Decision procedures

Ashutosh Gupta

TIFR, India

Compile date: 2016-12-06

Specialized algorithm for theories!

A decision procedure is a specialized algorithms to decide a sentence under a theory.

$$\models_{\mathcal{T}} F$$

For a given theory, we can know if there is a decision procedure.

Once we have it, we know it exists! Otherwise, the theory is called undecidable.

Example decision procedures

In this session, we will see a series of decision procedures.

- Ackermann's reduction for QF_EUF
- Cooper's method for integers
- Procedures for rationals.
  - Difference bound matrices(DBM) for difference logic
  - DBM for Octagonal constraints
  - Simplex for linear rational arithmetic

Topic 2.1

Ackermann's reduction

Reminder: Theory of equality and function symbols (EUF)

EUF syntax: first order formulas with signature $\mathbf{S} = (\mathbf{F}, \emptyset)$,
i.e., countably many function symbols and no predicates.

The theory axioms include

1. $\forall x.\ x \approx x$
2. $\forall x, y.\ x \approx y \Rightarrow y \approx x$
3. $\forall x, y, z.\ x \approx y \wedge y \approx z \Rightarrow x \approx z$
4. for each $f/n \in \mathbf{F}$,

$$\forall x_1, .., x_n, y_1, .., y_n.\ x_1 \approx y_1 \wedge .. \wedge x_n \approx y_n \Rightarrow f(x_1, .., x_n) \approx f(y_1, .., y_n)$$

Since the axioms are valid in FOL with equality, the theory is sometimes referred as the base theory.

Note: Predicates can be easily added if desired

Proofs in quantifier-free fragment of $\mathcal{T}_{EUF}$(QF_EUF)

Proof rules of $\mathcal{T}_{EUF}$

$$\frac{x \approx y}{y \approx x}\, Symmetry$$

$$\frac{x \approx y \quad y \approx z}{x \approx z}\, Transitivity$$

$$\frac{x_1 \approx y_1 \quad .. \quad x_n \approx y_n}{f(x_1, .., x_n) \approx f(y_1, .., y_n)}\, Congruence$$

Example 2.1

Consider: $y \approx x \wedge y \approx z \wedge f(x,u) \not\approx f(z,u)$

$$\dfrac{\dfrac{\dfrac{y \approx x}{}}{\dfrac{x \approx y \quad y \approx z}{x \approx z}} \quad f(x,u) \not\approx f(z,u)}{f(x,u) \approx f(z,u)}$$

$$\bot$$

Ackermann's Reduction

The insight: the rules needed to be applied only finitely many possible ways.

Algorithm: eagerly apply the rules and get a formula needing only Boolean reasoning.

Algorithm 2.1: QF_EUF_Sat(F)

Input: F formula QF_EUF
Output: SAT/UNSAT
Let Ts be subterms of F;
Let en be a function from Ts to fresh constants;
$G := en(F)$;
foreach $f(x_1, .., x_n), f(y_1, .., y_n) \in Ts$ **do**
  $G := G \wedge en(x_1 \approx y_1 \wedge .. \wedge x_n \approx y_n \Rightarrow f(x_1, .., x_n) \approx f(y_1, .., y_n))$
foreach $t_1, t_2, t_3 \in Ts$ **do**
  $G := G \wedge en(t_1 \approx t_2 \wedge t_2 \approx t_3 \Rightarrow t_1 \approx t_3)$
Let G' be obtained by substituting each atom by a fresh Boolean variable in G.
return *checkBooleanSAT(G')*

en is naturally extended on formulas
e.g., $en(t_1 \approx t_2) = (en(t_1) \approx en(t_2))$

Eager approach may produce too many consequences, tomorrow you will see a not so eager approach.

Example: Ackermann's Reduction

Example 2.2

Consider formula $F = f(f(x)) \not\approx x \wedge f(x) \approx x$

$Ts := \{f(f(x)), f(x), x\}.$

$en := \{f(f(x)) \mapsto f_1, f(x) \mapsto f_2, x \mapsto f_3\}$

$G := en(F) := f_1 \not\approx f_3 \wedge f_2 \approx f_3$

Adding congruence consequences:
$G := G \wedge (f_2 \approx f_3 \Rightarrow f_1 \approx f_2).$

Adding transitivity consequences:
$G := G \wedge (f_1 \approx f_2 \wedge f_2 \approx f_3 \Rightarrow f_1 \approx f_3).$

Since G' *is UNSAT,* F *is UNSAT.*

Boolean encoding:
$\{f_1 \approx f_3 \mapsto p_1, f_2 \approx f_3 \mapsto p_2,$
$f_1 \approx f_3 \mapsto p_3\}$

$G' := \neg p_1 \wedge p_2$

$G' := G' \wedge (p_2 \Rightarrow p_3).$

$G' := G' \wedge (p_3 \wedge p_2 \Rightarrow p_1).$

Topic 2.2

Cooper's method

Reminder: Presburger arithmetic

Let us the following theory for arithmetic $\mathcal{T}_{\mathbb{Z}}$.

