universität innsbruck

SS 2024 lecture 7

Constraint Solving

René Thiemann and Fabian Mitterwallner

based on a previous course by Aart Middeldorp

Outline

1. **Summary of Previous Lecture**
2. **Complexity of Simplex Algorithm**
3. **Unsatisfiable Cores and Farkas' Lemma**
4. **Simplex Algorithm for DPLL($\mathcal{T}$)**
5. **Further Reading**

Difference Logic

conjunction of constraints of the form $x - y \leqslant c$ or $x - y < c$

Definition Inequality Graph

conjunction φ of nonstrict difference constraints

- inequality graph of φ contains edge from $x \xrightarrow{c} y$ 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

Bellman-Ford Algorithm

computes distances in graphs from single source; detects negative cycles

Simplex – Representation

- represent m inequalities using m slack variables s_i and bounds $s_i \leq / \geq c$

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

- 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 $\rightarrow$

meaning of rows:

equalities, e.g.,

$s_4 = 3x - 1y$

Notation

- matrix is **tableau**, stored in combination with **bounds** $x \leq / \geq c$ and **assignment**
- B is set of **basic variables** (in tableau listed vertically)
- N is set of **nonbasic variables** (in tableau listed horizontally)

DPLL(T) Simplex Algorithm

Input: conjunction of LRA atoms φ without $<$
Output: satisfiable assignment or unsatisfiable

1. transform φ into tableau and bounds
2. 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

DPLL(T) Simplex Algorithm

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

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

$$\begin{array}{cc} & \text{nonbasic } \vec{x}_N \\ & x_j \\ \text{basic } \vec{x}_B \quad x_i & \begin{pmatrix} \dots & & \dots \\ & A_{ij} & \\ \dots & & \dots \end{pmatrix} \end{array}$$

Invariant

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

Suitability

- for $x_i \in B$ violating lower or upper bound, find suitable non-basic variable x_j such that increase (or decrease) of x_j is possible w.r.t. bounds of x_j and helps to solve violation of x_i

Pivoting

- swap basic x_i and nonbasic x_j, so $i \in B$ and $j \in N$
- reorder row i in tableau to obtain form $x_j = \ldots$ ($\star$), and substitute ($\star$) in remaining tableau
- result afterwards: tableau A' where $j \in B$ and $i \in N$

Update

- assignment of x_i is updated to previously violated bound l_i or u_i,
- assignment of each $x_k \in B$ is recomputed using A'

$\mathbb{Q}_\delta$: δ-Rationals

- δ-rationals are used for supporting strict inequalities in LRA and difference logic

$$\text{replace} \quad \mathit{expr} < c \quad \text{by} \quad \mathit{expr} \leq c - \delta$$

- δ represents some small positive rational number
- computation on $\mathbb{Q}_\delta$ is done symbolically, e.g., in simplex algorithm
- after solution for $\mathbb{Q}_\delta$ is detected, a concrete δ can be computed (exercise)

Outline

1. **Summary of Previous Lecture**
2. **Complexity of Simplex Algorithm**
3. **Unsatisfiable Cores and Farkas' Lemma**
4. **Simplex Algorithm for DPLL($\mathcal{T}$)**
5. **Further Reading**

Complexity of DPLL(T) Simplex Algorithm

- input: m inequalities using n problem variables

Complexity of DPLL(T) Simplex Algorithm

- input: m inequalities using n problem variables
- switch to general form: m basic variables, n nonbasic variables

Complexity of DPLL(T) Simplex Algorithm

- input: m inequalities using n problem variables
- switch to general form: m basic variables, n nonbasic variables
- number of different tableaux: $\binom{m+n}{n}$

Complexity of DPLL(T) Simplex Algorithm

- input: m inequalities using n problem variables
- switch to general form: m basic variables, n nonbasic variables
- number of different tableaux: $\binom{m+n}{n}$
- number of different configurations: $\binom{m+n}{n} \cdot 3^n$
  (each nonbasic variable gets assigned 0, lower bound, or upper bound)

Complexity of DPLL(T) Simplex Algorithm

- input: m inequalities using n problem variables
- switch to general form: m basic variables, n nonbasic variables
- number of different tableaux: $\binom{m+n}{n}$
- number of different configurations: $\binom{m+n}{n} \cdot 3^n$
  (each nonbasic variable gets assigned 0, lower bound, or upper bound)
- consequences

Complexity of DPLL(T) Simplex Algorithm

- input: m inequalities using n problem variables
- switch to general form: m basic variables, n nonbasic variables
- number of different tableaux: $\binom{m+n}{n}$
- number of different configurations: $\binom{m+n}{n} \cdot 3^n$
  (each nonbasic variable gets assigned 0, lower bound, or upper bound)
- consequences
  - bad news 1: assuming termination, obtain exponential worst-case complexity

Complexity of DPLL(T) Simplex Algorithm

- input: m inequalities using n problem variables
- switch to general form: m basic variables, n nonbasic variables
- number of different tableaux: $\binom{m+n}{n}$
- number of different configurations: $\binom{m+n}{n} \cdot 3^n$
  (each nonbasic variable gets assigned 0, lower bound, or upper bound)
- consequences
  - bad news 1: assuming termination, obtain exponential worst-case complexity
  - bad news 2: simplex algorithm does not terminate in general

Complexity of DPLL(T) Simplex Algorithm

- input: m inequalities using n problem variables
- switch to general form: m basic variables, n nonbasic variables
- number of different tableaux: $\binom{m+n}{n}$
- number of different configurations: $\binom{m+n}{n} \cdot 3^n$
  (each nonbasic variable gets assigned 0, lower bound, or upper bound)
- consequences
  - bad news 1: assuming termination, obtain exponential worst-case complexity
  - bad news 2: simplex algorithm does not terminate in general
  - good news 1: simplex algorithm terminates using Bland's rule

Complexity of DPLL(T) Simplex Algorithm

- input: m inequalities using n problem variables
- switch to general form: m basic variables, n nonbasic variables
- number of different tableaux: $\binom{m+n}{n}$
- number of different configurations: $\binom{m+n}{n} \cdot 3^n$
  (each nonbasic variable gets assigned 0, lower bound, or upper bound)
- consequences
  - bad news 1: assuming termination, obtain exponential worst-case complexity
  - bad news 2: simplex algorithm does not terminate in general
  - good news 1: simplex algorithm terminates using Bland's rule
  - good news 2: worst-case complexity rarely observed, often only $\mathcal{O}(m)$ many iterations

Complexity of DPLL(T) Simplex Algorithm

- input: m inequalities using n problem variables
- switch to general form: m basic variables, n nonbasic variables
- number of different tableaux: $\binom{m+n}{n}$
- number of different configurations: $\binom{m+n}{n} \cdot 3^n$
  (each nonbasic variable gets assigned 0, lower bound, or upper bound)
- consequences
  - bad news 1: assuming termination, obtain exponential worst-case complexity
  - bad news 2: simplex algorithm does not terminate in general
  - good news 1: simplex algorithm terminates using Bland's rule
  - good news 2: worst-case complexity rarely observed, often only $\mathcal{O}(m)$ many iterations

Bland's Rule

- in pivoting pick lexicographically smallest $(x_i, x_j) \in B \times N$ such that x_i and x_j are suitable; assumes some fixed order on variables

Example (Felgenhauer and Middeldorp)

$-1 \leqslant x_1 \leqslant 0 \qquad -4 \leqslant x_2 \leqslant 0 \qquad -5 \leqslant x_3 \leqslant -4 \qquad -7 \leqslant x_4 \leqslant 1$

$$\begin{array}{cc} & \begin{array}{cc} x_1 & x_2 \end{array} \\ \begin{array}{c} x_3 \\ x_4 \end{array} & \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix} \end{array}$$

Example (Felgenhauer and Middeldorp)

$$-1 \leqslant x_1 \leqslant 0 \qquad -4 \leqslant x_2 \leqslant 0 \qquad -5 \leqslant x_3 \leqslant -4 \qquad -7 \leqslant x_4 \leqslant 1$$

$$\begin{array}{c c} & \begin{array}{cc} x_1 & x_2 \end{array} \\ \begin{array}{c} x_3 \\ x_4 \end{array} & \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix} \end{array}$$

x_1	x_2	x_3	x_4
0	0	0	0

Example (Felgenhauer and Middeldorp)

$$-1 \leqslant x_1 \leqslant 0 \qquad -4 \leqslant x_2 \leqslant 0 \qquad -5 \leqslant x_3 \leqslant -4 \qquad -7 \leqslant x_4 \leqslant 1$$

$$\begin{array}{c c} & \begin{array}{cc} x_1 & x_2 \end{array} \\ \begin{array}{c} x_3 \\ x_4 \end{array} & \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix} \end{array}$$

x_1	x_2	x_3	x_4
0	0	0	0

Example (Felgenhauer and Middeldorp)

$$-1 \leqslant x_1 \leqslant 0 \qquad -4 \leqslant x_2 \leqslant 0 \qquad -5 \leqslant x_3 \leqslant -4 \qquad -7 \leqslant x_4 \leqslant 1$$

$$\begin{array}{cc} & \begin{array}{cc} x_1 & x_2 \end{array} \\ \begin{array}{c} x_3 \\ x_4 \end{array} & \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix} \end{array} \qquad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline 0 & 0 & 0 & 0 \end{array}$$

$$\downarrow$$

$$\begin{array}{cc} & \begin{array}{cc} x_3 & x_2 \end{array} \\ \begin{array}{c} x_1 \\ x_4 \end{array} & \begin{pmatrix} 1 & -2 \\ 2 & -3 \end{pmatrix} \end{array} \qquad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline -4 & 0 & -4 & -8 \end{array}$$

Example (Felgenhauer and Middeldorp)

$$-1 \leqslant x_1 \leqslant 0 \qquad -4 \leqslant x_2 \leqslant 0 \qquad -5 \leqslant x_3 \leqslant -4 \qquad -7 \leqslant x_4 \leqslant 1$$

$$\begin{array}{c c} & \begin{array}{cc} x_1 & x_2 \end{array} \\ \begin{array}{c} x_3 \\ x_4 \end{array} & \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix} \end{array} \qquad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline 0 & 0 & 0 & 0 \end{array}$$

$$\downarrow$$

$$\begin{array}{c c} & \begin{array}{cc} x_3 & x_2 \end{array} \\ \begin{array}{c} x_1 \\ x_4 \end{array} & \begin{pmatrix} 1 & -2 \\ 2 & -3 \end{pmatrix} \end{array} \qquad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline -4 & 0 & -4 & -8 \end{array}$$

Example (Felgenhauer and Middeldorp)

$$-1 \leqslant x_1 \leqslant 0 \qquad -4 \leqslant x_2 \leqslant 0 \qquad -5 \leqslant x_3 \leqslant -4 \qquad -7 \leqslant x_4 \leqslant 1$$

$$\begin{array}{cc} & \begin{array}{cc} x_1 & x_2 \end{array} \\ \begin{array}{c} x_3 \\ x_4 \end{array} & \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix} \end{array} \qquad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline 0 & 0 & 0 & 0 \end{array}$$

$$\downarrow$$

$$\begin{array}{cc} & \begin{array}{cc} x_3 & x_2 \end{array} \\ \begin{array}{c} x_1 \\ x_4 \end{array} & \begin{pmatrix} 1 & -2 \\ 2 & -3 \end{pmatrix} \end{array} \qquad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline -4 & 0 & -4 & -8 \end{array}$$

$$\downarrow$$

$$\begin{array}{cc} & \begin{array}{cc} x_3 & x_4 \end{array} \\ \begin{array}{c} x_1 \\ x_2 \end{array} & \begin{pmatrix} -\frac{1}{3} & \frac{2}{3} \\ \frac{2}{3} & -\frac{1}{3} \end{pmatrix} \end{array} \qquad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline -\frac{10}{3} & -\frac{1}{3} & -4 & -7 \end{array}$$

Example (Felgenhauer and Middeldorp)

$-1 \leqslant x_1 \leqslant 0 \qquad -4 \leqslant x_2 \leqslant 0 \qquad -5 \leqslant x_3 \leqslant -4 \qquad -7 \leqslant x_4 \leqslant 1$

$$\begin{array}{c|cc} & x_1 & x_2 \\ \hline x_3 & 1 & 2 \\ x_4 & 2 & 1 \end{array} \qquad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline 0 & 0 & 0 & 0 \end{array}$$

$$\downarrow$$

$$\begin{array}{c|cc} & x_3 & x_2 \\ \hline x_1 & 1 & -2 \\ x_4 & 2 & -3 \end{array} \qquad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline -4 & 0 & -4 & -8 \end{array}$$

$$\downarrow$$

$$\begin{array}{c|cc} & x_3 & x_4 \\ \hline x_1 & -\frac{1}{3} & \frac{2}{3} \\ x_2 & \frac{2}{3} & -\frac{1}{3} \end{array} \qquad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline -\frac{10}{3} & -\frac{1}{3} & -4 & -7 \end{array}$$

Example (Felgenhauer and Middeldorp)

$$-1 \leqslant x_1 \leqslant 0 \qquad -4 \leqslant x_2 \leqslant 0 \qquad -5 \leqslant x_3 \leqslant -4 \qquad -7 \leqslant x_4 \leqslant 1$$

$$\begin{array}{cc} & \begin{array}{cc} x_1 & x_2 \end{array} \\ \begin{array}{c} x_3 \\ x_4 \end{array} & \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix} \end{array} \qquad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline 0 & 0 & 0 & 0 \end{array}$$

$$\downarrow$$

$$\begin{array}{cc} & \begin{array}{cc} x_3 & x_2 \end{array} \\ \begin{array}{c} x_1 \\ x_4 \end{array} & \begin{pmatrix} 1 & -2 \\ 2 & -3 \end{pmatrix} \end{array} \qquad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline -4 & 0 & -4 & -8 \end{array}$$

$$\downarrow$$

$$\begin{array}{cc} & \begin{array}{cc} x_3 & x_4 \end{array} \\ \begin{array}{c} x_1 \\ x_2 \end{array} & \begin{pmatrix} -\frac{1}{3} & \frac{2}{3} \\ \frac{2}{3} & -\frac{1}{3} \end{pmatrix} \end{array} \qquad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline -\frac{10}{3} & -\frac{1}{3} & -4 & -7 \end{array}$$

$$\downarrow$$

$$\begin{array}{cc} & \begin{array}{cc} x_1 & x_4 \end{array} \\ \begin{array}{c} x_3 \\ x_2 \end{array} & \begin{pmatrix} -3 & 2 \\ -2 & 1 \end{pmatrix} \end{array} \qquad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline -1 & -5 & -11 & -7 \end{array}$$

