Automated Reasoning

Cynthia Kop & Sebastian Junges
Fall 2024

Lecture 6:
SMT for Linear Arithmetic

Slides inspired by:
Erika Abraham, RWTH,
Ashutosh Gupta, IITB,
Kroening & Strichman, Decision Procedures

Today's Lecture

- SMT with linear arithmetic & linear programming
- Generalized Simplex
- SMT-compliant Simplex
- Branch & Bound
- Difference logic

Goal:

Learn methods for automated reasoning about linear constraints in SMT

Linear Programming

An optimization problem

- Classic example with integer valued variables

A factory makes two products: Doors (D) and Frames (F).

- Constructing a D yields 70 euro profit
- Constructing a F yields 50 euro profit
- Building a D requires 2 units of wood, a F requires 1 unit of wood
- Building a D requires 4 hours, a F requires 3 hours
- We have 240 hours and 100 units of wood
- How to maximize profit?

Linear Programming

An optimization problem

- Classic example with real-valued variables

A factory makes two products: Beer (B) and Soda (S).

- Brewing B yields 70 euro profit
- Brewing HL of S yields 50 euro profit
- Brewing B requires 2 units of water, S requires 1 unit of water
- Brewing B requires 4 hours, S requires 3 hours
- We have 240 hours and 100 units of water
- How to maximize profit?

Encoding

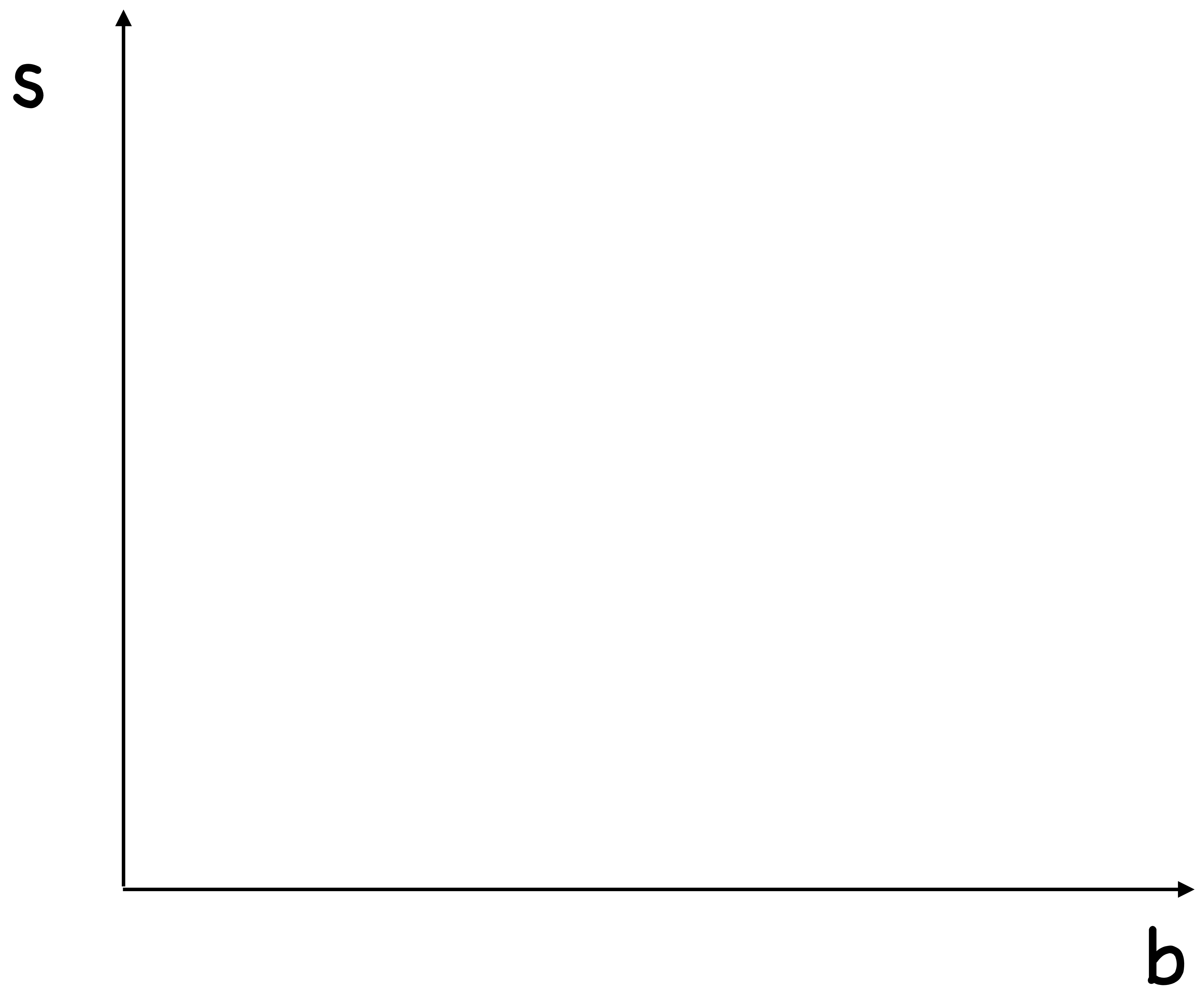
maximize $70b + 50s$ s.t.