$\forall x \neg(x + 1 \approx 0)$

$\forall x \forall y(x + 1 \approx y + 1 \Rightarrow x \approx y)$

$F(0) \wedge (\forall x(F(x) \Rightarrow F(x + 1)) \Rightarrow \forall x F(x))$

$\forall x(x + 0 \approx x)$

$\forall x \forall y(x + (y + 1) \approx (x + y) + 1)$

Note that the theory does not have multiplication.

However, one can simulate multiplication by constants in the theory.

Cooper's method

Cooper's method is one of the well known decision procedure for Presburger arithmetic.

This method proceeds by quantifier elimination.

However, the arithmetic does not allow quantifier elimination as it is.

Example 2.3

The following formula states that y is odd.

$$\exists x.2x + 1 \approx y$$

This can not be stated in the arithmetic.

Adding | operator to enable quantifier elimination

We need to introduce modulo operator | that expresses divisibility.

$$k|y \text{ means } k \text{ divides } y, \text{ where } k \in \mathbb{Z}^+$$

We also need to add the following axiom about | in $\mathcal{T}_{\mathbb{Z}}$.

$$\forall y.\ k|y \Leftrightarrow \exists x.\ kx \approx y$$

Example 2.4

Now we can eliminate existential quantifier

$$(\exists x.2x + 1 \approx y) \equiv 2|(y + 1)$$

We may not write the parenthesis

Exercise 2.1

Give an x that satisfies the following constraints

$$2|x + 1 \wedge 3|x + 5 \wedge \neg 5|x - 2$$

Cooper's method

Input: $F_1 := \exists x. A_1 \wedge ... \wedge A_n$, where A_i is a literal.

The method proceeds in fours steps

- Normalize literals
- Separate out x
- Scale up coefficients of x
- Replace x with x' such that no coefficient to x'
- Eliminate x'

In some notation, we will use formulas like F_1 as set of literals.

Cooper's method : Normalize literals

The literals must be in one of the following forms

- $s < t$
- $k|t$
- $\neg(k|t)$

We may normalize literals as follows and obtain F_2

- $s = t \equiv s < t + 1 \wedge t < s + 1$
- $s \neq t \equiv s < t \vee t < s$
- $\neg s < t \equiv t < s + 1$

Example 2.5

Consider $F_1 := 3x \approx 6y + 3$

After normalization we obtain, $F_2 = 3x < 6y + 3 + 1 \wedge 3x + 1 > 6y + 3$

Cooper's method : separate out x

For $h \in \mathbb{Z}^+$ and t does not contain x, write terms in literals in F_2 until they are in one of the following forms.

- $hx < t$
- $t < hx$
- $k|hx + t$
- $\neg(k|hx + t)$

We obtain F_3 after this transformation.

Example 2.6

Consider $F_2 := 2x + 3y < 6 \wedge -2x + 3y < 6 \wedge 3| - 5x + 2.$

$$F_3 := 2x < 6 - 3y \wedge -6 + 3y < 2x \wedge 3|5x - 2.$$

Cooper's method : scale up coefficients of x

Let

$$\lambda = lcm\{h | h \text{ is coefficient of } x \text{ in some literal } \}$$

We scale up all literals in F_3 as follows and obtain F_4.

- $hx < t \equiv \lambda x < \lambda' t$
- $t < hx \equiv \lambda' t < \lambda x$
- $k|hx + t \equiv \lambda' k|\lambda x + \lambda' t$
- $\neg(k|hx + t) \equiv \neg(\lambda' k|\lambda x + \lambda' t)$

where $\lambda' h = \lambda$.

Example 2.7

Consider $F_3 = 2x < z + 1 \wedge y - 3 < 3x \wedge 4|5x + 1$.

$\lambda = lcm\{2, 3, 5\} = 30$.

Therefore, $F_4 = 30x < 15z + 15 \wedge 10y - 30 < 30x \wedge 24|30x + 6$.

Cooper's method : replace x to remove coefficient

We aim to remove coefficients of x.

We substitute λx by x' in the formula

We also need to say that x' is divisible by λ.

We obtain

$$F_5 := F_4[\lambda x \mapsto x'] \wedge \lambda | x'.$$

Example 2.8

$F_4 = 30x < 15z + 15 \wedge 10y - 30 < 30x \wedge 24 | 30x + 6.$

After replacement:

$F_5 = x' < 15z + 15 \wedge 10y - 30 < x' \wedge 24 | x' + 6 \wedge 30 | x'.$

Cooper's method : eliminate x'

- $M := \{A \in F_5 |\ A = (k|t) \text{ or } A = \neg(k|t)\}$.
- $UB := \{x' < t | x' < t \in F_5\}$,
- $LB := \{t < x' | t < x' \in F_5\}$, and
- $\delta := lcm\{k|\ (k|t) \text{ or } \neg(k|t) \text{ in } M\}$.