Example (Felgenhauer and Middeldorp)

$$-1 \leqslant x_1 \leqslant 0 \qquad -4 \leqslant x_2 \leqslant 0 \qquad -5 \leqslant x_3 \leqslant -4 \qquad -7 \leqslant x_4 \leqslant 1$$

$$\begin{array}{c|cc} & x_1 & x_2 \\ \hline x_3 & 1 & 2 \\ x_4 & 2 & 1 \end{array} \qquad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline 0 & 0 & 0 & 0 \end{array}$$

$$\downarrow$$

$$\begin{array}{c|cc} & x_3 & x_2 \\ \hline x_1 & 1 & -2 \\ x_4 & 2 & -3 \end{array} \qquad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline -4 & 0 & -4 & -8 \end{array}$$

$$\downarrow$$

$$\begin{array}{c|cc} & x_3 & x_4 \\ \hline x_1 & -\frac{1}{3} & \frac{2}{3} \\ x_2 & \frac{2}{3} & -\frac{1}{3} \end{array} \qquad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline -\frac{10}{3} & -\frac{1}{3} & -4 & -7 \end{array}$$

$$\downarrow$$

$$\begin{array}{c|cc} & x_1 & x_4 \\ \hline x_3 & -3 & 2 \\ x_2 & -2 & 1 \end{array} \qquad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline -1 & -5 & -11 & -7 \end{array} \longrightarrow \begin{array}{c|cc} & x_1 & x_2 \\ \hline x_3 & 1 & 2 \\ x_4 & 2 & 1 \end{array} \qquad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline -1 & -4 & -9 & -6 \end{array}$$

Example (Felgenhauer and Middeldorp)

$$-1 \leqslant x_1 \leqslant 0 \qquad -4 \leqslant x_2 \leqslant 0 \qquad -5 \leqslant x_3 \leqslant -4 \qquad -7 \leqslant x_4 \leqslant 1$$

$$\begin{array}{cc} & \begin{array}{cc} x_1 & x_2 \end{array} \\ \begin{array}{c} x_3 \\ x_4 \end{array} & \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix} \end{array} \qquad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline 0 & 0 & 0 & 0 \end{array}$$

$$\downarrow$$

$$\begin{array}{cc} & \begin{array}{cc} x_3 & x_2 \end{array} \\ \begin{array}{c} x_1 \\ x_4 \end{array} & \begin{pmatrix} 1 & -2 \\ 2 & -3 \end{pmatrix} \end{array} \qquad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline -4 & 0 & -4 & -8 \end{array}$$

$$\downarrow$$

$$\begin{array}{cc} & \begin{array}{cc} x_3 & x_4 \end{array} \\ \begin{array}{c} x_1 \\ x_2 \end{array} & \begin{pmatrix} -\frac{1}{3} & \frac{2}{3} \\ \frac{2}{3} & -\frac{1}{3} \end{pmatrix} \end{array} \qquad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline -\frac{10}{3} & -\frac{1}{3} & -4 & -7 \end{array}$$

$$\downarrow$$

$$\begin{array}{cc} & \begin{array}{cc} x_1 & x_4 \end{array} \\ \begin{array}{c} x_3 \\ x_2 \end{array} & \begin{pmatrix} -3 & 2 \\ -2 & 1 \end{pmatrix} \end{array} \qquad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline -1 & -5 & -11 & -7 \end{array}$$

$$\longrightarrow$$

$$\begin{array}{cc} & \begin{array}{cc} x_1 & x_2 \end{array} \\ \begin{array}{c} x_3 \\ x_4 \end{array} & \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix} \end{array} \qquad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline -1 & -4 & -9 & -6 \end{array}$$

$$\uparrow$$

$$\begin{array}{cc} & \begin{array}{cc} x_3 & x_2 \end{array} \\ \begin{array}{c} x_1 \\ x_4 \end{array} & \begin{pmatrix} 1 & -2 \\ 2 & -3 \end{pmatrix} \end{array} \qquad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline 3 & -4 & -5 & 2 \end{array}$$

Example (Felgenhauer and Middeldorp)

$$-1 \leqslant x_1 \leqslant 0 \qquad -4 \leqslant x_2 \leqslant 0 \qquad -5 \leqslant x_3 \leqslant -4 \qquad -7 \leqslant x_4 \leqslant 1$$

$$\begin{array}{cc} & \begin{array}{cc} x_1 & x_2 \end{array} \\ \begin{array}{c} x_3 \\ x_4 \end{array} & \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix} \end{array} \qquad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline 0 & 0 & 0 & 0 \end{array}$$

$$\downarrow$$

$$\begin{array}{cc} & \begin{array}{cc} x_3 & x_2 \end{array} \\ \begin{array}{c} x_1 \\ x_4 \end{array} & \begin{pmatrix} 1 & -2 \\ 2 & -3 \end{pmatrix} \end{array} \qquad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline -4 & 0 & -4 & -8 \end{array}$$

$$\downarrow$$

$$\begin{array}{cc} & \begin{array}{cc} x_3 & x_4 \end{array} \\ \begin{array}{c} x_1 \\ x_2 \end{array} & \begin{pmatrix} -\frac{1}{3} & \frac{2}{3} \\ \frac{2}{3} & -\frac{1}{3} \end{pmatrix} \end{array} \qquad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline -\frac{10}{3} & -\frac{1}{3} & -4 & -7 \end{array}$$

$$\downarrow$$

$$\begin{array}{cc} & \begin{array}{cc} x_1 & x_4 \end{array} \\ \begin{array}{c} x_3 \\ x_2 \end{array} & \begin{pmatrix} -3 & 2 \\ -2 & 1 \end{pmatrix} \end{array} \qquad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline -1 & -5 & -11 & -7 \end{array}$$

$$\longrightarrow$$

$$\begin{array}{cc} & \begin{array}{cc} x_1 & x_2 \end{array} \\ \begin{array}{c} x_3 \\ x_4 \end{array} & \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix} \end{array} \qquad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline -1 & -4 & -9 & -6 \end{array}$$

$$\uparrow$$

$$\begin{array}{cc} & \begin{array}{cc} x_3 & x_2 \end{array} \\ \begin{array}{c} x_1 \\ x_4 \end{array} & \begin{pmatrix} 1 & -2 \\ 2 & -3 \end{pmatrix} \end{array} \qquad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline 3 & -4 & -5 & 2 \end{array}$$

$$\uparrow$$

$$\begin{array}{cc} & \begin{array}{cc} x_3 & x_4 \end{array} \\ \begin{array}{c} x_1 \\ x_2 \end{array} & \begin{pmatrix} -\frac{1}{3} & \frac{2}{3} \\ \frac{2}{3} & -\frac{1}{3} \end{pmatrix} \end{array} \qquad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline \frac{7}{3} & -\frac{11}{3} & -5 & 1 \end{array}$$

Example (Felgenhauer and Middeldorp)

$$-1 \leqslant x_1 \leqslant 0 \qquad -4 \leqslant x_2 \leqslant 0 \qquad -5 \leqslant x_3 \leqslant -4 \qquad -7 \leqslant x_4 \leqslant 1$$

$$\begin{array}{cc} & \begin{matrix} x_1 & x_2 \end{matrix} \\ \begin{matrix} x_3 \\ x_4 \end{matrix} & \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix} \end{array} \quad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline 0 & 0 & 0 & 0 \end{array}$$

$$\downarrow$$

$$\begin{array}{cc} & \begin{matrix} x_3 & x_2 \end{matrix} \\ \begin{matrix} x_1 \\ x_4 \end{matrix} & \begin{pmatrix} 1 & -2 \\ 2 & -3 \end{pmatrix} \end{array} \quad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline -4 & 0 & -4 & -8 \end{array}$$

$$\downarrow$$

$$\begin{array}{cc} & \begin{matrix} x_3 & x_4 \end{matrix} \\ \begin{matrix} x_1 \\ x_2 \end{matrix} & \begin{pmatrix} -\frac{1}{3} & \frac{2}{3} \\ \frac{2}{3} & -\frac{1}{3} \end{pmatrix} \end{array} \quad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline -\frac{10}{3} & -\frac{1}{3} & -4 & -7 \end{array}$$

$$\downarrow$$

$$\begin{array}{cc} & \begin{matrix} x_1 & x_4 \end{matrix} \\ \begin{matrix} x_3 \\ x_2 \end{matrix} & \begin{pmatrix} -3 & 2 \\ -2 & 1 \end{pmatrix} \end{array} \quad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline -1 & -5 & -11 & -7 \end{array}$$

$$\longrightarrow$$

$$\begin{array}{cc} & \begin{matrix} x_1 & x_2 \end{matrix} \\ \begin{matrix} x_3 \\ x_4 \end{matrix} & \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix} \end{array} \quad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline -1 & -4 & -9 & -6 \end{array}$$

$$\uparrow$$

$$\begin{array}{cc} & \begin{matrix} x_3 & x_2 \end{matrix} \\ \begin{matrix} x_1 \\ x_4 \end{matrix} & \begin{pmatrix} 1 & -2 \\ 2 & -3 \end{pmatrix} \end{array} \quad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline 3 & -4 & -5 & 2 \end{array}$$

$$\uparrow$$

$$\begin{array}{cc} & \begin{matrix} x_3 & x_4 \end{matrix} \\ \begin{matrix} x_1 \\ x_2 \end{matrix} & \begin{pmatrix} -\frac{1}{3} & \frac{2}{3} \\ \frac{2}{3} & -\frac{1}{3} \end{pmatrix} \end{array} \quad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline \frac{7}{3} & -\frac{11}{3} & -5 & 1 \end{array}$$

$$\uparrow$$

$$\begin{array}{cc} & \begin{matrix} x_1 & x_4 \end{matrix} \\ \begin{matrix} x_3 \\ x_2 \end{matrix} & \begin{pmatrix} -3 & 2 \\ -2 & 1 \end{pmatrix} \end{array} \quad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline 0 & 1 & 2 & 1 \end{array}$$

Example (Felgenhauer and Middeldorp)

$$-1 \leqslant x_1 \leqslant 0 \qquad -4 \leqslant x_2 \leqslant 0 \qquad -5 \leqslant x_3 \leqslant -4 \qquad -7 \leqslant x_4 \leqslant 1$$

$$\begin{array}{cc} & \begin{array}{cc} x_1 & x_2 \end{array} \\ \begin{array}{c} x_3 \\ x_4 \end{array} & \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix} \end{array} \quad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline 0 & 0 & 0 & 0 \end{array} \quad \longleftarrow \quad \begin{array}{cc} & \begin{array}{cc} x_1 & x_4 \end{array} \\ \begin{array}{c} x_3 \\ x_2 \end{array} & \begin{pmatrix} -3 & 2 \\ -2 & 1 \end{pmatrix} \end{array} \quad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline 0 & 1 & 2 & 1 \end{array}$$

$$\downarrow \qquad\qquad\qquad\qquad \uparrow$$

$$\begin{array}{cc} & \begin{array}{cc} x_3 & x_2 \end{array} \\ \begin{array}{c} x_1 \\ x_4 \end{array} & \begin{pmatrix} 1 & -2 \\ 2 & -3 \end{pmatrix} \end{array} \quad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline -4 & 0 & -4 & -8 \end{array} \qquad\qquad \begin{array}{cc} & \begin{array}{cc} x_3 & x_4 \end{array} \\ \begin{array}{c} x_1 \\ x_2 \end{array} & \begin{pmatrix} -\frac{1}{3} & \frac{2}{3} \\ \frac{2}{3} & -\frac{1}{3} \end{pmatrix} \end{array} \quad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline \frac{7}{3} & -\frac{11}{3} & -5 & 1 \end{array}$$

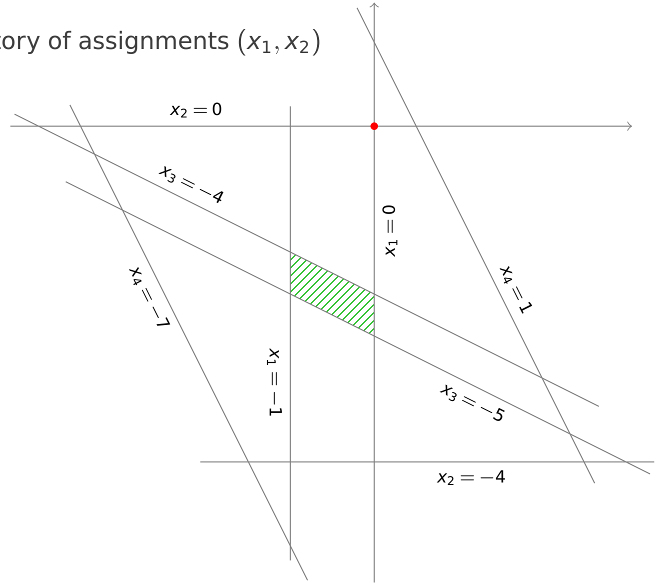
$$\downarrow \qquad\qquad\qquad\qquad \uparrow$$

$$\begin{array}{cc} & \begin{array}{cc} x_3 & x_4 \end{array} \\ \begin{array}{c} x_1 \\ x_2 \end{array} & \begin{pmatrix} -\frac{1}{3} & \frac{2}{3} \\ \frac{2}{3} & -\frac{1}{3} \end{pmatrix} \end{array} \quad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline -\frac{10}{3} & -\frac{1}{3} & -4 & -7 \end{array} \qquad\qquad \begin{array}{cc} & \begin{array}{cc} x_3 & x_2 \end{array} \\ \begin{array}{c} x_1 \\ x_4 \end{array} & \begin{pmatrix} 1 & -2 \\ 2 & -3 \end{pmatrix} \end{array} \quad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline 3 & -4 & -5 & 2 \end{array}$$