$$4 \cdot b + 3 \cdot s \leq 240$$
$$2 \cdot b + 1 \cdot s \leq 100$$
$$b \geq 0$$
$$s \geq 0$$

Encoding

maximize $70b + 50s$ s.t.

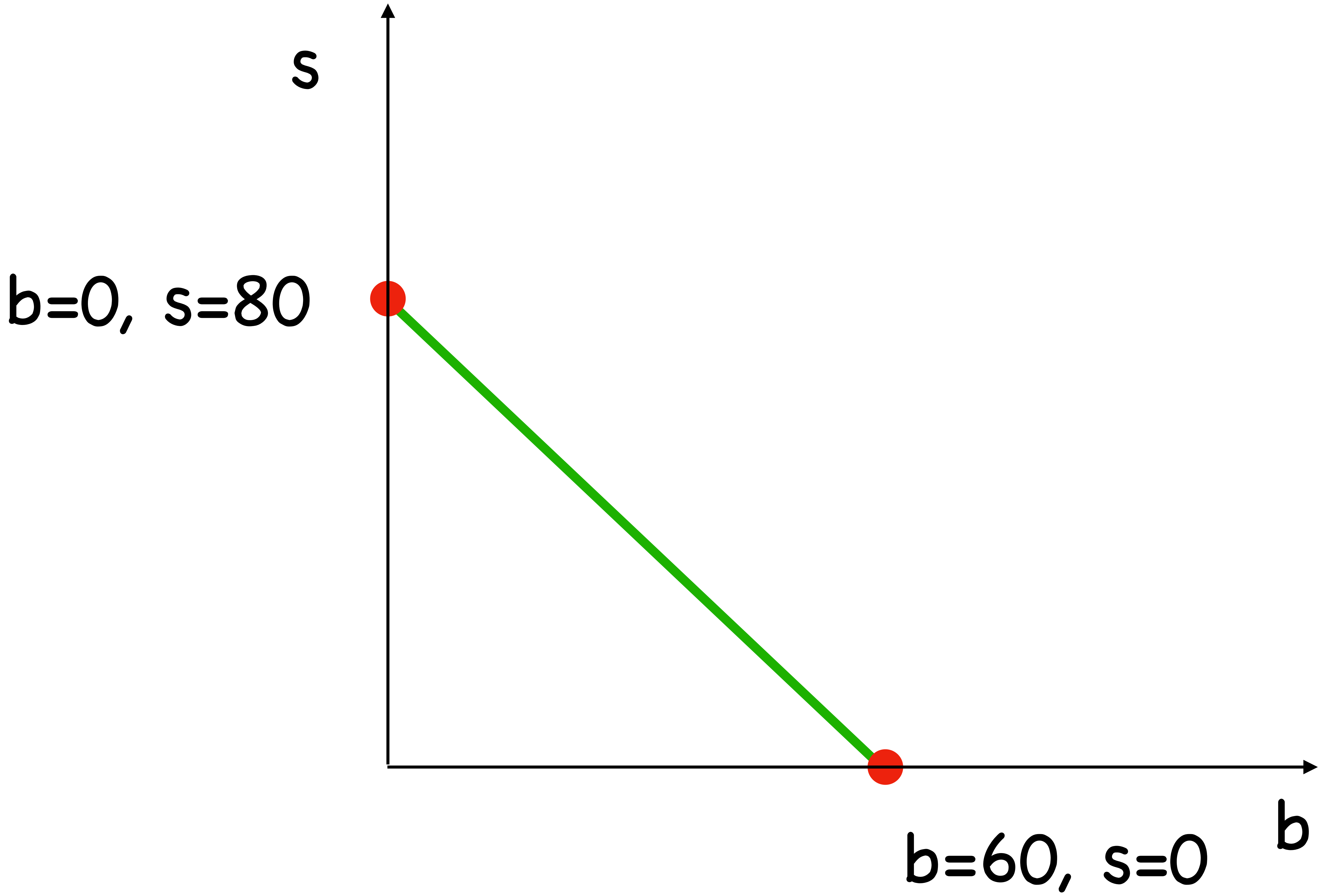
$$
\begin{aligned}
&4 \cdot b + 3 \cdot s \leq 240 \\
&2 \cdot b + 1 \cdot s \leq 100 \\
&b \geq 0 \\
&s \geq 0
\end{aligned}
$$