Now we have two cases.

- $LB = \emptyset$
- $LB \neq \emptyset$

case $LB = \emptyset$

- $\delta := lcm\{k|\ (k|t) \text{ or } \neg(k|t) \text{ in } M\}$,

Since there are no lower bounds in F_5, there is some x' that satisfies the upper bounds in F_5.

We only need to check that the mod literals are mutually satisfiable.

In every δ interval there must be a satisfying assignment.

Therefore, the following is an equivalent and quantifier-free formula.

$$F_6 := \bigvee_{i=1}^{\delta} M[x' \mapsto i]$$

Example : $LB = \emptyset$

Example 2.9

Consider the following formula with no lower bound:
$F_5 = x' < 15z + 15 \wedge 6|x' - y + 6 \wedge 9|x'$.

Since we can always choose small enough x' to satisfy $x' < 15z + 15$, we can ignore the literal.

$\delta = lcm\{6, 9\} = 18$.
In every interval of 18, one value of x' must satisfy the mod literals.

Therefore, the following is an equivalent and quantifier-free formula.

$$\bigvee_{i=1}^{18} 6|i - y + 6 \wedge 9|i$$

Exercise 2.2

Simplify the above formula

case $LB \neq \emptyset$

- $\delta := lcm\{k|\ (k|t)$ or $\neg(k|t)$ in $F_5\}$,

Let us suppose $x' = m$ satisfies F_5 then m is larger than the largest lower bound.

Let $x' < t \in LB$ be the largest lower bound.

- $LB[x' \mapsto t+1]$ is true
- Since there is a satisfying assignment, $UB[x' \mapsto t+1]$ is true. Furthermore, there is m such that
  - $UB[x' \mapsto t+i]$ is true for $1 \leq i \leq m$
  - $UB[x' \mapsto t+i]$ is false for $i > m$
- One of $M[x' \mapsto t+1], .., M[x' \mapsto t+\delta]$ must be true (same argument as previous case)

Therefore, one of the disjuncts in the following formula must be true.

$$F_6 := \bigvee_{t<x' \in LB} \bigvee_{i=1}^{\delta} F_5[t+j]$$

Example : $LB \neq \emptyset$

Example 2.10

Consider the following formula with lower bounds:
$F_5 = x' < 15z + 15 \wedge 10y - 30 < x' \wedge 24|x' + 6 \wedge 30|x'.$

$\delta = lcm(24, 30) = 120$

Since $LB := \{10y - 30 < x'\}$

$$F_6 := \bigvee_{i=1}^{120} 10y - 30 + i < 15z + 15 \wedge 10y - 30 < 10y - 30 + i \wedge 24|10y - 30 + i + 6 \wedge 30|10y - 30 + i$$

After simplification,
$F_6 := \bigvee_{i=1}^{120} 10y - 45 + i < 15z \wedge 24|10y - 24 + i \wedge 30|10y - 30 + i$

Topic 2.3

Theory of linear rational arithmetic

Theory of rational numbers $\mathcal{T}_{\mathbb{Q}}$

Signature $\mathbf{S} = (\{0/0, 1/1, +/2, -/1, \}, \{\geq /2\})$ (no multiplication.)

A axiomatization of $\mathcal{T}_{\mathbb{Q}}$ is

1. $\forall x, y, z.\ (x + y) + z \approx x + (y + z)$
2. $\forall x, y.\ x + y \approx y + x$
3. $\forall x.\ x + 0 \approx x$
4. $\forall x.\ x + (-x) \approx 0$
5. $1 \geq 0$
6. $1 \not\approx 0$
7. $\forall x, y.\ x \geq y \wedge y \geq x \Rightarrow x \approx y$
8. $\forall x, y, z.\ x \geq y \wedge y \geq z \Rightarrow x \geq z$
9. $\forall x, y.\ x \geq y \vee y \geq x$
10. $\forall x, y, z.\ x \geq y \Rightarrow x + z \geq y + z$
11. For every n, $\forall x, y, z.\ x = \underbrace{y + \ldots + y}_{n}$

The above theory is decidable.

Solvers for $\mathcal{T}_{\mathbb{Q}}$

We will present decision procedures for the increasingly wider fragments of $\mathcal{T}_{\mathbb{Q}}$.

- Quantifier-free difference logic
- Quantifier-free octagonal constraints
- Quantifier-free linear arithmetic

Topic 2.4

Difference logic

Difference Logic

Difference Logic over the integers(QF_IDL):
Boolean combinations of inequalities of the form $x - y \leq b$, where x and y are integer variables and b is an integer constant.

Difference Logic over the rationals(QF_RDL):
Boolean combinations of inequalities of the form $x - y \leq b$, where x and y are rational variables and b is an rational constant.

Widely used in analysis of timed systems for comparing clocks.

We will present an $O(n^3)$ method to decide conjunction of literals in QF_RDL and QF_IDL.

