ in E
   - (c) repeatedly merge sets

     $$\{\dots, f(s_1, \dots, s_n), \dots\} \ \text{ and } \ \{\dots, f(t_1, \dots, t_n), \dots\}$$

     if s_i and t_i belong to same set for all $1 \leqslant i \leqslant n$
2. if s and t belong to same set then return valid else return invalid

Outline

Difference Logic

atoms are constraints of the form

- $x - y \leqslant c$
- $x - y < c$

where x and y are variables and c is some constant of $\mathbb{Z}$ or $\mathbb{Q}$

Remarks

- difference logic is fragment of linear arithmetic; advantage: faster decision procedure
- domains: $\mathbb{Z}$ or $\mathbb{Q}$ (T-solver in polynomial time)
- $x - y = c \iff x - y \leqslant c \wedge y - x \leqslant -c$
- $x - y \geqslant c \iff y - x \leqslant -c$
- $x - y > c \iff y - x < -c$
- $x < c \iff x - x_0 < c$ where x_0 is fresh variable that must be assigned 0

Example Job-Shop Scheduling

- m machines $(M_1, \ldots, M_m)$ and n jobs $(J_1, \ldots, J_n)$
- each job J_i is sequence $(M^i_1, d^i_1), \ldots, (M^i_{n_i}, d^i_{n_i})$ of operations consisting of machine and duration (rational number; $\tau(M, d) = d$)
- O is multiset of all operations from all jobs
- schedule is function S that defines for each operation $v \in O$ its starting time $S(v)$ on machine specified by v
- schedule S is feasible if

$$S(v) \geqslant 0 \qquad \text{for all } v \in O$$

$$S(v_i) + \tau(v_i) \leqslant S(v_j) \qquad \text{for all consecutive } v_i, v_j \text{ in same job}$$

$$S(v_i) + \tau(v_i) \leqslant S(v_j) \vee S(v_j) + \tau(v_j) \leqslant S(v_i)$$

for every pair of different operations v_i, v_j scheduled on same machine

- aim: minimize global duration gd; add constraints $S(v) + \tau(v) \leq gd$ for each $v \in O$

Definition Inequality Graph

conjunction φ of nonstrict difference constraints

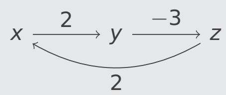
- inequality graph of φ contains edge from x to y with weight c for every constraint $x - y \leqslant c$ in φ

Theorem

conjunction φ of nonstrict difference constraints is satisfiable $\iff$ inequality graph of φ has no negative cycle

Example

$$x - y \leqslant 2$$
$$y - z \leqslant -3$$
$$z - x \leqslant 2$$



satisfiable

Theorem

conjunction φ of nonstrict difference constraints is satisfiable $\iff$ inequality graph of φ has no negative cycle

Proof

$\Rightarrow$ negative cycle $x_1 \xrightarrow{k_1} x_2 \xrightarrow{k_2} x_3 \longrightarrow \cdots \longrightarrow x_n \xrightarrow{k_n} x_1$

in inequality graph of φ corresponds to conjuction

$$x_1 - x_2 \leqslant k_1 \;\wedge\; x_2 - x_3 \leqslant k_2 \;\wedge\; \cdots \;\wedge\; x_n - x_1 \leqslant k_n$$

adding these literals gives

$$0 \leqslant k_1 + k_2 + \cdots + k_n$$

with $k_1 + k_2 + \cdots + k_n < 0$ ↯

Theorem

conjunction φ of nonstrict difference constraints is satisfiable $\iff$ inequality graph of φ has no negative cycle

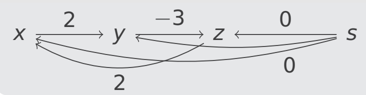
Proof

$\Leftarrow$ assume inequality graph of φ has no negative cycle

construct satisfying assignment for φ as follows

- add additional starting node s in graph, add edges $s \to x$ with weight 0 for all variables x
- define $v(x) = -\text{distance}(s, x)$; well-defined, since there are no negative cycles
- v satisfies φ

Example



$v(x) = 1, v(y) = 0, v(z) = 3$

Algorithms for Distance Computation and Negative Cycle Detection

- Dijkstra
  - computes distances from a single source to all other nodes
  - complexity: $\mathcal{O}(|V| \cdot log(|V|) + |E|)$
  - restriction: no negative cycles allowed
- Bellman–Ford
  - computes distances from a single source to all other nodes
  - complexity: $\mathcal{O}(|V| \cdot |E|)$
  - can also detect negative cycles
- Floyd–Warshall
  - computes distances between all nodes
  - complexity: $\mathcal{O}(|V|^3)$
  - can also detect negative cycles