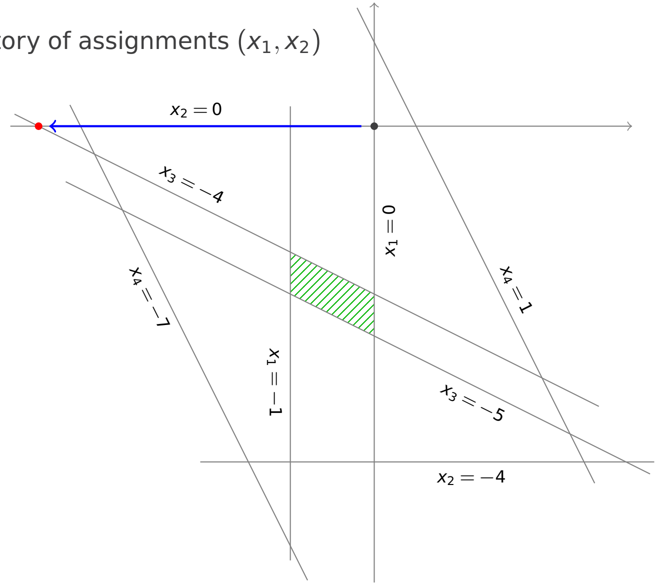
$$\downarrow \qquad\qquad\qquad\qquad \uparrow$$

$$\begin{array}{cc} & \begin{array}{cc} x_1 & x_4 \end{array} \\ \begin{array}{c} x_3 \\ x_2 \end{array} & \begin{pmatrix} -3 & 2 \\ -2 & 1 \end{pmatrix} \end{array} \quad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline -1 & -5 & -11 & -7 \end{array} \quad \longrightarrow \quad \begin{array}{cc} & \begin{array}{cc} x_1 & x_2 \end{array} \\ \begin{array}{c} x_3 \\ x_4 \end{array} & \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix} \end{array} \quad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline -1 & -4 & -9 & -6 \end{array}$$

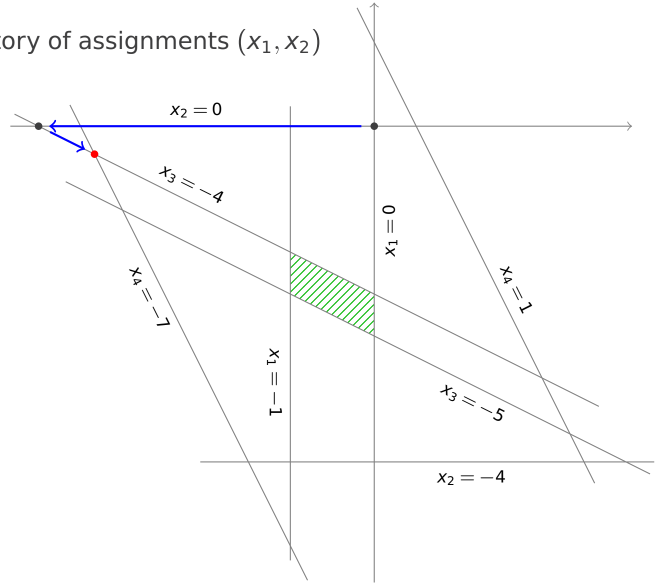
trajectory of assignments (x_1, x_2)



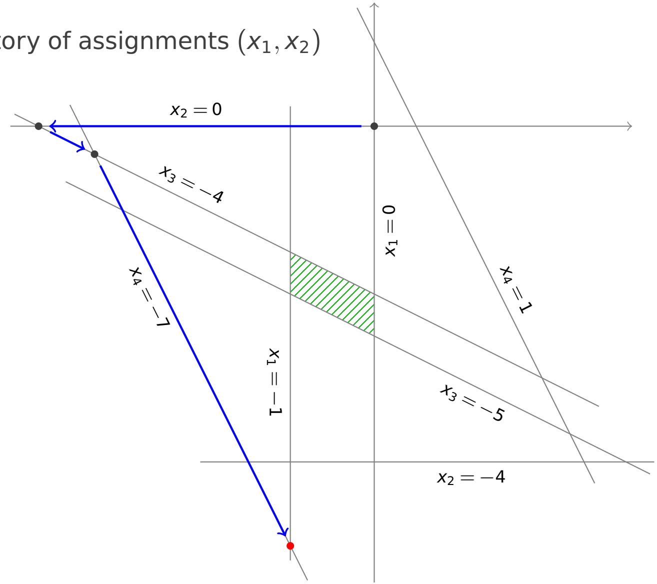
trajectory of assignments (x_1, x_2)



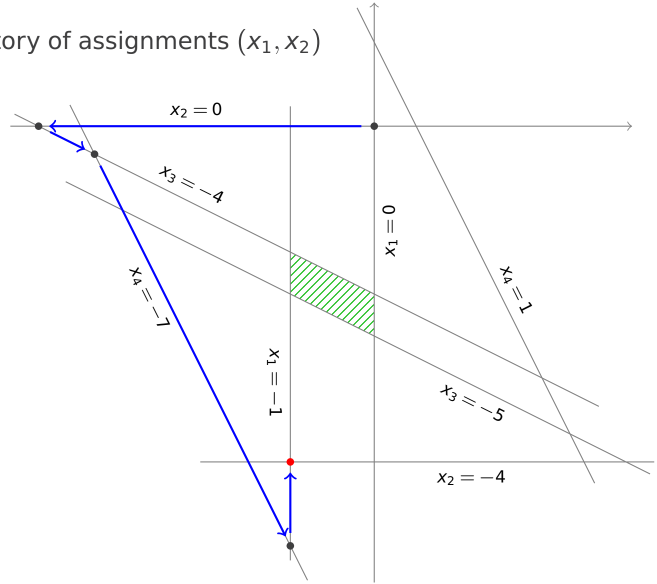
trajectory of assignments (x_1, x_2)



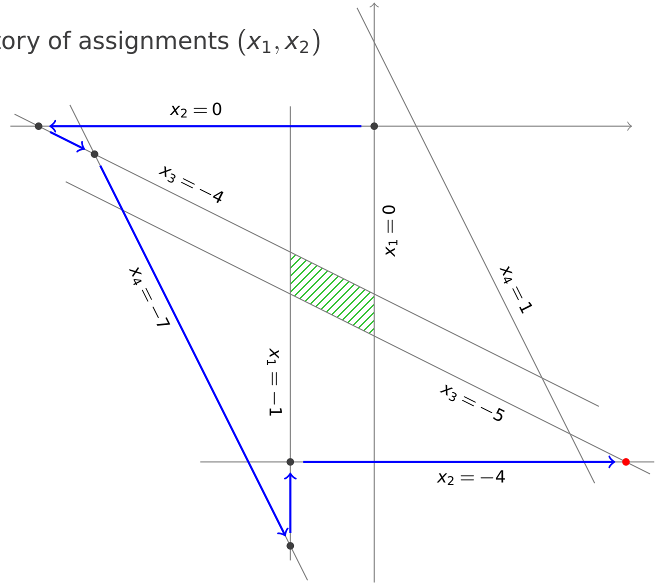
trajectory of assignments (x_1, x_2)



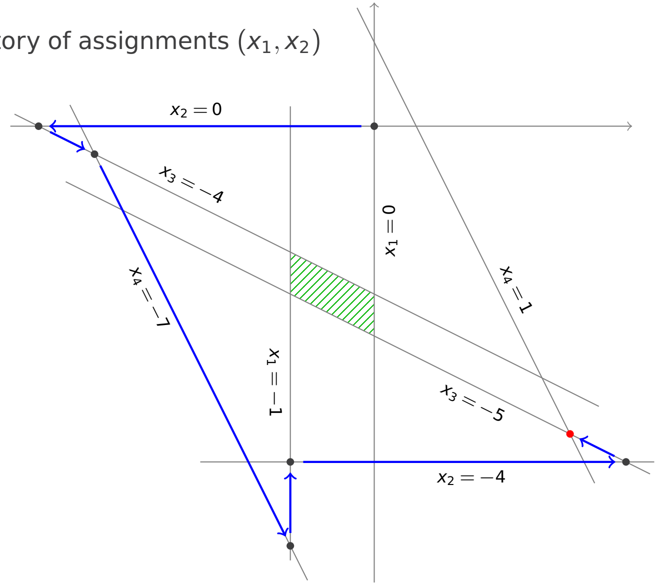
trajectory of assignments (x_1, x_2)



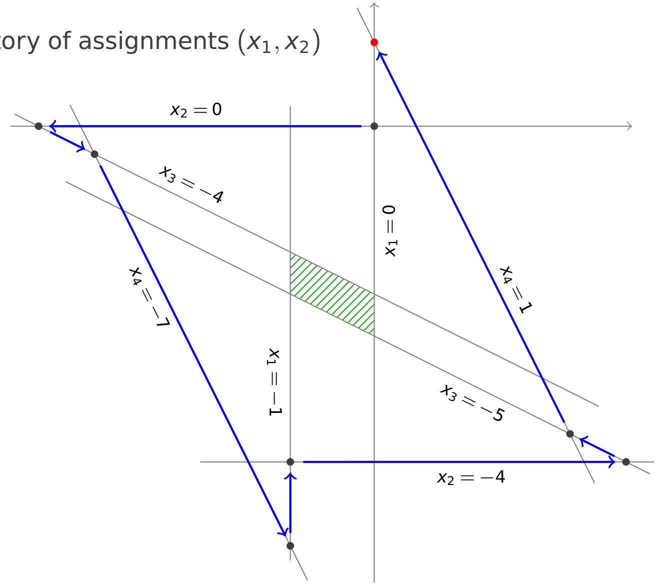
trajectory of assignments (x_1, x_2)



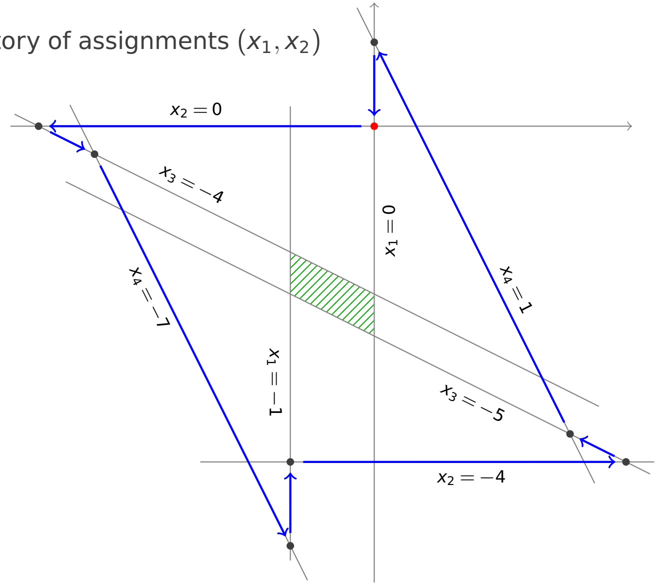
trajectory of assignments (x_1, x_2)



trajectory of assignments (x_1, x_2)



trajectory of assignments (x_1, x_2)



Example (Felgenhauer and Middeldorp)

$-1 \leqslant x_1 \leqslant 0 \qquad -4 \leqslant x_2 \leqslant 0 \qquad -5 \leqslant x_3 \leqslant -4 \qquad -7 \leqslant x_4 \leqslant 1$

$$\begin{array}{c|cc} & x_1 & x_2 \\ \hline x_3 & 1 & 2 \\ x_4 & 2 & 1 \end{array} \qquad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline 0 & 0 & 0 & 0 \end{array}$$

$$\downarrow$$

$$\begin{array}{c|cc} & x_3 & x_2 \\ \hline x_1 & 1 & -2 \\ x_4 & 2 & -3 \end{array} \qquad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline -4 & 0 & -4 & -8 \end{array}$$

violation of Bland's rule

$$\downarrow$$

$$\begin{array}{c|cc} & x_3 & x_4 \\ \hline x_1 & -\frac{1}{3} & \frac{2}{3} \\ x_2 & \frac{2}{3} & -\frac{1}{3} \end{array} \qquad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline -\frac{10}{3} & -\frac{1}{3} & -4 & -7 \end{array}$$

Example (Felgenhauer and Middeldorp)

$-1 \leqslant x_1 \leqslant 0 \qquad -4 \leqslant x_2 \leqslant 0 \qquad -5 \leqslant x_3 \leqslant -4 \qquad -7 \leqslant x_4 \leqslant 1$

$$\begin{array}{c} \\ x_3 \\ x_4 \end{array} \begin{array}{c} \begin{array}{cc} x_1 & x_2 \end{array} \\ \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix} \end{array} \qquad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline 0 & 0 & 0 & 0 \end{array}$$

$$\downarrow$$

$$\begin{array}{c} \\ x_1 \\ x_4 \end{array} \begin{array}{c} \begin{array}{cc} x_3 & x_2 \end{array} \\ \begin{pmatrix} 1 & -2 \\ 2 & -3 \end{pmatrix} \end{array} \qquad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline -4 & 0 & -4 & -8 \end{array}$$

Example (Felgenhauer and Middeldorp)

$$-1 \leqslant x_1 \leqslant 0 \qquad -4 \leqslant x_2 \leqslant 0 \qquad -5 \leqslant x_3 \leqslant -4 \qquad -7 \leqslant x_4 \leqslant 1$$

$$\begin{array}{cc} & \begin{array}{cc} x_1 & x_2 \end{array} \\ \begin{array}{c} x_3 \\ x_4 \end{array} & \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix} \end{array} \qquad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline 0 & 0 & 0 & 0 \end{array}$$

$$\downarrow$$

$$\begin{array}{cc} & \begin{array}{cc} x_3 & x_2 \end{array} \\ \begin{array}{c} x_1 \\ x_4 \end{array} & \begin{pmatrix} 1 & -2 \\ 2 & -3 \end{pmatrix} \end{array} \qquad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline -4 & 0 & -4 & -8 \end{array}$$

$$\downarrow$$

$$\begin{array}{cc} & \begin{array}{cc} x_3 & x_1 \end{array} \\ \begin{array}{c} x_2 \\ x_4 \end{array} & \begin{pmatrix} \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{3}{2} \end{pmatrix} \end{array} \qquad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline -1 & -\frac{3}{2} & -4 & -\frac{7}{2} \end{array}$$