Encoding

maximize $70b + 50s$ s.t.

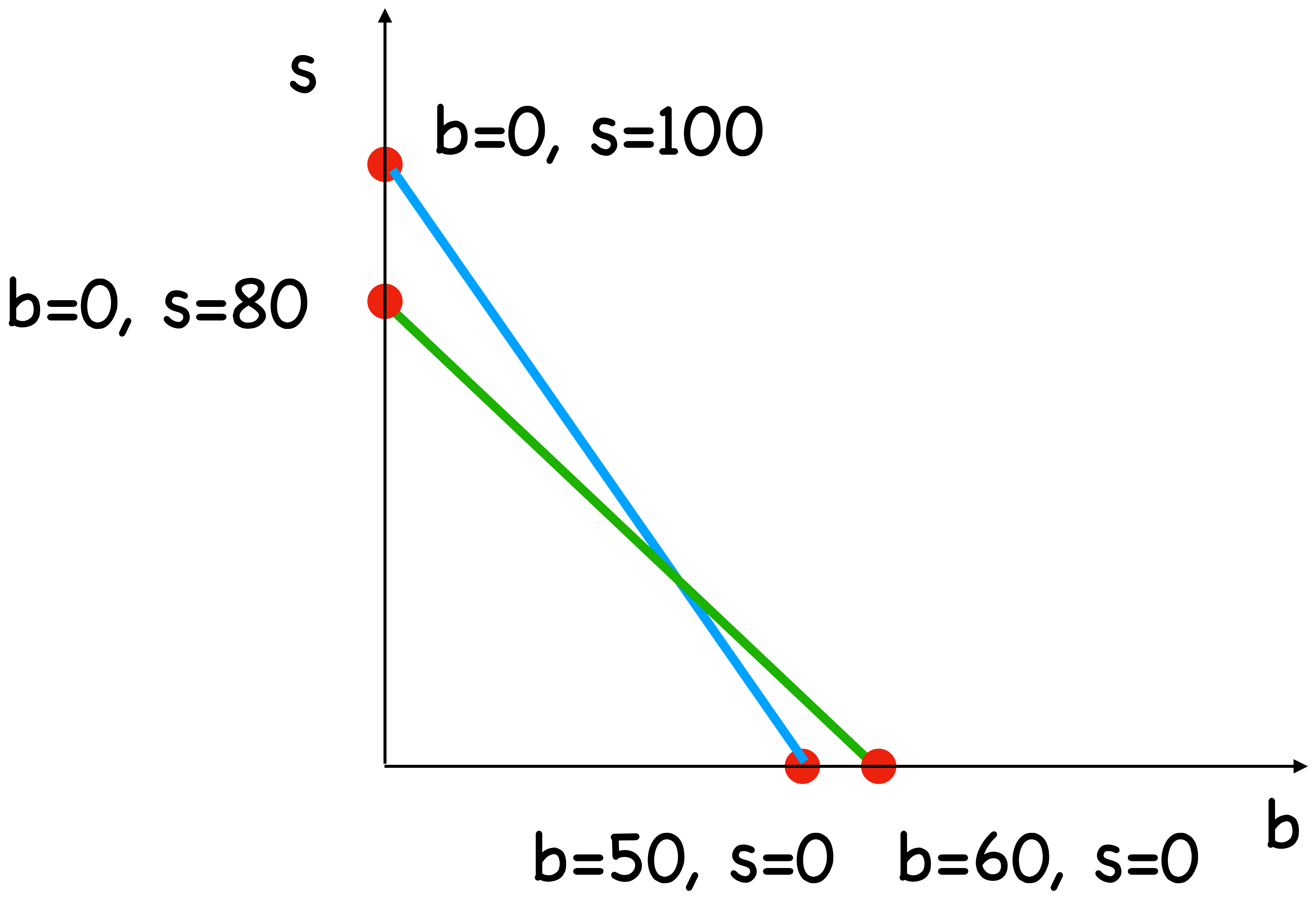
$$4 \cdot b + 3 \cdot s \leq 240$$
$$2 \cdot b + 1 \cdot s \leq 100$$
$$b \geq 0$$
$$s \geq 0$$