$\Rightarrow$ use Bellman–Ford algorithm for difference logic

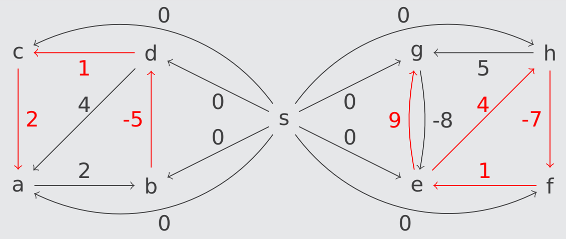
Bellman–Ford Algorithm for Inequality Graphs

Input inequality graph (V, E, w) with fresh starting node s

Output "$\exists$ negative cycle" or distances to node s

1. distance[v] := 0 for all nodes $v \in V$ (this step is special for inequality graphs)
2. repeat $|V| - 1$ times
   for all $(u, v) \in E$ do
   if distance[v] > distance[u] + $w(u, v)$ then
   distance[v] := distance[u] + $w(u, v)$
   predecessor[v] := u
3. for all $(u, v) \in E$ do
   if distance[v] > distance[u] + $w(u, v)$ then
   return "$\exists$ negative cycle"
   which can be reconstructed using predecessor array
4. return distance array, shortest paths available via predecessor array

Example Bellman–Ford in Action



iteration	a	b	c	d	e	f	g	h
0	0	0	0	0	0	0	0	0
1	0	0	0	-5	-8	-7	0	0
2	-1	0	-4	-5	-8	-7	0	-4
3	-2	0	-4	-5	-8	-11	0	-4
4	-2	0	-4	-5	-10	-11	0	-4
5	-2	0	-4	-5	-10	-11	-1	-6

. . . 2 more iterations, then negative cycle is detected

Outline

Definition (Theory of Linear Arithmetic over D)

- for variables $x_1, \ldots, x_n$, built quantifier-free formulas according to grammar

$$\varphi ::= \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid t < t \mid t \leq t \qquad s = t \text{ is encoded as } s \leq t \wedge t \leq s$$

$$t ::= a_1 x_1 + \cdots + a_n x_n + b \qquad \text{for } a_1, \ldots, a_n, b \in \text{ in domain } D$$

- **solution** assigns values in D to $x_1, \ldots, x_n$

Definitions

- **Linear Real Arithmetic** (**LRA**) uses domain $D = \mathbb{R}$
- **Linear Integer Arithmetic** (**LIA**) uses domain $D = \mathbb{Z}$

Example

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

Relevance of Linear Arithmetic

LRA and LIA admit more natural and succinct encodings of

- everything with cardinality constraints: n-queens, Sudoku, Minesweeper, . . .
- planning problems
- scheduling problems
- component configuration problems
- . . .

T-Solver for Conjunctions of Linear Arithmetic Inequalities

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

exponential worst-case complexity

Some History

1947 **Danzig** proposed Simplex algorithm to solve optimization problem:

maximize $c(\vec{x})$ such that $A\vec{x} \leq b$ and $\vec{x} \geq 0$

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- T-solver version of simplex algorithm to solve satisfiability problem

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

Problem Input (General Form)

- m equalities $a_1x_1 + \cdots + a_nx_n = 0$
- (optional) lower and upper bounds on variables $l_i \leq x_i \leq u_i$

Lemma

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

Example

$$\begin{array}{r} x - y \geq -1 \\ y \leq 4 \\ x + y \geq 6 \\ 3x - y \leq 7 \end{array} \implies \begin{array}{rl} -x + y - s_1 = 0 & s_1 \leq 1 \\ y - s_2 = 0 & s_2 \leq 4 \\ -x - y - s_3 = 0 & s_3 \leq -6 \\ 3x - y - s_4 = 0 & s_4 \leq 7 \end{array}$$

slack variables

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

Representation

- represent equalities

$$\begin{aligned} -x + y - s_1 &= 0 \\ y - s_2 &= 0 \\ -x - y - s_3 &= 0 \\ 3x - y - s_4 &= 0 \end{aligned}$$

via $m \times n$ matrix presentation

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

$\leftarrow$ **nonbasic** variables

basic variables $\rightarrow$

Notation

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

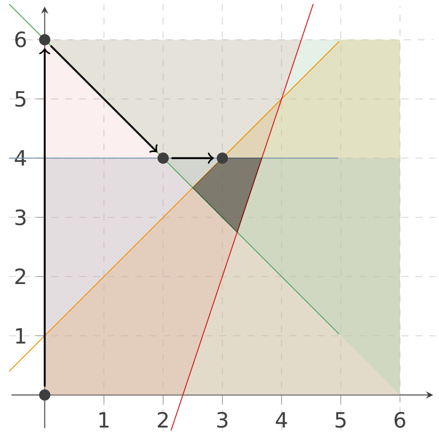
Simplex Algorithm as T-solver

Input: conjunction of LRA atoms φ without $<$
Output: satisfiable assignment 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 **current (satisfying) assignment**
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
  $x - y \geq -1$
  $y \leq 4$
  $x + y \geq 6$
  $3x - y \leq 7$
- solution space
- Simplex solution search