Example (Felgenhauer and Middeldorp)

$$-1 \leqslant x_1 \leqslant 0 \qquad -4 \leqslant x_2 \leqslant 0 \qquad -5 \leqslant x_3 \leqslant -4 \qquad -7 \leqslant x_4 \leqslant 1$$

$$\begin{array}{c|cc} & x_1 & x_2 \\ \hline x_3 & 1 & 2 \\ x_4 & 2 & 1 \end{array} \qquad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline 0 & 0 & 0 & 0 \end{array}$$

$$\downarrow$$

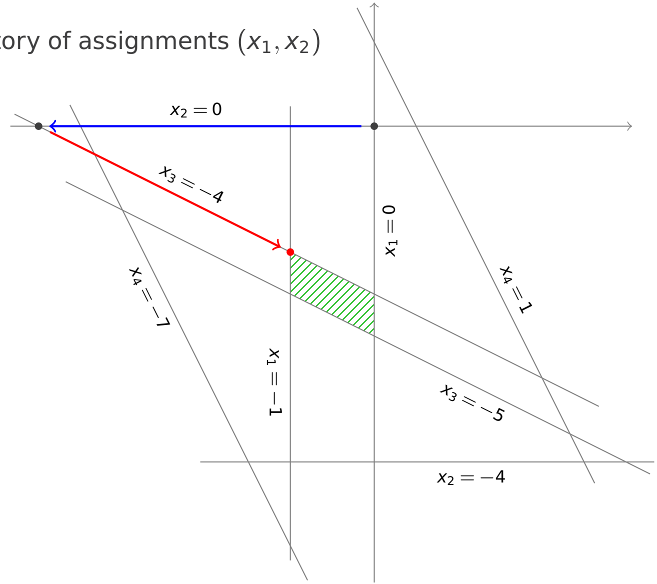
$$\begin{array}{c|cc} & x_3 & x_2 \\ \hline x_1 & 1 & -2 \\ x_4 & 2 & -3 \end{array} \qquad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline -4 & 0 & -4 & -8 \end{array}$$

$$\downarrow$$

$$\begin{array}{c|cc} & x_3 & x_1 \\ \hline x_2 & \frac{1}{2} & -\frac{1}{2} \\ x_4 & \frac{1}{2} & \frac{3}{2} \end{array} \qquad \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline -1 & -\frac{3}{2} & -4 & -\frac{7}{2} \end{array}$$

satisfying assignment

trajectory of assignments (x_1, x_2)



Outline

Unsatisfiable Cores for DPLL(T)

- recall: for DPLL(T) it is beneficial if theory solvers produce small unsatisfiable cores

Unsatisfiable Cores for DPLL(T)

- recall: for DPLL(T) it is beneficial if theory solvers produce small unsatisfiable cores
- simplex algorithm can easily identify unsatisfiable cores

Unsatisfiable Cores for DPLL(T)

- recall: for DPLL(T) it is beneficial if theory solvers produce small unsatisfiable cores
- simplex algorithm can easily identify unsatisfiable cores
- in detail: consider that unsatisfiability is detected via basic variable x_i

Unsatisfiable Cores for DPLL(T)

- recall: for DPLL(T) it is beneficial if theory solvers produce small unsatisfiable cores
- simplex algorithm can easily identify unsatisfiable cores
- in detail: consider that unsatisfiability is detected via basic variable x_i
  - w.l.o.g. we only consider the case $v(x_i) < l_i$ (the other case is symmetric)

Unsatisfiable Cores for DPLL(T)

- recall: for DPLL(T) it is beneficial if theory solvers produce small unsatisfiable cores
- simplex algorithm can easily identify unsatisfiable cores
- in detail: consider that unsatisfiability is detected via basic variable x_i
  - w.l.o.g. we only consider the case $v(x_i) < l_i$ (the other case is symmetric)
  - in this case tableau contains equation

$$x_i = \sum_{j \in N_{pos}} A_{ij} x_j + \sum_{k \in N_{neg}} A_{ik} x_k$$

such that $A_{ij} > 0 \wedge v(x_j) = u_j$ for all $j \in N_{pos}$ and $A_{ik} < 0 \wedge v(x_k) = l_k$ for all $k \in N_{neg}$

Unsatisfiable Cores for DPLL(T)

- recall: for DPLL(T) it is beneficial if theory solvers produce small unsatisfiable cores
- simplex algorithm can easily identify unsatisfiable cores
- in detail: consider that unsatisfiability is detected via basic variable x_i
  - w.l.o.g. we only consider the case $v(x_i) < l_i$ (the other case is symmetric)
  - in this case tableau contains equation

$$x_i = \sum_{j \in N_{pos}} A_{ij} x_j + \sum_{k \in N_{neg}} A_{ik} x_k$$

    such that $A_{ij} > 0 \wedge v(x_j) = u_j$ for all $j \in N_{pos}$ and $A_{ik} < 0 \wedge v(x_k) = l_k$ for all $k \in N_{neg}$
  - then the set of (original) constraints (corresponding to)

$$\begin{aligned} x_i &\geq l_i & & \\ x_j &\leq u_j & & \text{for all } j \in N_{pos} \\ x_k &\geq l_k & & \text{for all } k \in N_{neg} \end{aligned}$$

    is an unsatisfiable core

Unsatisfiable Cores for DPLL(T)

- recall: for DPLL(T) it is beneficial if theory solvers produce small unsatisfiable cores
- simplex algorithm can easily identify unsatisfiable cores
- in detail: consider that unsatisfiability is detected via basic variable x_i
  - w.l.o.g. we only consider the case $v(x_i) < l_i$ (the other case is symmetric)
  - in this case tableau contains equation

$$x_i = \sum_{j \in N_{pos}} A_{ij} x_j + \sum_{k \in N_{neg}} A_{ik} x_k$$

    such that $A_{ij} > 0 \wedge v(x_j) = u_j$ for all $j \in N_{pos}$ and $A_{ik} < 0 \wedge v(x_k) = l_k$ for all $k \in N_{neg}$
  - then the set of (original) constraints (corresponding to)

$$\begin{aligned} x_i &\geq l_i & & \\ x_j &\leq u_j & & \text{for all } j \in N_{pos} \\ x_k &\geq l_k & & \text{for all } k \in N_{neg} \end{aligned}$$

    is an unsatisfiable core
  - this core is minimal w.r.t. the subset-relation

Farkas' Lemma

- consider $\leq$-constraints $\underbrace{-x \leq -5}_{\ell_1 \leq r_1} \wedge \underbrace{2x + y \leq 12}_{\ell_2 \leq r_2} \wedge \underbrace{-y \leq -3}_{\ell_3 \leq r_3} \wedge \underbrace{x - 3y \leq 2}_{\ell_4 \leq r_4}$
- alternative way to prove unsatisfiability of constraints $\ell_1 \leq r_1, \ell_2 \leq r_2, \dots$

Farkas' Lemma

- consider $\leq$-constraints $\underbrace{-x \leq -5}_{\ell_1 \leq r_1} \wedge \underbrace{2x + y \leq 12}_{\ell_2 \leq r_2} \wedge \underbrace{-y \leq -3}_{\ell_3 \leq r_3} \wedge \underbrace{x - 3y \leq 2}_{\ell_4 \leq r_4}$
- alternative way to prove unsatisfiability of constraints $\ell_1 \leq r_1, \ell_2 \leq r_2, \ldots$
  - find Farkas' coefficients, i.e., non-negative coefficients $c_1, c_2, \ldots$ such that

$$\mathbb{Q} \ni \sum_i c_i \ell_i > \sum_i c_i r_i \in \mathbb{Q}$$

Farkas' Lemma

- consider $\leq$-constraints $\underbrace{-x \leq -5}_{\ell_1 \leq r_1} \wedge \underbrace{2x + y \leq 12}_{\ell_2 \leq r_2} \wedge \underbrace{-y \leq -3}_{\ell_3 \leq r_3} \wedge \underbrace{x - 3y \leq 2}_{\ell_4 \leq r_4}$
- alternative way to prove unsatisfiability of constraints $\ell_1 \leq r_1, \ell_2 \leq r_2, \ldots$
  - find Farkas' coefficients, i.e., non-negative coefficients $c_1, c_2, \ldots$ such that

$$\mathbb{Q} \ni \sum_i c_i \ell_i > \sum_i c_i r_i \in \mathbb{Q}$$

  - example: choose $c_1 = 2$, $c_2 = c_3 = 1$, $c_4 = 0$

$$2 \cdot (-x) + 1 \cdot (2x + y) + 1 \cdot (-y) + 0 \cdot (x - 3y) = 0 > -1 = 2 \cdot (-5) + 1 \cdot 12 + 1 \cdot (-3) + 0 \cdot 2$$

Farkas' Lemma

- consider $\leq$-constraints $\underbrace{-x \leq -5}_{\ell_1 \leq r_1} \land \underbrace{2x + y \leq 12}_{\ell_2 \leq r_2} \land \underbrace{-y \leq -3}_{\ell_3 \leq r_3} \land \underbrace{x - 3y \leq 2}_{\ell_4 \leq r_4}$
- alternative way to prove unsatisfiability of constraints $\ell_1 \leq r_1, \ell_2 \leq r_2, \ldots$
  - find Farkas' coefficients, i.e., non-negative coefficients $c_1, c_2, \ldots$ such that

$$\mathbb{Q} \ni \sum_i c_i \ell_i > \sum_i c_i r_i \in \mathbb{Q}$$

  - example: choose $c_1 = 2$, $c_2 = c_3 = 1$, $c_4 = 0$

$$2 \cdot (-x) + 1 \cdot (2x + y) + 1 \cdot (-y) + 0 \cdot (x - 3y) = 0 > -1 = 2 \cdot (-5) + 1 \cdot 12 + 1 \cdot (-3) + 0 \cdot 2$$

  - Farkas' Lemma: finite set of $\leq$-constraints is unsatisfiable iff Farkas' coefficients exists

Farkas' Lemma

- consider $\leq$-constraints $\underbrace{-x \leq -5}_{\ell_1 \leq r_1} \wedge \underbrace{2x + y \leq 12}_{\ell_2 \leq r_2} \wedge \underbrace{-y \leq -3}_{\ell_3 \leq r_3} \wedge \underbrace{x - 3y \leq 2}_{\ell_4 \leq r_4}$
- alternative way to prove unsatisfiability of constraints $\ell_1 \leq r_1, \ell_2 \leq r_2, \dots$
  - find Farkas' coefficients, i.e., non-negative coefficients $c_1, c_2, \dots$ such that

$$\mathbb{Q} \ni \sum_i c_i \ell_i > \sum_i c_i r_i \in \mathbb{Q}$$

  - example: choose $c_1 = 2$, $c_2 = c_3 = 1$, $c_4 = 0$

$$2 \cdot (-x) + 1 \cdot (2x + y) + 1 \cdot (-y) + 0 \cdot (x - 3y) = 0 > -1 = 2 \cdot (-5) + 1 \cdot 12 + 1 \cdot (-3) + 0 \cdot 2$$

  - Farkas' Lemma: finite set of $\leq$-constraints is unsatisfiable iff Farkas' coefficients exists
    - soundness: existence of Farkas' coefficients obviously shows unsatisfiability

Farkas' Lemma

- consider $\leq$-constraints $\underbrace{-x \leq -5}_{\ell_1 \leq r_1} \wedge \underbrace{2x + y \leq 12}_{\ell_2 \leq r_2} \wedge \underbrace{-y \leq -3}_{\ell_3 \leq r_3} \wedge \underbrace{x - 3y \leq 2}_{\ell_4 \leq r_4}$
- alternative way to prove unsatisfiability of constraints $\ell_1 \leq r_1, \ell_2 \leq r_2, \dots$
  - find Farkas' coefficients, i.e., non-negative coefficients $c_1, c_2, \dots$ such that

$$\mathbb{Q} \ni \sum_i c_i \ell_i > \sum_i c_i r_i \in \mathbb{Q}$$

  - example: choose $c_1 = 2$, $c_2 = c_3 = 1$, $c_4 = 0$

$$2 \cdot (-x) + 1 \cdot (2x + y) + 1 \cdot (-y) + 0 \cdot (x - 3y) = 0 > -1 = 2 \cdot (-5) + 1 \cdot 12 + 1 \cdot (-3) + 0 \cdot 2$$

  - Farkas' Lemma: finite set of $\leq$-constraints is unsatisfiable iff Farkas' coefficients exists
    - soundness: existence of Farkas' coefficients obviously shows unsatisfiability
    - completeness: if constraints are unsatisfiable, then simplex will detect this; whenever unsatisfiability is detected in simplex algorithm, one can extract Farkas' coefficients from the tableau equation (similar to the detection of unsatisfiable cores, but with finer analysis)

Example (Application of Linear Arithmetic: Termination Proving)

- consider program

```
factorial(n) {
  i = 1;
  r = 1;
  while (i <= n) {
    r = r * i;
    i = i + 1;  }
  return r;          }
```

Example (Application of Linear Arithmetic: Termination Proving)

- consider program

```
factorial(n) {
  i = 1;
  r = 1;
  while (i <= n) {
    r = r * i;
    i = i + 1;  }
  return r;           }
```

- φ describes one iteration of loop (primed variables store values after iteration)

$$\varphi := i \leq n \land i' = i + 1 \land r' = r \cdot i \land n' = n$$

Example (Application of Linear Arithmetic: Termination Proving)

- consider program

```
factorial(n) {
  i = 1;
  r = 1;
  while (i <= n) {
    r = r * i;
    i = i + 1;  }
  return r;           }
```

- φ describes one iteration of loop (primed variables store values after iteration)

$$\varphi := i \leq n \wedge i' = i + 1 \wedge r' = r \cdot i \wedge n' = n$$

- proving termination: find linear ranking function, i.e., linear expression $e(i, n, r)$, decrease factor d with $0 < d \in \mathbb{Q}$, and bound $f \in \mathbb{Q}$ such that
  - $\varphi \to e(i, n, r) \geq e(i', n', r') + d$ (expression decreases in every iteration by at least d)
  - $\varphi \to e(i, n, r) \geq f$ (expression is bounded from below by f)

Example (Termination Proof Continued)