Encoding

maximize $70b + 50s$ s.t.

$$4 \cdot b + 3 \cdot s \leq 240$$
$$2 \cdot b + 1 \cdot s \leq 100$$
$$b \geq 0$$
$$s \geq 0$$



Encoding

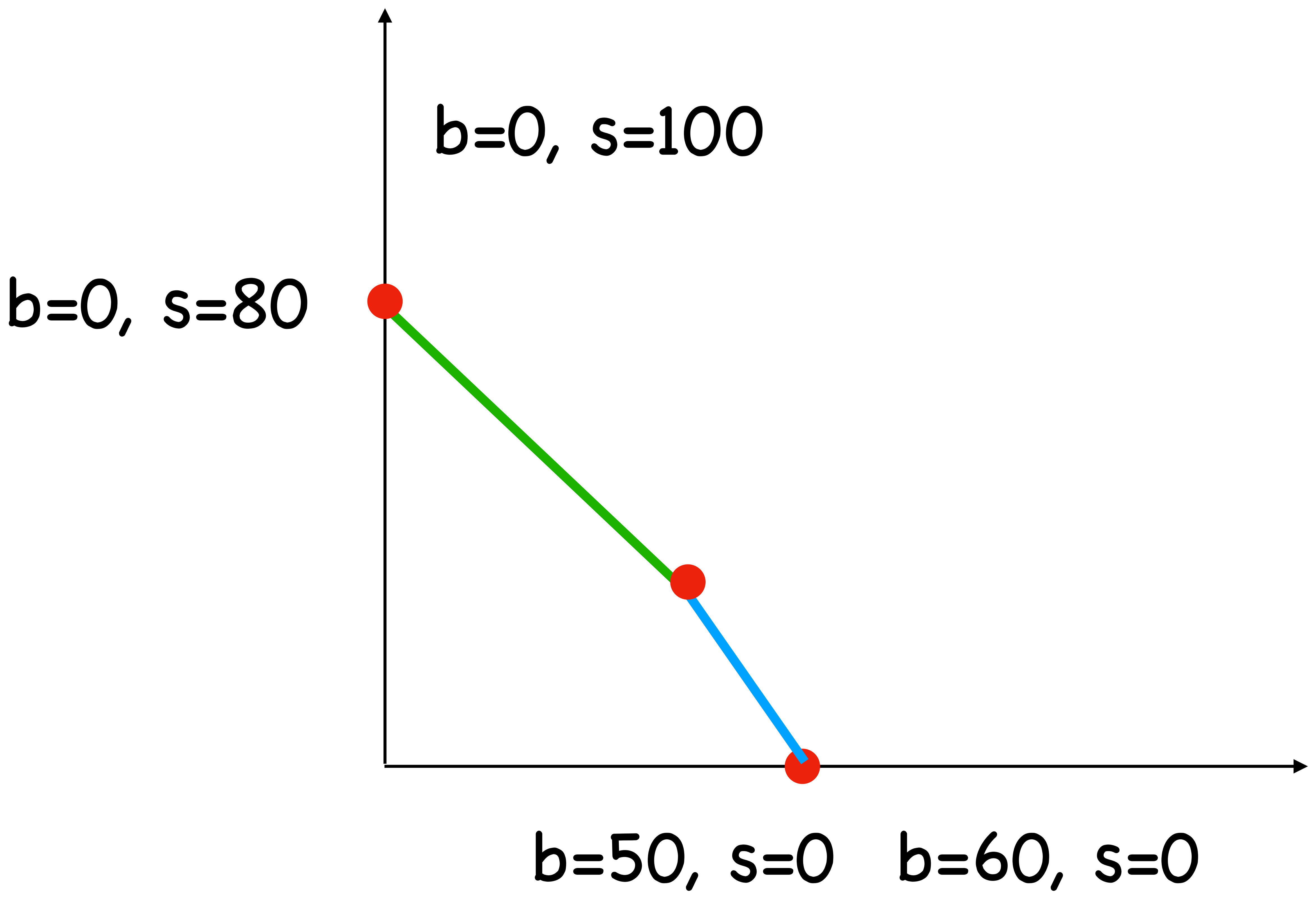
maximize $70b + 50s$ s.t.

$$4 \cdot b + 3 \cdot s \leq 240$$
$$2 \cdot b + 1 \cdot s \leq 100$$
$$b \geq 0$$
$$s \geq 0$$

Encoding

maximize $70b + 50s$ s.t.