Difference Graph

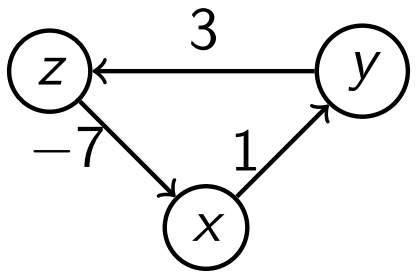
We may view an atom $x - y \leq b$ as a weighted directed edge between two nodes x and y with weight b in graph over variables. This graph is called difference graph.

Theorem 2.1

A conjunction of difference inequalities is unsatisfiable iff the corresponding difference graph has negative cycles.

Example 2.11

$$x - y \leq 1 \wedge y - z \leq 3 \wedge z - x \leq -7$$



Difference bound matrix

Another view of difference graph.

Definition 2.1

Let F be conjunction of difference inequalities over rational variables $\{x_1, \ldots, x_n\}$. The difference bound matrix(DBM) A is defined as follows.

$$A_{ij} = \begin{cases} 0 & i = j \\ b & x_i - x_j \leq b \in F \\ \infty & \textit{otherwise} \end{cases}$$

Let $F[A] \triangleq \bigwedge_{i,j \in 1..n} x_i - x_j \leq A_{ij}$.

Let $A_{i_0 \ldots i_m} \triangleq \sum_{k=1}^{m} A_{i_{k-1} i_k}$.

Example: DBM

Example 2.12

Consider:

$$x_2 - x_1 \leq 4 \wedge x_1 - x_2 \leq -1 \wedge x_3 - x_1 \leq 3 \wedge x_1 - x_3 \leq -1 \wedge x_2 - x_3 \leq 1$$

Constraints has three variables x_1, x_2, and x_3.

The corresponding DBM is

$$\begin{bmatrix} 0 & -1 & -1 \\ 4 & 0 & \_\_\_ \\ 3 & \_\_\_ & 0 \end{bmatrix}$$

Exercise 2.3

Fill the blanks

Shortest path closure

Definition 2.2

The shortest path closure $A^\bullet$ of A is defined as follows.

$$(A^\bullet)_{ij} = \min_{i=i_0,i_1,\ldots,i_m=j \text{ and } m\leq n} A_{i_0\ldots i_m}$$

Theorem 2.2

F is unsatisfiable iff $\exists i \in 1..n.\ A^\bullet_{ii} < 0$

Exercise 2.4 *If F is sat, $A^\bullet_{ij} \leq A^\bullet_{ikj}$.*

Implication checking and canonical form

Definition 2.3

A set of objects R represents a class of formulas Σ canonically if for each $F, F' \in \Sigma$ if $F \equiv F'$ and $o \in R$ represents F then o represents F'.

Theorem 2.3

The set of shortest path closed DBMs canonically represents difference logic formulas.

Exercise 2.5

Give an efficient method of checking equisatisfiablity and implication using DBMs.

Floyd-Warshall Algorithm for shortest closure

We can compute $A^\bullet$ using the following iterations generating $A^0, \ldots, A^n$.

$$A^0 = A$$
$$A^k_{ij} = \min(A^{k-1}_{ij}, A^{k-1}_{ikj})$$

Theorem 2.4

$A^\bullet = A^n$

Exercise 2.6

a. Extend the above algorithm to support strict inequalities

b. Does the above algorithm also works for $\mathbb{Z}$?

Example: DBM

Example 2.13

Consider DBM: $A^0 = \begin{bmatrix} 0 & -1 & -1 \\ 4 & 0 & 1 \\ 3 & \infty & 0 \end{bmatrix}$

Apply first iteration:

$$A^1 = min(A^0, \begin{bmatrix} A^0_{111} & A^0_{112} & A^0_{113} \\ A^0_{211} & A^0_{212} & A^0_{213} \\ A^0_{311} & A^0_{312} & A^0_{313} \end{bmatrix}) = min(A^0, \begin{bmatrix} 0 & -1 & -1 \\ 4 & 3 & 3 \\ 3 & 2 & 2 \end{bmatrix}) = \begin{bmatrix} 0 & -1 & -1 \\ 4 & 0 & 1 \\ 3 & 2 & 0 \end{bmatrix}$$

Apply second iteration:

$$A^2 = min(A^1, \begin{bmatrix} A^1_{121} & A^1_{122} & A^1_{123} \\ A^1_{221} & A^1_{222} & A^1_{223} \\ A^1_{321} & A^1_{322} & A^1_{323} \end{bmatrix}) = min(A^1, \begin{bmatrix} 3 & -1 & 0 \\ 4 & 0 & 1 \\ 6 & 2 & 2 \end{bmatrix}) = \begin{bmatrix} 0 & -1 & -1 \\ 4 & 0 & 1 \\ 3 & 2 & 0 \end{bmatrix}$$