- loop iteration $\varphi := i \leq n \wedge i' = i + 1 \wedge r' = r \cdot i \wedge n' = n$
- proving termination by validity of formulas

$$\varphi \to e(i,n,r) \geq e(i',n',r') + d \qquad\qquad \varphi \to e(i,n,r) \geq f$$

Example (Termination Proof Continued)

- loop iteration $\varphi := i \leq n \wedge i' = i + 1 \wedge r' = r \cdot i \wedge n' = n$
- proving termination by validity of formulas

$$\varphi \to e(i,n,r) \geq e(i',n',r') + d \qquad \varphi \to e(i,n,r) \geq f$$

- is equivalent to unsatisfiability of negated formulas

$$\varphi \wedge e(i,n,r) < e(i',n',r') + d \qquad \varphi \wedge e(i,n,r) < f$$

Example (Termination Proof Continued)

- loop iteration $\varphi := i \leq n \wedge i' = i + 1 \wedge r' = r \cdot i \wedge n' = n$
- proving termination by validity of formulas

$$\varphi \to e(i,n,r) \geq e(i',n',r') + d \qquad\qquad \varphi \to e(i,n,r) \geq f$$

- is equivalent to unsatisfiability of negated formulas

$$\varphi \wedge e(i,n,r) < e(i',n',r') + d \qquad\qquad \varphi \wedge e(i,n,r) < f$$

- choose linear ranking function $e(i,n,r) := n - i$ and $d := 1$ and $f = -1$, and drop all non-linear constraints to get two linear problems:
  - $i < n \wedge i' = i + 1 \wedge n' = n \wedge n - i < n' - i' + 1$ (violate decrease)
  - $i < n \wedge i' = i + 1 \wedge n' = n \wedge n - i < -1$ (violate boundedness)

Example (Termination Proof Continued)

- loop iteration $\varphi := i \leq n \wedge i' = i + 1 \wedge r' = r \cdot i \wedge n' = n$
- proving termination by validity of formulas

$$\varphi \to e(i,n,r) \geq e(i',n',r') + d \qquad\qquad \varphi \to e(i,n,r) \geq f$$

- is equivalent to unsatisfiability of negated formulas

$$\varphi \wedge e(i,n,r) < e(i',n',r') + d \qquad\qquad \varphi \wedge e(i,n,r) < f$$

- choose linear ranking function $e(i,n,r) := n - i$ and $d := 1$ and $f = -1$, and drop all non-linear constraints to get two linear problems:
  - $i < n \wedge i' = i + 1 \wedge n' = n \wedge n - i < n' - i' + 1$ (violate decrease)
  - $i < n \wedge i' = i + 1 \wedge n' = n \wedge n - i < -1$ (violate boundedness)

  both problems are unsatisfiable over $\mathbb{Q}$ (just run simplex), so termination is proved

Example (Termination Proof Continued)

- loop iteration $\varphi := i \leq n \wedge i' = i + 1 \wedge r' = r \cdot i \wedge n' = n$
- proving termination by validity of formulas

$$\varphi \to e(i,n,r) \geq e(i',n',r') + d \qquad\qquad \varphi \to e(i,n,r) \geq f$$

- is equivalent to unsatisfiability of negated formulas

$$\varphi \wedge e(i,n,r) < e(i',n',r') + d \qquad\qquad \varphi \wedge e(i,n,r) < f$$

- choose linear ranking function $e(i,n,r) := n - i$ and $d := 1$ and $f = -1$, and drop all non-linear constraints to get two linear problems:
  - $i < n \wedge i' = i + 1 \wedge n' = n \wedge n - i < n' - i' + 1$ (violate decrease)
  - $i < n \wedge i' = i + 1 \wedge n' = n \wedge n - i < -1$ (violate boundedness)

  both problems are unsatisfiable over $\mathbb{Q}$ (just run simplex), so termination is proved
- problem: how to find linear expression $e(i,n,r)$ and constants d and f?

Example (Termination Proof Continued)

- loop iteration $\varphi := i \leq n \land i' = i + 1 \land r' = r \cdot i \land n' = n$
- proving termination by validity of formulas

$$\varphi \to e(i,n,r) \geq e(i',n',r') + d \qquad\qquad \varphi \to e(i,n,r) \geq f$$

- is equivalent to unsatisfiability of negated formulas

$$\varphi \land e(i,n,r) < e(i',n',r') + d \qquad\qquad \varphi \land e(i,n,r) < f$$

- choose linear ranking function $e(i,n,r) := n - i$ and $d := 1$ and $f = -1$, and drop all non-linear constraints to get two linear problems:
  - $i < n \land i' = i + 1 \land n' = n \land n - i < n' - i' + 1$ (violate decrease)
  - $i < n \land i' = i + 1 \land n' = n \land n - i < -1$ (violate boundedness)

  both problems are unsatisfiable over $\mathbb{Q}$ (just run simplex), so termination is proved
- problem: how to find linear expression $e(i,n,r)$ and constants d and f?
- solution: combined search for $e(i,n,r)$, d and f and Farkas' coefficients

Towards Algorithm to Synthesize Linear Ranking Function

- assume loop is given as transition formula in the form of linear inequalities $A\vec{x} + A'\vec{x'} \leq \vec{b}$ between primed and unprimed variables

Towards Algorithm to Synthesize Linear Ranking Function

- assume loop is given as transition formula in the form of linear inequalities $A\vec{x} + A'\vec{x'} \leq \vec{b}$ between primed and unprimed variables
- example

$$\underbrace{\begin{pmatrix} 1 & -1 \\ 0 & 1 \\ 0 & -1 \\ 1 & 0 \\ -1 & 0 \end{pmatrix}}_{A} \cdot \begin{pmatrix} i \\ n \end{pmatrix} + \underbrace{\begin{pmatrix} 0 & 0 \\ 0 & -1 \\ 0 & 1 \\ -1 & 0 \\ 1 & 0 \end{pmatrix}}_{A'} \cdot \begin{pmatrix} i' \\ n' \end{pmatrix} \leq \underbrace{\begin{pmatrix} 0 \\ 0 \\ 0 \\ -1 \\ 1 \end{pmatrix}}_{\vec{b}}$$

encodes transition formula $i \leq n \wedge n' = n \wedge i' = i + 1$
(formula φ after removal of non-linear part)

Algorithm to Synthesize Linear Ranking Function [Podelski, Rybalschenko]

- assume loop is given as transition formula in the form of linear inequalities $A\vec{x} + A'\vec{x'} \leq \vec{b}$ between primed and unprimed variables
- idea: find parameters of ranking function and Farkas' coefficients in one go

Algorithm to Synthesize Linear Ranking Function [Podelski, Rybalschenko]

- assume loop is given as transition formula in the form of linear inequalities $A\vec{x} + A'\vec{x'} \leq \vec{b}$ between primed and unprimed variables
- idea: find parameters of ranking function and Farkas' coefficients in one go
- algorithm: encode the following constraints for row vectors $\vec{c_1}, \vec{c_2}$ of variables
  - $\vec{c_1} \geq \vec{0}$, $\vec{c_2} \geq \vec{0}$
  - $\vec{c_1}A' = 0$
  - $\vec{c_1}A = \vec{c_2}A$
  - $\vec{c_2}A = -\vec{c_2}A'$
  - $\vec{c_2}\vec{b} < 0$

Algorithm to Synthesize Linear Ranking Function [Podelski, Rybalschenko]

- assume loop is given as transition formula in the form of linear inequalities $A\vec{x} + A'\vec{x'} \leq \vec{b}$ between primed and unprimed variables
- idea: find parameters of ranking function and Farkas' coefficients in one go
- algorithm: encode the following constraints for row vectors $\vec{c_1}, \vec{c_2}$ of variables
  - $\vec{c_1} \geq \vec{0}$, $\vec{c_2} \geq \vec{0}$
  - $\vec{c_1}A' = 0$
  - $\vec{c_1}A = \vec{c_2}A$
  - $\vec{c_2}A = -\vec{c_2}A'$
  - $\vec{c_2}\vec{b} < 0$

  and return "linear ranking function exists" iff constraints are satisfiable

Algorithm to Synthesize Linear Ranking Function [Podelski, Rybalschenko]

- assume loop is given as transition formula in the form of linear inequalities $A\vec{x} + A'\vec{x'} \leq \vec{b}$ between primed and unprimed variables
- idea: find parameters of ranking function and Farkas' coefficients in one go
- algorithm: encode the following constraints for row vectors $\vec{c_1}, \vec{c_2}$ of variables
  - $\vec{c_1} \geq \vec{0}$, $\vec{c_2} \geq \vec{0}$
  - $\vec{c_1}A' = 0$
  - $\vec{c_1}A = \vec{c_2}A$
  - $\vec{c_2}A = -\vec{c_2}A'$
  - $\vec{c_2}\vec{b} < 0$

  and return "linear ranking function exists" iff constraints are satisfiable
- completeness is based on Farkas' lemma (assumes satisfiable transition formula)

Algorithm to Synthesize Linear Ranking Function [Podelski, Rybalschenko]

- assume loop is given as transition formula in the form of linear inequalities $A\vec{x} + A'\vec{x'} \leq \vec{b}$ between primed and unprimed variables
- idea: find parameters of ranking function and Farkas' coefficients in one go
- algorithm: encode the following constraints for row vectors $\vec{c_1}, \vec{c_2}$ of variables
  - $\vec{c_1} \geq \vec{0}$, $\vec{c_2} \geq \vec{0}$
  - $\vec{c_1}A' = 0$
  - $\vec{c_1}A = \vec{c_2}A$
  - $\vec{c_2}A = -\vec{c_2}A'$
  - $\vec{c_2}\vec{b} < 0$

  and return "linear ranking function exists" iff constraints are satisfiable
- completeness is based on Farkas' lemma (assumes satisfiable transition formula)
- soundness: extract parameters of ranking function from concrete solution $\vec{c_1}, \vec{c_2}$

$$e(\vec{x}) = \vec{c_2}A'\vec{x} \qquad d = -\vec{c_2}\vec{b} \qquad f = -\vec{c_1}\vec{b}$$

Example Application (Continue in Termination Proof)

- $\begin{pmatrix} c_1 & c_2 & c_3 & c_4 & c_5 \end{pmatrix} \geq \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad \begin{pmatrix} c_6 & c_7 & c_8 & c_9 & c_{10} \end{pmatrix} \geq \begin{pmatrix} 0 & \dots & 0 \end{pmatrix}$
- $\begin{pmatrix} -c_4 + c_5 & -c_2 + c_3 \end{pmatrix} = \begin{pmatrix} 0 & 0 \end{pmatrix}$
- $\begin{pmatrix} c_1 + c_4 - c_5 & -c_1 + c_2 - c_3 \end{pmatrix} = \begin{pmatrix} c_6 + c_9 - c_{10} & -c_6 + c_7 - c_8 \end{pmatrix}$
- $\begin{pmatrix} c_6 + c_9 - c_{10} & -c_6 + c_7 - c_8 \end{pmatrix} = -\begin{pmatrix} -c_9 + c_{10} & -c_7 + c_8 \end{pmatrix}$
- $c_{10} - c_9 < 0$

Example Application (Continue in Termination Proof)

- $\begin{pmatrix} c_1 & c_2 & c_3 & c_4 & c_5 \end{pmatrix} \geq \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \end{pmatrix}$, $\begin{pmatrix} c_6 & c_7 & c_8 & c_9 & c_{10} \end{pmatrix} \geq \begin{pmatrix} 0 & \dots & 0 \end{pmatrix}$
- $\begin{pmatrix} -c_4 + c_5 & -c_2 + c_3 \end{pmatrix} = \begin{pmatrix} 0 & 0 \end{pmatrix}$
- $\begin{pmatrix} c_1 + c_4 - c_5 & -c_1 + c_2 - c_3 \end{pmatrix} = \begin{pmatrix} c_6 + c_9 - c_{10} & -c_6 + c_7 - c_8 \end{pmatrix}$
- $\begin{pmatrix} c_6 + c_9 - c_{10} & -c_6 + c_7 - c_8 \end{pmatrix} = -\begin{pmatrix} -c_9 + c_{10} & -c_7 + c_8 \end{pmatrix}$
- $c_{10} - c_9 < 0$
- find solution $c_1 = c_8 = c_9 = 1$, $c_i = 0$ for $i \notin \{1, 8, 9\}$

Example Application (Continue in Termination Proof)

- $\begin{pmatrix} c_1 & c_2 & c_3 & c_4 & c_5 \end{pmatrix} \geq \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \end{pmatrix}$, $\begin{pmatrix} c_6 & c_7 & c_8 & c_9 & c_{10} \end{pmatrix} \geq \begin{pmatrix} 0 & \dots & 0 \end{pmatrix}$
- $\begin{pmatrix} -c_4 + c_5 & -c_2 + c_3 \end{pmatrix} = \begin{pmatrix} 0 & 0 \end{pmatrix}$
- $\begin{pmatrix} c_1 + c_4 - c_5 & -c_1 + c_2 - c_3 \end{pmatrix} = \begin{pmatrix} c_6 + c_9 - c_{10} & -c_6 + c_7 - c_8 \end{pmatrix}$
- $\begin{pmatrix} c_6 + c_9 - c_{10} & -c_6 + c_7 - c_8 \end{pmatrix} = -\begin{pmatrix} -c_9 + c_{10} & -c_7 + c_8 \end{pmatrix}$
- $c_{10} - c_9 < 0$
- find solution $c_1 = c_8 = c_9 = 1$, $c_i = 0$ for $i \notin \{1, 8, 9\}$
- extract ranking function parameters
  - $e(i, n) = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 \end{pmatrix} A' \begin{pmatrix} i \\ n \end{pmatrix} = n - i$
  - $d = -\begin{pmatrix} 0 & 0 & 1 & 1 & 0 \end{pmatrix} \vec{b} = 1$
  - $f = -\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \end{pmatrix} \vec{b} = 0$

Soundness Proof: Details

- loop transition formula: $A\vec{x} + A'\vec{x'} \leq \vec{b}$
- constraints: $\vec{c_1} \geq \vec{0}$, $\vec{c_2} \geq \vec{0}$, $\vec{c_1}A' = 0$, $\vec{c_1}A = \vec{c_2}A$, $\vec{c_2}A = -\vec{c_2}A'$, and $\vec{c_2}\vec{b} < 0$
- ranking function parameters: $e(\vec{x}) = \vec{c_2}A'\vec{x}$, $d = -\vec{c_2}\vec{b}$, and $f = -\vec{c_1}\vec{b}$