Example

tableau

	s_2	s_1
s_3	-2	1
x	1	-1
y	1	0
s_4	2	-3

bounds

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

assignment

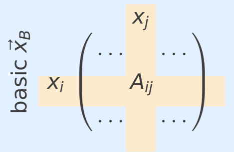
x	y	s_1	s_2	s_3	s_4
3	4	1	4	-7	5

3 Iteration 3

- s_1 violates its bounds
- decreasing s_1 requires to decrease s_2 or s_3 (both suitable since they have no lower bound)
- pivot s_1 with s_3: $s_3 = s_1 - 2s_2$ $\quad x = -s_1 + s_2$ $\quad y = s_2$ $\quad s_4 = -3s_1 + 2s_2$
- update assignment $s_1 := 1$ $\quad s_2 = 4$ $\quad s_3 := -7$ $\quad x := 3$ $\quad y := 4$ $\quad s_4 := 5$

Simplex Algorithm as T-solver

$$A\vec{x}_N = \vec{x}_B \qquad (1)$$
$$-\infty \leq l_i \leq x_i \leq u_i \leq +\infty \qquad (2)$$



Invariant

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

Pivoting

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

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

- 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 A' for all $k \in B - \{i\} \cup \{j\}$

Simplex Algorithm as T-solver

$$A\vec{x}_N = \vec{x}_B \qquad (1)$$
$$\forall i \in N. \ -\infty \leq l_i \leq x_i \leq u_i \leq +\infty \qquad (2)$$

Suitability

- basic variable x_i violates lower or upper bound
- pick nonbasic variable x_j such that
  - if $u_i < l_i$: problem is trivially unsatisfiable and no suitable x_j exists
  - 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

Outline

Motivation

strict inequalities naturally arise, e.g., as negated non-strict inequalities in DPLL(T)

$$\neg(x+3 \leq 5y) \quad \leftrightarrow \quad x+3 > 5y$$

How to Treat Strict Inequalities

replace in conjunction of inequalities C every strict inequality

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

by non-strict inequality

$$a_1x_1 + \cdots + a_nx_n \geq b + \delta \qquad\qquad a_1x_1 + \cdots + a_nx_n \leq b - \delta$$

to obtain constraints C_δ in LRA without $>$ and $<$, and treat δ symbolically during simplex algorithm (δ represents small positive rational number)

Lemma

C is satisfiable $\iff$ $\exists$ rational number $\delta > 0$ such that C_δ is satisfiable

Symbolical computation with δ

- δ represents small positive rational number, i.e., smaller than every concrete rational number that occurs during the computations of the simplex algorithm
- treat δ symbolically: $\mathbb{Q}_\delta = \{(c,k) \mid c,k \in \mathbb{Q}\}$ with (c,k) representing $c + k\delta$
- operations for all $a, c_1, k_1, c_2, k_2 \in \mathbb{Q}$
  - addition: $(c_1,k_1) + (c_2,k_2) = (c_1+c_2, k_1+k_2)$
  - multiplication: $a \cdot (c_1,k_1) = (ac_1, ak_1)$
  - equality: $(c_1,k_1) = (c_2,k_2) \quad \leftrightarrow \quad c_1 = c_2 \wedge k_1 = k_2$
  - comparison: $(c_1,k_1) < (c_2,k_2) \quad \leftrightarrow \quad c_1 < c_2 \vee c_1 = c_2 \wedge k_1 < k_2$
    $(c_1,k_1) \leq (c_2,k_2) \quad \leftrightarrow \quad c_1 < c_2 \vee c_1 = c_2 \wedge k_1 \leq k_2$
- multiplication of two $\mathbb{Q}_\delta$-numbers is not defined, but also not required for the simplex algorithm
  - coefficients in the tableau stay in $\mathbb{Q}$
  - only bounds and assignment require $\mathbb{Q}_\delta$

Example

tableau constraints assignment

$$\begin{array}{c} \\ x \\ s_2 \\ s_3 \end{array} \begin{array}{c} \begin{array}{cc} s_1 & y \end{array} \\ \begin{pmatrix} 1 & -1 \\ 2 & -3 \\ -1 & 3 \end{pmatrix} \end{array} \qquad \begin{array}{l} (2,1) < s_1 \\ (0,0) \leqslant s_2 \\ (1,0) \leqslant s_3 \end{array}$$

x	y	s_1	s_2	s_3
$(2,1)$	$(0,0)$	$(2,1)$	$(4,2)$	$(-2,-1)$

- pivot s_1 with x $\implies$ $x = s_1 - y$

$$s_2 = 2(s_1 - y) - y \quad = 2s_1 - 3y$$

$$s_3 = -(s_1 - y) + 2y = -s_1 + 3y$$

Outline

Kröning and Strichmann

- Sections 5.1, 5.2 and 5.7

Further Reading

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.

Important Concepts

- basic and nonbasic variables
- Bellman–Ford algorithm
- difference logic
- linear arithmetic (LRA and LIA)
- negative cycles
- pivoting
- $\mathbb{Q}_\delta$
- simplex algorithm
- suitable pair of variables
- tableau