Apply third iteration:

$$A^3 = min(A^2, \begin{bmatrix} A^2_{131} & A^2_{132} & A^2_{133} \\ A^2_{231} & A^2_{232} & A^2_{233} \\ A^2_{331} & A^2_{332} & A^2_{333} \end{bmatrix}) = min(A^2, \begin{bmatrix} 2 & 1 & -1 \\ 4 & 3 & 1 \\ 3 & 2 & 0 \end{bmatrix}) = \begin{bmatrix} 0 & -1 & -1 \\ 4 & 0 & 1 \\ 3 & 2 & 0 \end{bmatrix}$$

Topic 2.5

Octagonal constraints

Octagonal constraints

Definition 2.4

Octagonal constraints *are boolean combinations of inequalities of the form* $\pm x \pm y \leq b$ *or* $\pm x \leq b$ *where* x *and* y *are* $\mathbb{Z}/\mathbb{Q}$ *variables and* b *is an* $\mathbb{Z}/\mathbb{Q}$ *constant.*

We can always translate octagonal constraints into equisatisfiable difference constraints.

Octagon to difference logic encoding (contd.)

Consider conjunction of octagonal atoms F over variables $V = \{x_1, \ldots, x_n\}$.

We construct a difference logic formula F' over variables $V' = \{x'_1, \ldots, x'_{2n}\}$.

In the encoding, x'_{2i-1} represents x_i and x'_{2i} represents $-x_i$.

Octagon to difference logic encoding

F' is constructed as follows

$$
\begin{array}{lrcll}
F \ni & x_i \leq b & \rightsquigarrow & x'_{2i-1} - x'_{2i} \leq 2b & \in F' \\
F \ni & -x_i \leq b & \rightsquigarrow & x'_{2i} - x'_{2i-1} \leq 2b & \in F' \\
F \ni & x_i - x_j \leq b & \rightsquigarrow & x'_{2i-1} - x'_{2j-1} \leq b,\ x'_{2j} - x'_{2i} \leq b & \in F' \\
F \ni & x_i + x_j \leq b & \rightsquigarrow & x'_{2i-1} - x'_{2j} \leq b, \quad x'_{2j-1} - x'_{2i} \leq b & \in F' \\
F \ni & -x_i - x_j \leq b & \rightsquigarrow & x'_{2i} - x'_{2j-1} \leq b, \quad x'_{2j} - x'_{2i-1} \leq b & \in F'
\end{array}
$$

Definition 2.5

The DBM corresponding to F' are called octagonal DBMs(ODBMs).

Theorem 2.5

F and F' are equisatisfiable.

Example: octagonal DBM

Exercise 2.7

Consider:

$$x_1 + x_2 \leq 4 \wedge x_2 - x_1 \leq 5 \wedge x_1 - x_2 \leq 3 \wedge -x_1 - x_2 \leq 1 \wedge x_2 \leq 2 \wedge -x_2 \leq 7$$

Corresponding ODBM

$$\begin{bmatrix} 0 & \infty & 3 & 4 \\ \infty & 0 & 1 & 5 \\ 5 & 4 & 0 & 4 \\ 1 & 3 & 14 & 0 \end{bmatrix}$$

$x_1 + x_2 \leq 4 \rightsquigarrow x_1 - x_4 \leq 4, x_3 - x_2 \leq 4$

$x_2 - x_1 \leq 5 \rightsquigarrow x_3 - x_1 \leq 5, x_2 - x_4 \leq 5$

$x_1 - x_2 \leq 3 \rightsquigarrow x_1 - x_3 \leq 3, x_4 - x_2 \leq 3$

$-x_1 - x_2 \leq 1 \rightsquigarrow x_1 - x_4 \leq 1, x_3 - x_2 \leq 1$

$x_2 \leq 2 \rightsquigarrow x_3 - x_4 \leq 4$

$-x_2 \leq 7 \rightsquigarrow x_3 - x_4 \leq 14$

Unsatisfiability

For $\mathbb{Q}$, the method of checking unsatisfiability of difference constraints will work on ODBMs.

Floyd-Warshall Algorithm on ODBM of F with n variables will let us know in $2n$ steps if F is sat.

Topic 2.6

Simplex

Incremental simplex as solver for $\mathcal{T}_{\mathbb{Q}}$

We will present simplex algorithm that is a solver for quantifier-free fragment of $\mathcal{T}_{\mathbb{Q}}$ (QF_LRA).

Similar to the Cooper's method, we assume that the input formula is conjunction of literals.

We further restrict our presentation by only considering formulas of the following form.

$$a_{11}x_1 + ...a_{1n}x_n \leq b_i \wedge .. \wedge a_{m1}x_1 + ...a_{mn}x_n \leq b_m$$

We may write the above formula in matrix notation

$$Ax \leq b,$$

where A is a $m \times n$ matrix.