Soundness Proof: Details

- loop transition formula: $A\vec{x} + A'\vec{x'} \leq \vec{b}$
- constraints: $\vec{c_1} \geq \vec{0}$, $\vec{c_2} \geq \vec{0}$, $\vec{c_1}A' = 0$, $\vec{c_1}A = \vec{c_2}A$, $\vec{c_2}A = -\vec{c_2}A'$, and $\vec{c_2}\vec{b} < 0$
- ranking function parameters: $e(\vec{x}) = \vec{c_2}A'\vec{x}$, $d = -\vec{c_2}\vec{b}$, and $f = -\vec{c_1}\vec{b}$
- choice of d: $d = -\vec{c_2}\vec{b} > 0$

Soundness Proof: Details

- loop transition formula: $A\vec{x} + A'\vec{x'} \leq \vec{b}$
- constraints: $\vec{c_1} \geq \vec{0}$, $\vec{c_2} \geq \vec{0}$, $\vec{c_1}A' = 0$, $\vec{c_1}A = \vec{c_2}A$, $\vec{c_2}A = -\vec{c_2}A'$, and $\vec{c_2}\vec{b} < 0$
- ranking function parameters: $e(\vec{x}) = \vec{c_2}A'\vec{x}$, $d = -\vec{c_2}\vec{b}$, and $f = -\vec{c_1}\vec{b}$
- choice of d: $d = -\vec{c_2}\vec{b} > 0$
- boundedness

Soundness Proof: Details

- loop transition formula: $A\vec{x} + A'\vec{x'} \leq \vec{b}$
- constraints: $\vec{c_1} \geq \vec{0}$, $\vec{c_2} \geq \vec{0}$, $\vec{c_1}A' = 0$, $\vec{c_1}A = \vec{c_2}A$, $\vec{c_2}A = -\vec{c_2}A'$, and $\vec{c_2}\vec{b} < 0$
- ranking function parameters: $e(\vec{x}) = \vec{c_2}A'\vec{x}$, $d = -\vec{c_2}\vec{b}$, and $f = -\vec{c_1}\vec{b}$
- choice of d: $d = -\vec{c_2}\vec{b} > 0$
- boundedness
  - assume $\vec{x}$ and $\vec{x'}$ satisfy loop transition formula

Soundness Proof: Details

- loop transition formula: $A\vec{x} + A'\vec{x'} \leq \vec{b}$
- constraints: $\vec{c_1} \geq \vec{0}$, $\vec{c_2} \geq \vec{0}$, $\vec{c_1}A' = 0$, $\vec{c_1}A = \vec{c_2}A$, $\vec{c_2}A = -\vec{c_2}A'$, and $\vec{c_2}\vec{b} < 0$
- ranking function parameters: $e(\vec{x}) = \vec{c_2}A'\vec{x}$, $d = -\vec{c_2}\vec{b}$, and $f = -\vec{c_1}\vec{b}$
- choice of d: $d = -\vec{c_2}\vec{b} > 0$
- boundedness
  - assume $\vec{x}$ and $\vec{x'}$ satisfy loop transition formula
  - hence $\vec{c_1}(A\vec{x} + A'\vec{x'}) \leq \vec{c_1}\vec{b}$

Soundness Proof: Details

- loop transition formula: $A\vec{x} + A'\vec{x'} \leq \vec{b}$
- constraints: $\vec{c_1} \geq \vec{0}$, $\vec{c_2} \geq \vec{0}$, $\vec{c_1}A' = 0$, $\vec{c_1}A = \vec{c_2}A$, $\vec{c_2}A = -\vec{c_2}A'$, and $\vec{c_2}\vec{b} < 0$
- ranking function parameters: $e(\vec{x}) = \vec{c_2}A'\vec{x}$, $d = -\vec{c_2}\vec{b}$, and $f = -\vec{c_1}\vec{b}$
- choice of d: $d = -\vec{c_2}\vec{b} > 0$
- boundedness
  - assume $\vec{x}$ and $\vec{x'}$ satisfy loop transition formula
  - hence $\vec{c_1}(A\vec{x} + A'\vec{x'}) \leq \vec{c_1}\vec{b}$
  - hence $\vec{c_1}A\vec{x} + \vec{c_1}A'\vec{x'} \leq \vec{c_1}\vec{b}$

Soundness Proof: Details

- loop transition formula: $A\vec{x} + A'\vec{x'} \leq \vec{b}$
- constraints: $\vec{c_1} \geq \vec{0}$, $\vec{c_2} \geq \vec{0}$, $\vec{c_1}A' = 0$, $\vec{c_1}A = \vec{c_2}A$, $\vec{c_2}A = -\vec{c_2}A'$, and $\vec{c_2}\vec{b} < 0$
- ranking function parameters: $e(\vec{x}) = \vec{c_2}A'\vec{x}$, $d = -\vec{c_2}\vec{b}$, and $f = -\vec{c_1}\vec{b}$
- choice of d: $d = -\vec{c_2}\vec{b} > 0$
- boundedness
  - assume $\vec{x}$ and $\vec{x'}$ satisfy loop transition formula
  - hence $\vec{c_1}(A\vec{x} + A'\vec{x'}) \leq \vec{c_1}\vec{b}$
  - hence $\vec{c_1}A\vec{x} + \vec{c_1}A'\vec{x'} \leq \vec{c_1}\vec{b}$
  - hence $\vec{c_1}A\vec{x} \leq \vec{c_1}\vec{b}$

Soundness Proof: Details

- loop transition formula: $A\vec{x} + A'\vec{x'} \leq \vec{b}$
- constraints: $\vec{c_1} \geq \vec{0}$, $\vec{c_2} \geq \vec{0}$, $\vec{c_1}A' = 0$, $\vec{c_1}A = \vec{c_2}A$, $\vec{c_2}A = -\vec{c_2}A'$, and $\vec{c_2}\vec{b} < 0$
- ranking function parameters: $e(\vec{x}) = \vec{c_2}A'\vec{x}$, $d = -\vec{c_2}\vec{b}$, and $f = -\vec{c_1}\vec{b}$
- choice of d: $d = -\vec{c_2}\vec{b} > 0$
- boundedness
  - assume $\vec{x}$ and $\vec{x'}$ satisfy loop transition formula
  - hence $\vec{c_1}(A\vec{x} + A'\vec{x'}) \leq \vec{c_1}\vec{b}$
  - hence $\vec{c_1}A\vec{x} + \vec{c_1}A'\vec{x'} \leq \vec{c_1}\vec{b}$
  - hence $\vec{c_1}A\vec{x} \leq \vec{c_1}\vec{b}$
  - hence $\vec{c_2}A\vec{x} \leq \vec{c_1}\vec{b}$

Soundness Proof: Details

- loop transition formula: $A\vec{x} + A'\vec{x'} \leq \vec{b}$
- constraints: $\vec{c_1} \geq \vec{0}$, $\vec{c_2} \geq \vec{0}$, $\vec{c_1}A' = 0$, $\vec{c_1}A = \vec{c_2}A$, $\vec{c_2}A = -\vec{c_2}A'$, and $\vec{c_2}\vec{b} < 0$
- ranking function parameters: $e(\vec{x}) = \vec{c_2}A'\vec{x}$, $d = -\vec{c_2}\vec{b}$, and $f = -\vec{c_1}\vec{b}$
- choice of d: $d = -\vec{c_2}\vec{b} > 0$
- boundedness
  - assume $\vec{x}$ and $\vec{x'}$ satisfy loop transition formula
  - hence $\vec{c_1}(A\vec{x} + A'\vec{x'}) \leq \vec{c_1}\vec{b}$
  - hence $\vec{c_1}A\vec{x} + \vec{c_1}A'\vec{x'} \leq \vec{c_1}\vec{b}$
  - hence $\vec{c_1}A\vec{x} \leq \vec{c_1}\vec{b}$
  - hence $\vec{c_2}A\vec{x} \leq \vec{c_1}\vec{b}$
  - hence $-\vec{c_2}A'\vec{x} \leq \vec{c_1}\vec{b}$

Soundness Proof: Details

- loop transition formula: $A\vec{x} + A'\vec{x'} \leq \vec{b}$
- constraints: $\vec{c_1} \geq \vec{0}$, $\vec{c_2} \geq \vec{0}$, $\vec{c_1}A' = 0$, $\vec{c_1}A = \vec{c_2}A$, $\vec{c_2}A = -\vec{c_2}A'$, and $\vec{c_2}\vec{b} < 0$
- ranking function parameters: $e(\vec{x}) = \vec{c_2}A'\vec{x}$, $d = -\vec{c_2}\vec{b}$, and $f = -\vec{c_1}\vec{b}$
- choice of d: $d = -\vec{c_2}\vec{b} > 0$
- boundedness
  - assume $\vec{x}$ and $\vec{x'}$ satisfy loop transition formula
  - hence $\vec{c_1}(A\vec{x} + A'\vec{x'}) \leq \vec{c_1}\vec{b}$
  - hence $\vec{c_1}A\vec{x} + \vec{c_1}A'\vec{x'} \leq \vec{c_1}\vec{b}$
  - hence $\vec{c_1}A\vec{x} \leq \vec{c_1}\vec{b}$
  - hence $\vec{c_2}A\vec{x} \leq \vec{c_1}\vec{b}$
  - hence $-\vec{c_2}A'\vec{x} \leq \vec{c_1}\vec{b}$
  - hence $-e(\vec{x}) \leq -f$

Soundness Proof: Details

- loop transition formula: $A\vec{x} + A'\vec{x'} \leq \vec{b}$
- constraints: $\vec{c_1} \geq \vec{0}$, $\vec{c_2} \geq \vec{0}$, $\vec{c_1}A' = 0$, $\vec{c_1}A = \vec{c_2}A$, $\vec{c_2}A = -\vec{c_2}A'$, and $\vec{c_2}\vec{b} < 0$
- ranking function parameters: $e(\vec{x}) = \vec{c_2}A'\vec{x}$, $d = -\vec{c_2}\vec{b}$, and $f = -\vec{c_1}\vec{b}$
- choice of d: $d = -\vec{c_2}\vec{b} > 0$
- boundedness
  - assume $\vec{x}$ and $\vec{x'}$ satisfy loop transition formula
  - hence $\vec{c_1}(A\vec{x} + A'\vec{x'}) \leq \vec{c_1}\vec{b}$
  - hence $\vec{c_1}A\vec{x} + \vec{c_1}A'\vec{x'} \leq \vec{c_1}\vec{b}$
  - hence $\vec{c_1}A\vec{x} \leq \vec{c_1}\vec{b}$
  - hence $\vec{c_2}A\vec{x} \leq \vec{c_1}\vec{b}$
  - hence $-\vec{c_2}A'\vec{x} \leq \vec{c_1}\vec{b}$
  - hence $-e(\vec{x}) \leq -f$
  - hence $e(\vec{x}) \geq f$

Soundness Proof: Details

- loop transition formula: $A\vec{x} + A'\vec{x'} \leq \vec{b}$
- constraints: $\vec{c_1} \geq \vec{0}$, $\vec{c_2} \geq \vec{0}$, $\vec{c_1}A' = 0$, $\vec{c_1}A = \vec{c_2}A$, $\vec{c_2}A = -\vec{c_2}A'$, and $\vec{c_2}\vec{b} < 0$
- ranking function parameters: $e(\vec{x}) = \vec{c_2}A'\vec{x}$, $d = -\vec{c_2}\vec{b}$, and $f = -\vec{c_1}\vec{b}$
- choice of d: $d = -\vec{c_2}\vec{b} > 0$
- boundedness
  - assume $\vec{x}$ and $\vec{x'}$ satisfy loop transition formula
  - hence $\vec{c_1}(A\vec{x} + A'\vec{x'}) \leq \vec{c_1}\vec{b}$
  - hence $\vec{c_1}A\vec{x} + \vec{c_1}A'\vec{x'} \leq \vec{c_1}\vec{b}$
  - hence $\vec{c_1}A\vec{x} \leq \vec{c_1}\vec{b}$
  - hence $\vec{c_2}A\vec{x} \leq \vec{c_1}\vec{b}$
  - hence $-\vec{c_2}A'\vec{x} \leq \vec{c_1}\vec{b}$
  - hence $-e(\vec{x}) \leq -f$
  - hence $e(\vec{x}) \geq f$
- decrease $e(\vec{x}) \geq e(\vec{x'}) + d$: similar to boundedness

Outline

Why use simplex algorithm for LRA?

- situation

Why use simplex algorithm for LRA?

- situation
  - solving LRA problems is in P (interior point method, . . .)

Why use simplex algorithm for LRA?

- situation
  - solving LRA problems is in P (interior point method, . . .)
  - simplex algorithm has exponential worst-case time complexity

Why use simplex algorithm for LRA?

- situation
  - solving LRA problems is in P (interior point method, . . .)
  - simplex algorithm has exponential worst-case time complexity
- simplex algorithm is still popular

Why use simplex algorithm for LRA?

- situation
  - solving LRA problems is in P (interior point method, . . .)
  - simplex algorithm has exponential worst-case time complexity
- simplex algorithm is still popular
  - exponential behaviour rarely triggered, only on artificial examples

Why use simplex algorithm for LRA?

- situation
  - solving LRA problems is in P (interior point method, . . .)
  - simplex algorithm has exponential worst-case time complexity
- simplex algorithm is still popular
  - exponential behaviour rarely triggered, only on artificial examples
  - simplex can be used incrementally

Why use simplex algorithm for LRA?

- situation
  - solving LRA problems is in P (interior point method, ...)
  - simplex algorithm has exponential worst-case time complexity
- simplex algorithm is still popular
  - exponential behaviour rarely triggered, only on artificial examples
  - simplex can be used incrementally
- incrementality

Why use simplex algorithm for LRA?

- situation
  - solving LRA problems is in P (interior point method, . . .)
  - simplex algorithm has exponential worst-case time complexity
- simplex algorithm is still popular
  - exponential behaviour rarely triggered, only on artificial examples
  - simplex can be used incrementally
- incrementality
  - simplex can be used as theory solver for DPLL(T), where often constraints are activated and deactivated

Why use simplex algorithm for LRA?

- situation
  - solving LRA problems is in P (interior point method, . . .)
  - simplex algorithm has exponential worst-case time complexity
- simplex algorithm is still popular
  - exponential behaviour rarely triggered, only on artificial examples
  - simplex can be used incrementally
- incrementality
  - simplex can be used as theory solver for DPLL(T),
    where often constraints are activated and deactivated
  - simplex can be used as backend for LIA solver (next lecture),
    where new constraints are added on-the-fly

Why use simplex algorithm for LRA?

- situation
  - solving LRA problems is in P (interior point method, . . .)
  - simplex algorithm has exponential worst-case time complexity
- simplex algorithm is still popular
  - exponential behaviour rarely triggered, only on artificial examples
  - simplex can be used incrementally
- incrementality
  - simplex can be used as theory solver for DPLL(T), where often constraints are activated and deactivated
  - simplex can be used as backend for LIA solver (next lecture), where new constraints are added on-the-fly
  - aim: do not restart simplex from scratch when slightly modifying constraints

Incremental Interface for DPLL(T)

- make all constraints available to simplex at the beginning of the algorithm, ignoring Boolean structure $\Longrightarrow$ obtain global tableau for all constraints

Incremental Interface for DPLL(T)

- make all constraints available to simplex at the beginning of the algorithm, ignoring Boolean structure $\Longrightarrow$ obtain global tableau for all constraints
- modify simplex algorithm by adding active-flags for upper and lower bounds

Incremental Interface for DPLL(T)

- make all constraints available to simplex at the beginning of the algorithm, ignoring Boolean structure $\Longrightarrow$ obtain global tableau for all constraints
- modify simplex algorithm by adding active-flags for upper and lower bounds
- when executing simplex, all inactive bounds are ignored

Incremental Interface for DPLL(T)

- make all constraints available to simplex at the beginning of the algorithm, ignoring Boolean structure $\Longrightarrow$ obtain global tableau for all constraints
- modify simplex algorithm by adding active-flags for upper and lower bounds
- when executing simplex, all inactive bounds are ignored
- initially all bounds are inactive

Incremental Interface for DPLL(T)

- make all constraints available to simplex at the beginning of the algorithm, ignoring Boolean structure $\Longrightarrow$ obtain global tableau for all constraints
- modify simplex algorithm by adding active-flags for upper and lower bounds
- when executing simplex, all inactive bounds are ignored
- initially all bounds are inactive
- activation of a constraint

Incremental Interface for DPLL(T)

- make all constraints available to simplex at the beginning of the algorithm, ignoring Boolean structure $\Longrightarrow$ obtain global tableau for all constraints
- modify simplex algorithm by adding active-flags for upper and lower bounds
- when executing simplex, all inactive bounds are ignored
- initially all bounds are inactive
- activation of a constraint
  - activate corresponding lower or upper bound $s_i \leq c$ or $c \leq s_i$ for slack variable s_i

Incremental Interface for DPLL(T)

- make all constraints available to simplex at the beginning of the algorithm, ignoring Boolean structure $\Longrightarrow$ obtain global tableau for all constraints
- modify simplex algorithm by adding active-flags for upper and lower bounds
- when executing simplex, all inactive bounds are ignored
- initially all bounds are inactive
- activation of a constraint
  - activate corresponding lower or upper bound $s_i \leq c$ or $c \leq s_i$ for slack variable s_i
  - if new bound gives rise to a conflict $u_i < l_i$ then report unsatisfiable

Incremental Interface for DPLL(T)

- make all constraints available to simplex at the beginning of the algorithm, ignoring Boolean structure $\Longrightarrow$ obtain global tableau for all constraints
- modify simplex algorithm by adding active-flags for upper and lower bounds
- when executing simplex, all inactive bounds are ignored
- initially all bounds are inactive
- activation of a constraint
  - activate corresponding lower or upper bound $s_i \leq c$ or $c \leq s_i$ for slack variable s_i
  - if new bound gives rise to a conflict $u_i < l_i$ then report unsatisfiable
  - otherwise, if $s_i \in N$ and s_i violates bound, then update assignment of s_i to c (at this point, invariants (1) and (2) are established for the active bounds)

Incremental Interface for DPLL(T)

- make all constraints available to simplex at the beginning of the algorithm, ignoring Boolean structure $\Longrightarrow$ obtain global tableau for all constraints
- modify simplex algorithm by adding active-flags for upper and lower bounds
- when executing simplex, all inactive bounds are ignored
- initially all bounds are inactive
- activation of a constraint
  - activate corresponding lower or upper bound $s_i \leq c$ or $c \leq s_i$ for slack variable s_i
  - if new bound gives rise to a conflict $u_i < l_i$ then report unsatisfiable
  - otherwise, if $s_i \in N$ and s_i violates bound, then update assignment of s_i to c (at this point, invariants (1) and (2) are established for the active bounds)
  - run simplex on current tableau and assignment to determine satisfiability

Incremental Interface for DPLL(T)

- make all constraints available to simplex at the beginning of the algorithm, ignoring Boolean structure $\Longrightarrow$ obtain global tableau for all constraints
- modify simplex algorithm by adding active-flags for upper and lower bounds
- when executing simplex, all inactive bounds are ignored
- initially all bounds are inactive
- activation of a constraint
  - activate corresponding lower or upper bound $s_i \leq c$ or $c \leq s_i$ for slack variable s_i
  - if new bound gives rise to a conflict $u_i < l_i$ then report unsatisfiable
  - otherwise, if $s_i \in N$ and s_i violates bound, then update assignment of s_i to c (at this point, invariants (1) and (2) are established for the active bounds)
  - run simplex on current tableau and assignment to determine satisfiability
- deactivation of a constraint

Incremental Interface for DPLL(T)

- make all constraints available to simplex at the beginning of the algorithm, ignoring Boolean structure $\Longrightarrow$ obtain global tableau for all constraints
- modify simplex algorithm by adding active-flags for upper and lower bounds
- when executing simplex, all inactive bounds are ignored
- initially all bounds are inactive
- activation of a constraint
  - activate corresponding lower or upper bound $s_i \leq c$ or $c \leq s_i$ for slack variable s_i
  - if new bound gives rise to a conflict $u_i < l_i$ then report unsatisfiable
  - otherwise, if $s_i \in N$ and s_i violates bound, then update assignment of s_i to c (at this point, invariants (1) and (2) are established for the active bounds)
  - run simplex on current tableau and assignment to determine satisfiability
- deactivation of a constraint
  - deactivate corresponding lower or upper bound

Incremental Interface for DPLL(T)

- make all constraints available to simplex at the beginning of the algorithm, ignoring Boolean structure $\Longrightarrow$ obtain global tableau for all constraints
- modify simplex algorithm by adding active-flags for upper and lower bounds
- when executing simplex, all inactive bounds are ignored
- initially all bounds are inactive
- activation of a constraint
  - activate corresponding lower or upper bound $s_i \leq c$ or $c \leq s_i$ for slack variable s_i
  - if new bound gives rise to a conflict $u_i < l_i$ then report unsatisfiable
  - otherwise, if $s_i \in N$ and s_i violates bound, then update assignment of s_i to c (at this point, invariants (1) and (2) are established for the active bounds)
  - run simplex on current tableau and assignment to determine satisfiability
- deactivation of a constraint
  - deactivate corresponding lower or upper bound
  - usually occurs after a conflict has been detected

Incremental Interface for DPLL(T)

- make all constraints available to simplex at the beginning of the algorithm, ignoring Boolean structure $\Longrightarrow$ obtain global tableau for all constraints
- modify simplex algorithm by adding active-flags for upper and lower bounds
- when executing simplex, all inactive bounds are ignored
- initially all bounds are inactive
- activation of a constraint
  - activate corresponding lower or upper bound $s_i \leq c$ or $c \leq s_i$ for slack variable s_i
  - if new bound gives rise to a conflict $u_i < l_i$ then report unsatisfiable
  - otherwise, if $s_i \in N$ and s_i violates bound, then update assignment of s_i to c (at this point, invariants (1) and (2) are established for the active bounds)
  - run simplex on current tableau and assignment to determine satisfiability
- deactivation of a constraint
  - deactivate corresponding lower or upper bound
  - usually occurs after a conflict has been detected
  - tableau and assignment can be reused from before: they satisfy invariants (1) and (2)

Example (Simplex as DPLL(T) Theory Solver)

- input is formula φ

$$(c_1 \vee c_5) \wedge (c_6 \vee \neg c_3) \wedge (c_4 \vee c_5) \wedge (c_2 \vee c_7 \vee \neg c_8) \wedge \dots$$

$$\underbrace{x \geq 5}_{c_1} \qquad \underbrace{2x + y \leq 12}_{c_2} \qquad \underbrace{y < 3}_{c_3} \qquad \underbrace{x - 3y \leq 2}_{c_4} \qquad \dots$$

- example execution (without theory propagation)
  - start simplex with tableau for all atoms and negated atoms within φ, but no activated bounds (obtain tableau T_0, assignment $v_0(x) = 0$)

Example (Simplex as DPLL(T) Theory Solver)

- input is formula φ

$$(c_1 \vee c_5) \wedge (c_6 \vee \neg c_3) \wedge (c_4 \vee c_5) \wedge (c_2 \vee c_7 \vee \neg c_8) \wedge \dots$$

$$\underbrace{x \geq 5}_{c_1} \qquad \underbrace{2x + y \leq 12}_{c_2} \qquad \underbrace{y < 3}_{c_3} \qquad \underbrace{x - 3y \leq 2}_{c_4} \qquad \dots$$

- example execution (without theory propagation)
  - start simplex with tableau for all atoms and negated atoms within φ, but no activated bounds (obtain tableau T_0, assignment $v_0(x) = 0$)
  - assume partial Boolean assignment $c_1, \neg c_3, c_4$ from Boolean solver $\Longrightarrow$ activate the bounds, execute simplex on T_0 and v_0 to obtain T_1 and v_1

Example (Simplex as DPLL(T) Theory Solver)

- input is formula φ

$$(c_1 \vee c_5) \wedge (c_6 \vee \neg c_3) \wedge (c_4 \vee c_5) \wedge (c_2 \vee c_7 \vee \neg c_8) \wedge \dots$$

$$\underbrace{x \geq 5}_{c_1} \qquad \underbrace{2x + y \leq 12}_{c_2} \qquad \underbrace{y < 3}_{c_3} \qquad \underbrace{x - 3y \leq 2}_{c_4} \qquad \dots$$

- example execution (without theory propagation)
  - start simplex with tableau for all atoms and negated atoms within φ, but no activated bounds (obtain tableau T_0, assignment $v_0(x) = 0$)
  - assume partial Boolean assignment $c_1, \neg c_3, c_4$ from Boolean solver $\Longrightarrow$ activate the bounds, execute simplex on T_0 and v_0 to obtain T_1 and v_1
  - all bounds are satisfied, so detect LRA-consistency

Example (Simplex as DPLL(T) Theory Solver)

- input is formula φ

$$(c_1 \vee c_5) \wedge (c_6 \vee \neg c_3) \wedge (c_4 \vee c_5) \wedge (c_2 \vee c_7 \vee \neg c_8) \wedge \dots$$

$$\underbrace{x \geq 5}_{c_1} \qquad \underbrace{2x + y \leq 12}_{c_2} \qquad \underbrace{y < 3}_{c_3} \qquad \underbrace{x - 3y \leq 2}_{c_4} \qquad \dots$$

- example execution (without theory propagation)
  - start simplex with tableau for all atoms and negated atoms within φ, but no activated bounds (obtain tableau T_0, assignment $v_0(x) = 0$)
  - assume partial Boolean assignment $c_1, \neg c_3, c_4$ from Boolean solver $\Longrightarrow$ activate the bounds, execute simplex on T_0 and v_0 to obtain T_1 and v_1
  - all bounds are satisfied, so detect LRA-consistency
  - extend to $c_1, \neg c_3, c_4, c_2$, activate bound, execute simplex on T_1 and v_1 to obtain T_2 and v_2

Example (Simplex as DPLL(T) Theory Solver)

- input is formula φ

$$(c_1 \vee c_5) \wedge (c_6 \vee \neg c_3) \wedge (c_4 \vee c_5) \wedge (c_2 \vee c_7 \vee \neg c_8) \wedge \dots$$

$$\underbrace{x \geq 5}_{c_1} \qquad \underbrace{2x + y \leq 12}_{c_2} \qquad \underbrace{y < 3}_{c_3} \qquad \underbrace{x - 3y \leq 2}_{c_4} \qquad \dots$$

- example execution (without theory propagation)
  - start simplex with tableau for all atoms and negated atoms within φ, but no activated bounds (obtain tableau T_0, assignment $v_0(x) = 0$)
  - assume partial Boolean assignment $c_1, \neg c_3, c_4$ from Boolean solver $\Longrightarrow$ activate the bounds, execute simplex on T_0 and v_0 to obtain T_1 and v_1
  - all bounds are satisfied, so detect LRA-consistency
  - extend to $c_1, \neg c_3, c_4, c_2$, activate bound, execute simplex on T_1 and v_1 to obtain T_2 and v_2
  - detect unsatisfiable core $c_1, \neg c_3, c_2$, so learn $\neg c_1 \vee c_3 \vee \neg c_2$

Example (Simplex as DPLL(T) Theory Solver)

- input is formula φ

$$(c_1 \vee c_5) \wedge (c_6 \vee \neg c_3) \wedge (c_4 \vee c_5) \wedge (c_2 \vee c_7 \vee \neg c_8) \wedge \dots$$

$$\underbrace{x \geq 5}_{c_1} \qquad \underbrace{2x + y \leq 12}_{c_2} \qquad \underbrace{y < 3}_{c_3} \qquad \underbrace{x - 3y \leq 2}_{c_4} \qquad \dots$$

- example execution (without theory propagation)
  - start simplex with tableau for all atoms and negated atoms within φ, but no activated bounds (obtain tableau T_0, assignment $v_0(x) = 0$)
  - assume partial Boolean assignment $c_1, \neg c_3, c_4$ from Boolean solver $\Longrightarrow$ activate the bounds, execute simplex on T_0 and v_0 to obtain T_1 and v_1
  - all bounds are satisfied, so detect LRA-consistency
  - extend to $c_1, \neg c_3, c_4, c_2$, activate bound, execute simplex on T_1 and v_1 to obtain T_2 and v_2
  - detect unsatisfiable core $c_1, \neg c_3, c_2$, so learn $\neg c_1 \vee c_3 \vee \neg c_2$
  - ... next simplex invocation starting from T_2 and v_2 ...