Simplex - notation

Consider constraints

$$Ax \leq b.$$

By introducing fresh variables, we transform the above into

$$\begin{bmatrix} -\mathtt{I} & A \end{bmatrix} \begin{bmatrix} s \\ x \end{bmatrix} = 0 \text{ and } s \leq b.$$

s are called slack variables. Since there is no reason to distinguish x and s in simplex, A will refer to $\begin{bmatrix} -\mathtt{I} & A \end{bmatrix}$ and x will refer to $\begin{bmatrix} s \\ x \end{bmatrix}$.

The above constraints will be denoted by

$$Ax = 0 \text{ and } \bigwedge_{i=1}^{m+n} l_i \leq x_i \leq u_i.$$

l_i and u_i are $+\infty$ and $-\infty$ if there is no lower and upper bound, respectively.

Simplex - notation (contd.)

$$Ax = 0 \text{ and } \bigwedge_{i=1}^{m+n} l_i \leq x_i \leq u_i$$

- A is $m \times (m+n)$ matrix.
- Since $Ax = 0$ defines an n-dim subspace in $(m+n)$-dim space, if we choose values of n variables then we fix values of the other m variables.
- We will refer to ith column of A as the column corresponding to x_i.

Example 2.14

Consider: $-x + y \leq -2 \wedge x \leq 3$

We have slack variables s_1 and s_2. In matrix form,

$$\left[\begin{array}{cc:cc} -1 & 0 & -1 & 1 \\ 0 & -1 & 1 & 0 \end{array}\right] \left[\begin{array}{c} s_1 \\ s_2 \\ \hdashline x \\ y \end{array}\right] = 0 \qquad \begin{array}{c} s_1 \leq -2 \\ s_2 \leq 3 \end{array}$$

Simplex - basic and nonbasic variables

Definition 2.6

Simplex assumes all the columns of $-\mathtt{I}$ occur in A.The variables corresponding to the columns are called basic variables. Others are called nonbasic variables.

Note: the assumption is true about initial A

Definition 2.7

Let B be the set of indexes for basic variables and $NB \triangleq 1..n - B$. For $j \in B$, let k_j be a row s.t. $A_{k_j j} = -1$.

Example 2.15

$$\begin{bmatrix} -1 & 0 & -1 & 1 \\ 0 & -1 & 1 & 0 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ x \\ y \end{bmatrix} = 0 \qquad \begin{matrix} s_1 \leq -2 \\ s_2 \leq 3 \end{matrix}$$

Currently, s_1 and s_2 are basic and x and y are nonbasic.
$B = \{1, 2\}$, $NB = \{3, 4\}$, $k_1 = 1$, and $k_2 = 2$.

Simplex - current assignment

Definition 2.8

Simplex maintains current assignment $v : x \to \mathbb{Q}$ *s.t.*

- $Av = 0$,
- *nonbasic variables satisfy their bounds, and,*
- *consequently values for basic variables in* v *are fixed and* v *may violate a bound of at most one basic variable.*

Explained later why "at most" one

Example 2.16

$$\left[\begin{array}{cc:cc} -1 & 0 & -1 & 1 \\ 0 & -1 & 1 & 0 \end{array}\right] \left[\begin{array}{c} s_1 \\ s_2 \\ \hdashline x \\ y \end{array}\right] = 0$$

Currently violated

$s_1 \leq -2$

$s_2 \leq 3$

Initially, $v = \{\underbrace{x \mapsto 0, y \mapsto 0}, s_1 \mapsto 0, s_2 \mapsto 0\}$

Choose values for nonbasic variables, others follow!

Simplex - state

For variable $i \in NB$, $v(x_i)$ is equal to one of the existing bounds of x_i and if no bounds exists for x_i then $v(x_i) = 0$.

Definition 2.9

A bound on x_i is called active if $v(x_i)$ is equal to the bound.
We will mark the active bounds by *.

Definition 2.10

The NB set and bound activity defines the current state of simplex.

Example 2.17

$$\begin{bmatrix} -1 & 0 & -1 & 1 \\ 0 & -1 & 1 & 0 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ x \\ y \end{bmatrix} = 0 \qquad \begin{matrix} s_1 \leq -2 \\ s_2 \leq 3 \end{matrix}$$