Incremental Simplex Algorithm – Interface

- initSimplex: takes list of indexed constraints, initially all inactive

Incremental Simplex Algorithm – Interface

- initSimplex: takes list of indexed constraints, initially all inactive
- assert i: activates constraint with index i, may detect unsat core

Incremental Simplex Algorithm – Interface

- initSimplex: takes list of indexed constraints, initially all inactive
- assert i: activates constraint with index i, may detect unsat core
- check: check if currently activated constraints are satisfiable, may detect unsat core

Incremental Simplex Algorithm – Interface

- initSimplex: takes list of indexed constraints, initially all inactive
- assert i: activates constraint with index i, may detect unsat core
- check: check if currently activated constraints are satisfiable, may detect unsat core
- solution: after successful check, deliver satisfying assignment

Incremental Simplex Algorithm – Interface

- initSimplex: takes list of indexed constraints, initially all inactive
- assert i: activates constraint with index i, may detect unsat core
- check: check if currently activated constraints are satisfiable, may detect unsat core
- solution: after successful check, deliver satisfying assignment
- checkpoint: after successful check, get checkpoint information to return to this state

Incremental Simplex Algorithm – Interface

- initSimplex: takes list of indexed constraints, initially all inactive
- assert i: activates constraint with index i, may detect unsat core
- check: check if currently activated constraints are satisfiable, may detect unsat core
- solution: after successful check, deliver satisfying assignment
- checkpoint: after successful check, get checkpoint information to return to this state
- backtrack cp: return to previous checkpoint cp, where subset of constraints has been asserted

Incremental Simplex Algorithm – Interface

- initSimplex: takes list of indexed constraints, initially all inactive
- assert i: activates constraint with index i, may detect unsat core
- check: check if currently activated constraints are satisfiable, may detect unsat core
- solution: after successful check, deliver satisfying assignment
- checkpoint: after successful check, get checkpoint information to return to this state
- backtrack cp: return to previous checkpoint cp, where subset of constraints has been asserted

Haskell Implementation

- available under
  `http://cl-informatik.uibk.ac.at/teaching/ss24/cs/simplex.tgz`

Incremental Simplex Algorithm – Interface

- initSimplex: takes list of indexed constraints, initially all inactive
- assert i: activates constraint with index i, may detect unsat core
- check: check if currently activated constraints are satisfiable, may detect unsat core
- solution: after successful check, deliver satisfying assignment
- checkpoint: after successful check, get checkpoint information to return to this state
- backtrack cp: return to previous checkpoint cp, where subset of constraints has been asserted

Haskell Implementation

- available under
  `http://cl-informatik.uibk.ac.at/teaching/ss24/cs/simplex.tgz`
- `SimplexInternals.hs` – verified simplex implementation

Incremental Simplex Algorithm – Interface

- initSimplex: takes list of indexed constraints, initially all inactive
- assert i: activates constraint with index i, may detect unsat core
- check: check if currently activated constraints are satisfiable, may detect unsat core
- solution: after successful check, deliver satisfying assignment
- checkpoint: after successful check, get checkpoint information to return to this state
- backtrack cp: return to previous checkpoint cp, where subset of constraints has been asserted

Haskell Implementation

- available under
  `http://cl-informatik.uibk.ac.at/teaching/ss24/cs/simplex.tgz`
- `SimplexInternals.hs` – verified simplex implementation
- `SimplexCommon.hs` – common interface to access verified types

Incremental Simplex Algorithm – Interface

- initSimplex: takes list of indexed constraints, initially all inactive
- assert i: activates constraint with index i, may detect unsat core
- check: check if currently activated constraints are satisfiable, may detect unsat core
- solution: after successful check, deliver satisfying assignment
- checkpoint: after successful check, get checkpoint information to return to this state
- backtrack cp: return to previous checkpoint cp, where subset of constraints has been asserted

Haskell Implementation

- available under `http://cl-informatik.uibk.ac.at/teaching/ss24/cs/simplex.tgz`
- `SimplexInternals.hs` – verified simplex implementation
- `SimplexCommon.hs` – common interface to access verified types
- `SimplexInterface.hs` – wrapper for non-incremental simplex algorithm + example

Incremental Simplex Algorithm – Interface

- initSimplex: takes list of indexed constraints, initially all inactive
- assert i: activates constraint with index i, may detect unsat core
- check: check if currently activated constraints are satisfiable, may detect unsat core
- solution: after successful check, deliver satisfying assignment
- checkpoint: after successful check, get checkpoint information to return to this state
- backtrack cp: return to previous checkpoint cp, where subset of constraints has been asserted

Haskell Implementation

- available under
  `http://cl-informatik.uibk.ac.at/teaching/ss24/cs/simplex.tgz`
- `SimplexInternals.hs` – verified simplex implementation
- `SimplexCommon.hs` – common interface to access verified types
- `SimplexInterface.hs` – wrapper for non-incremental simplex algorithm + example
- `SimplexIOInterface.hs` – wrapper for incremental simplex algorithm + example

`ghci SimplexIOInterface.hs`

```
ghci SimplexIOInterface.hs
ghci>
```

```
ghci SimplexIOInterface.hs
ghci> initSimplex exampleSlidesInput
```

```
ghci SimplexIOInterface.hs
ghci> initSimplex exampleSlidesInput
Okay <state 0, asserted: []>
ghci>
```

```
ghci SimplexIOInterface.hs
ghci> initSimplex exampleSlidesInput
Okay <state 0, asserted: []>
ghci> assert 1
Okay <state 1, asserted: [1]>
ghci>
```

```
ghci SimplexIOInterface.hs
ghci> initSimplex exampleSlidesInput
Okay <state 0, asserted: []>
ghci> assert 1
Okay <state 1, asserted: [1]>
ghci> assert (-3)
Okay <state 2, asserted: [-3,1]>
ghci> assert 4
Okay <state 3, asserted: [4,-3,1]>
ghci>
```

```
ghci SimplexIOInterface.hs
ghci> initSimplex exampleSlidesInput
Okay <state 0, asserted: []>
ghci> assert 1
Okay <state 1, asserted: [1]>
ghci> assert (-3)
Okay <state 2, asserted: [-3,1]>
ghci> assert 4
Okay <state 3, asserted: [4,-3,1]>
ghci> check
Okay <state 4, asserted: [4,-3,1]>
ghci>
```

```
ghci SimplexIOInterface.hs
ghci> initSimplex exampleSlidesInput
Okay <state 0, asserted: []>
ghci> assert 1
Okay <state 1, asserted: [1]>
ghci> assert (-3)
Okay <state 2, asserted: [-3,1]>
ghci> assert 4
Okay <state 3, asserted: [4,-3,1]>
ghci> check
Okay <state 4, asserted: [4,-3,1]>
ghci> checkpoint
checkpoint "cp5" created
Okay <state 5, asserted: [4,-3,1]>
ghci>
```

```
ghci SimplexIOInterface.hs
ghci> initSimplex exampleSlidesInput
Okay <state 0, asserted: []>
ghci> assert 1
Okay <state 1, asserted: [1]>
ghci> assert (-3)
Okay <state 2, asserted: [-3,1]>
ghci> assert 4
Okay <state 3, asserted: [4,-3,1]>
ghci> check
Okay <state 4, asserted: [4,-3,1]>
ghci> checkpoint
checkpoint "cp5" created
Okay <state 5, asserted: [4,-3,1]>
ghci> assert 2
Okay <state 6, asserted: [2,4,-3,1]>
ghci> check
unsat-core [2,1,-3] detected, use backtrack to one of ["cp5"]
ghci>
```

```
ghci SimplexIOInterface.hs
ghci> initSimplex exampleSlidesInput
Okay <state 0, asserted: []>
ghci> assert 1
Okay <state 1, asserted: [1]>
ghci> assert (-3)
Okay <state 2, asserted: [-3,1]>
ghci> assert 4
Okay <state 3, asserted: [4,-3,1]>
ghci> check
Okay <state 4, asserted: [4,-3,1]>
ghci> checkpoint
checkpoint "cp5" created
Okay <state 5, asserted: [4,-3,1]>
ghci> assert 2
Okay <state 6, asserted: [2,4,-3,1]>
ghci> check
unsat-core [2,1,-3] detected, use backtrack to one of ["cp5"]
ghci> backtrack "cp5"
Okay <state 7, asserted: [4,-3,1]>
```

Linear Programming

- linear programming: find solution in $\mathbb{Q}$ of linear constraints φ that maximizes linear function f

---

[a] Alexander Schrijver, Theory of Linear and Integer Programming, Chapter 11

Linear Programming

- linear programming: find solution in $\mathbb{Q}$ of linear constraints φ that maximizes linear function f
- potential implementation

[a] Alexander Schrijver, Theory of Linear and Integer Programming, Chapter 11

Linear Programming

- linear programming: find solution in $\mathbb{Q}$ of linear constraints φ that maximizes linear function f
- potential implementation
  - perform standard setup for running simplex on φ

---

[a] Alexander Schrijver, Theory of Linear and Integer Programming, Chapter 11

Linear Programming

- linear programming: find solution in $\mathbb{Q}$ of linear constraints φ that maximizes linear function f
- potential implementation
  - perform standard setup for running simplex on φ
  - add additional slack variable s with tableau equality $s = f$, then start simplex

[a] Alexander Schrijver, Theory of Linear and Integer Programming, Chapter 11

Linear Programming

- linear programming: find solution in $\mathbb{Q}$ of linear constraints φ that maximizes linear function f
- potential implementation
  - perform standard setup for running simplex on φ
  - add additional slack variable s with tableau equality $s = f$, then start simplex
  - once solution v has been detected, compute $f(v)$ and change bound of s to $s \geq f(v) + \delta$

---

[a] Alexander Schrijver, Theory of Linear and Integer Programming, Chapter 11

Linear Programming

- linear programming: find solution in $\mathbb{Q}$ of linear constraints φ that maximizes linear function f
- potential implementation
  - perform standard setup for running simplex on φ
  - add additional slack variable s with tableau equality $s = f$, then start simplex
  - once solution v has been detected, compute $f(v)$ and change bound of s to $s \geq f(v) + \delta$
  - iterate to find better solution, or detect optimality if unsat is returned

[a] Alexander Schrijver, Theory of Linear and Integer Programming, Chapter 11

Linear Programming

- linear programming: find solution in $\mathbb{Q}$ of linear constraints φ that maximizes linear function f
- potential implementation
  - perform standard setup for running simplex on φ
  - add additional slack variable s with tableau equality $s = f$, then start simplex
  - once solution v has been detected, compute $f(v)$ and change bound of s to $s \geq f(v) + \delta$
  - iterate to find better solution, or detect optimality if unsat is returned
- problem: algorithm does not terminate if f can be increased arbitrarily

[a] Alexander Schrijver, Theory of Linear and Integer Programming, Chapter 11

Linear Programming

- linear programming: find solution in $\mathbb{Q}$ of linear constraints φ that maximizes linear function f
- potential implementation
  - perform standard setup for running simplex on φ
  - add additional slack variable s with tableau equality $s = f$, then start simplex
  - once solution v has been detected, compute $f(v)$ and change bound of s to $s \geq f(v) + \delta$
  - iterate to find better solution, or detect optimality if unsat is returned
- problem: algorithm does not terminate if f can be increased arbitrarily
- solution: use standard simplex algorithm for linear programming,[a] and not the DPLL(T)-variant of simplex for decidability that was presented here

[a] Alexander Schrijver, Theory of Linear and Integer Programming, Chapter 11

Linear Programming

- linear programming: find solution in $\mathbb{Q}$ of linear constraints φ that maximizes linear function f
- potential implementation
  - perform standard setup for running simplex on φ
  - add additional slack variable s with tableau equality $s = f$, then start simplex
  - once solution v has been detected, compute $f(v)$ and change bound of s to $s \geq f(v) + \delta$
  - iterate to find better solution, or detect optimality if unsat is returned
- problem: algorithm does not terminate if f can be increased arbitrarily
- solution: use standard simplex algorithm for linear programming,[a] and not the DPLL(T)-variant of simplex for decidability that was presented here
  - at least one difference: solving $A\vec{x} \leq \vec{b}$ is formulated via slack variables $\vec{s} \geq \vec{0}$ as tableau $A\vec{x} + \vec{s} = \vec{b}$ so the tableau equations can have non-zero constants

---

[a] Alexander Schrijver, Theory of Linear and Integer Programming, Chapter 11

Outline

1. **Summary of Previous Lecture**
2. **Complexity of Simplex Algorithm**
3. **Unsatisfiable Cores and Farkas' Lemma**
4. **Simplex Algorithm for DPLL($\mathcal{T}$)**
5. **Further Reading**

Kröning and Strichmann

- Sections 5.1 and 5.2

Further Reading

Bruno Dutertre and Leonardo de Moura.
A Fast Linear-Arithmetic Solver for DPLL(T)
In Proc. CAV, LNCS 4144, pp. 81–94, 2006.

Bertram Felgenhauer and Aart Middeldorp
Constructing Cycles in the Simplex Method for DPLL(T)
Proc. 14th ICTAC, LNCS 10580, pp. 213–228, 2017

Andreas Podelski and Andrey Rybalchenko
A Complete Method for the Synthesis of Linear Ranking Functions
Proc. VMCAI 2004, LNCS 2937, pp. 239–251, 2004

Ralph Bottesch, Max W. Haslbeck, and René Thiemann
Verifying an Incremental Theory Solver for Linear Arithmetic in Isabelle/HOL
In Proc. FroCoS, LNAI 11715, pp. 223—239, 2019.

Important Concepts

- active and inactive bounds
- Bland’s selection rule
- Farkas’ coefficients
- Farkas’ lemma
- incremental simplex algorithm
- linear programming
- linear ranking function
- unsatisfiable core