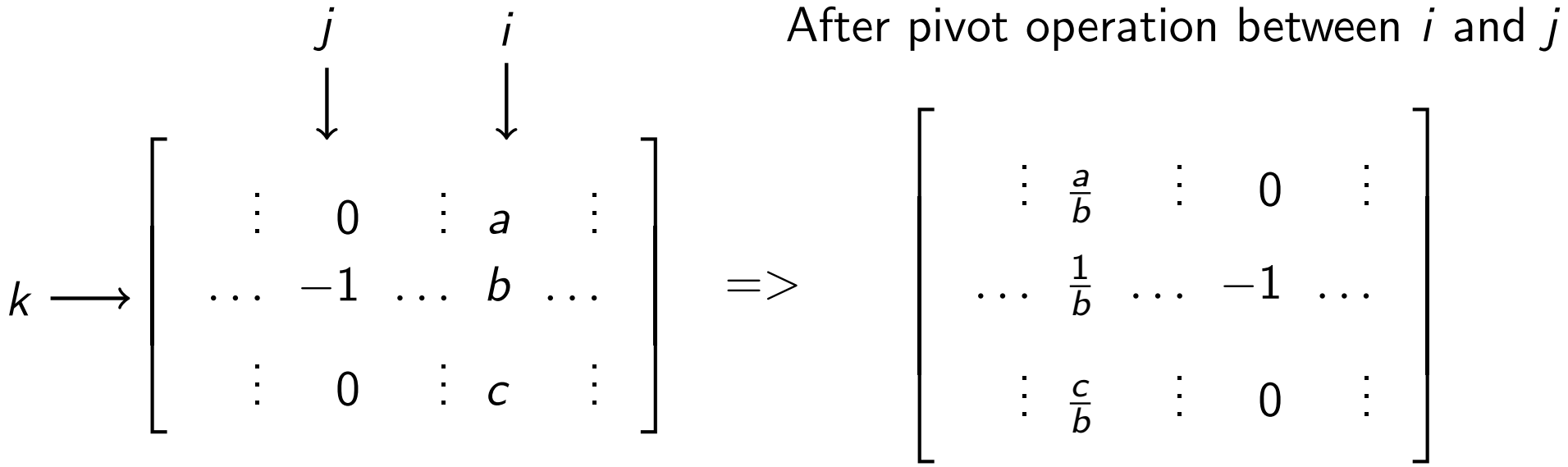
Since all nonbasic variables have no bounds, no bound is marked active.

Simplex - pivot operation

If v violates a bound constraint of some basic variable, then simplex correct it by applying pivot operation

Definition 2.11

Let us suppose x_j is basic, column j has -1 at row k, and x_i is nonbasic. A pivot operation between i and j exchanges the roll between x_i and x_j. The pivot operation applies row operations until column i has -1 at row k and all other entries in the column are zero.



$$k \longrightarrow \begin{bmatrix} \vdots & 0 & \vdots & a & \vdots \\ \dots & -1 & \dots & b & \dots \\ \vdots & 0 & \vdots & c & \vdots \end{bmatrix} \Rightarrow \begin{bmatrix} \vdots & \frac{a}{b} & \vdots & 0 & \vdots \\ \dots & \frac{1}{b} & \dots & -1 & \dots \\ \vdots & \frac{c}{b} & \vdots & 0 & \vdots \end{bmatrix}$$

Simplex - choosing non-basic column for pivot

Wlog, let $1 \in B$, $k_1 = 1$, and $v(x_1)$ violates u_1. We need to decrease $v(x_1)$.
We call $v(x_1) - u_1$ violation difference.
Since $x_1 = \sum_{i \in NB} a_{1i} x_i$, we need to change $v(x_i)$ of some x_i s.t. $a_{1i} x_i$ decreases

Definition 2.12

A column $i \in NB$ is suitable if

- x_i *is unbounded,*
- $x_i = u_i$ *and* $a_{1i} > 0$, *or*
- $x_i = l_i$ *and* $a_{1i} < 0$.

i is selected suitable column if i is the smallest suitable column.

Example 2.18

$$\begin{bmatrix} -1 & 0 & -1 & 1 \\ 0 & -1 & 1 & 0 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ x \\ y \end{bmatrix} = 0$$

$s_1 \leq -2$
$s_2 \leq 3$

Exercise 2.8

Write other cases that are ignored due to "wlog"

Column 3 and 4 are suitable.

Simplex - choosing basic column for pivot

v satisfies all bounds except u_1. Change in $v(x_i)$ may lead to violations, because x_i appears in the definitions of basic variables.

Wlog, let x_i is unbounded and $a_{1i} < 0$. Therefore, we need to increase $v(x_i)$.

What are the other cases?

Definition 2.13

We need to find the maximum allowed change.

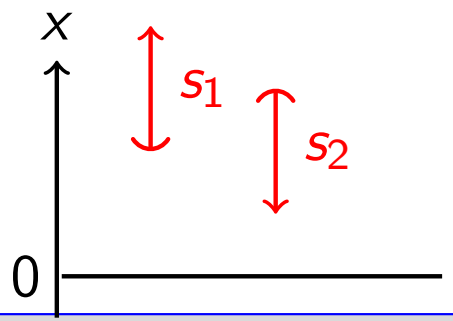
$$ch := min \bigcup_{j \in B} \{\frac{v(x_j) - u_j}{a_{k_j i}} | a_{k_j i} < 0\} \cup \{\frac{v(x_j) - l_j}{a_{k_j i}} | a_{k_j i} > 0\}$$

Let j be the smallest index for which the above min is attained.

Example 2.19

We change x(selected suitable column) to reduce violation difference.

Since $v(y) = 0$ and we are varying x, $s_1 = -x$ and $s_2 = x$.
We have $s_1 \leq -2$, and $s_2 \leq 0$.
Therefore, s_1 allows $2 \leq x$ and s_2 allows $x \leq 3$.
Clearly, $ch = 3$ and $j = 2$.



Simplex - pivoting operation to reduce violation difference

We carry ch and j from the last slide. Wlog, $ch = \frac{v(x_j)-u_j}{a_{k_j i}}$.

Now there are three possibilities

- If $ch = u_i = +\infty$, pivot between i and 1 and activate u_1
- If $ch > (u_i - l_i)$, we assign $v(x_i) = l_i$ update values of basic variables and no pivoting
- Otherwise, we choose j for the basic variable for pivoting and apply pivoting between i and j. We activate u_j bound on variable x_j.

If violation still persists then look for further pivot operations.

Theorem 2.6

Pivoting operation never increases violation difference

Example 2.20

After pivoting between 3 *and* 2.

$$\begin{bmatrix} -1 & -1 & 0 & 1 \\ 0 & 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ x \\ y \end{bmatrix} = 0 \qquad \begin{aligned} s_1 &\leq -2 \\ s_2 &\leq 3^* \end{aligned}$$

Now v is satisfying.

Incremental simplex and single violation assumption

Before adding next atom, incremental simplex finds a solution of atoms added so far.

New atom $ax \leq b$ is added in the following steps.

- A fresh slack variable s is introduced
- $s = ax$ is added as a row in A and $s \leq b$ is added in the bounds
- The new row may have non-zeros in basic columns. They are removed by row operations on the new row.
- s is added to B.

Therefore, current assignment can only violate the bound of s.

Example: inserting a new atom

Example 2.21

Let us add atom $-2x - y \leq -8$ *in our running example*
We add a new slack variable s_3 *and a corresponding row.*

$$\begin{bmatrix} -1 & 0 & 0 & -2 & -1 \\ 0 & -1 & -1 & 0 & 1 \\ 0 & 0 & 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} s_3 \\ s_1 \\ s_2 \\ x \\ y \end{bmatrix} = 0 \qquad \begin{aligned} s_3 &\leq -8 \\ s_1 &\leq -2 \\ s_2 &\leq 3^* \end{aligned}$$

After remove basic variables $(\{s_1, x\})$ *from the top row*

$$\begin{bmatrix} -1 & 0 & -2 & 0 & -1 \\ 0 & -1 & -1 & 0 & 1 \\ 0 & 0 & 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} s_3 \\ s_1 \\ s_2 \\ x \\ y \end{bmatrix} = 0 \qquad \begin{aligned} s_3 &\leq -8 \\ s_1 &\leq -2 \\ s_2 &\leq 3^* \end{aligned}$$

Exercise 2.9

Now s_3 *is violated. Pivot if possible.*

Simplex - iterations

Simplex is a sequence of pivot operations

- If simplex fails to find a suitable column for some violation then input is unsat.
- If a state is reached without violation then v is a satisfying assignment.

Example 2.22

s_3 is still in violation.

$$\begin{bmatrix} -1 & -1 & -3 & 0 & 0 \\ 0 & 1 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} s_3 \\ s_1 \\ s_2 \\ x \\ y \end{bmatrix} = 0 \qquad \begin{aligned} s_3 &\leq -8 \\ s_1 &\leq -2^* \\ s_2 &\leq 3^* \end{aligned}$$

Now, we can not find a suitable column.

Therefore, the constraints are unsat.

Example 2.23

Run simplex on $x_1 \leq 5 \wedge 4x_1 + x_2 \leq 25 \wedge -2x_1 - x_2 \leq -25$

After push of the first atom

$$\begin{bmatrix} -1 & 1 \end{bmatrix}\begin{bmatrix} s_1 \\ x_1 \end{bmatrix} = 0 \qquad s_1 \leq 5 \qquad v = \{x_1 \mapsto 0, s_1 \mapsto 0\}$$

After push of the second atom

$$\begin{bmatrix} -1 & 0 & 4 & 1 \\ 0 & -1 & 1 & 0 \end{bmatrix}\begin{bmatrix} s_2 \\ s_1 \\ x_1 \\ x_2 \end{bmatrix} = 0 \qquad \begin{matrix} s_1 \leq 5 \\ s_2 \leq 25 \end{matrix} \qquad v = \{\_ \mapsto 0\}$$

After push of the last atom

$$\begin{bmatrix} -1 & 0 & 0 & -2 & -1 \\ 0 & -1 & 0 & 4 & 1 \\ 0 & 0 & -1 & 1 & 0 \end{bmatrix}\begin{bmatrix} s_3 \\ s_2 \\ s_1 \\ x_1 \\ x_2 \end{bmatrix} = 0 \qquad \begin{matrix} s_1 \leq 5 \\ s_2 \leq 25 \\ s_3 \leq -25 \end{matrix} \qquad v = \{\_ \mapsto 0\}$$

Exercise 2.10

Finish the run