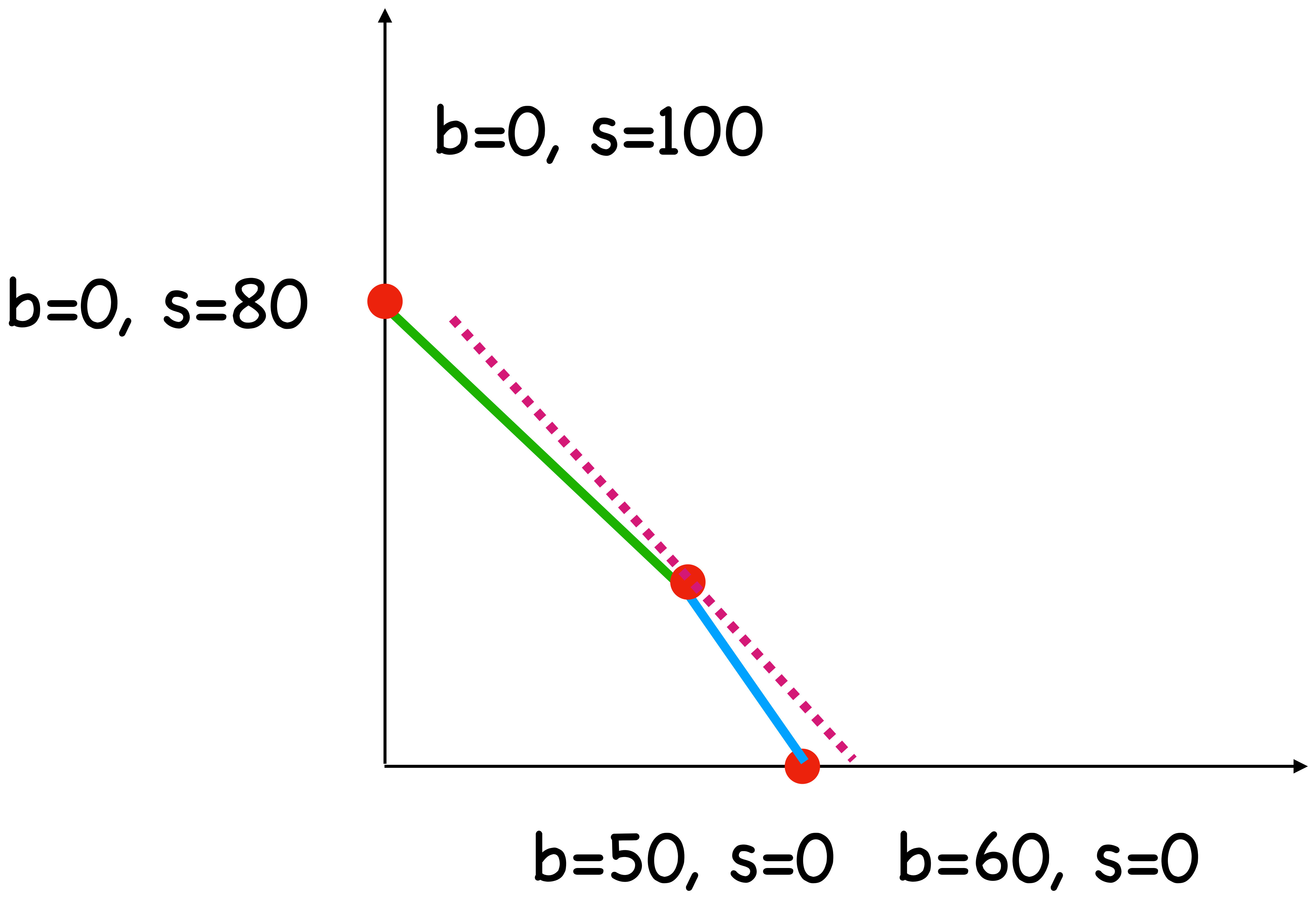
$$
\begin{aligned}
&4 \cdot b + 3 \cdot s \leq 240 \\
&2 \cdot b + 1 \cdot s \leq 100 \\
&b \geq 0 \\
&s \geq 0
\end{aligned}
$$



Encoding

maximize $70b + 50s$ s.t.

$$4 \cdot b + 3 \cdot s \leq 240$$
$$2 \cdot b + 1 \cdot s \leq 100$$
$$b \geq 0$$
$$s \geq 0$$



Encoding

Matrix Normal Form

maximize $70b + 50s$ s.t.

$$
\begin{aligned}
&-4 \cdot b - 3 \cdot s \geq -240 \\
&-2 \cdot b - 1 \cdot s \geq -100 \\
&b \geq 0 \\
&s \geq 0
\end{aligned}
$$

Encoding

Matrix Normal Form

maximize $70b + 50s$ s.t.

$$
\begin{aligned}
-4 \cdot b - 3 \cdot s &\geq -240 \\
-2 \cdot b - 1 \cdot s &\geq -100 \\
b &\geq 0 \\
s &\geq 0
\end{aligned}
$$

$$
\begin{bmatrix} -4 & -3 \\ -2 & -1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} b \\ s \end{bmatrix} \geq \begin{bmatrix} -240 \\ -100 \\ 0 \\ 0 \end{bmatrix}
$$

Quantifier-Free Linear Real Arithmetic

QF_LRA

Quantifier-Free Linear Real Arithmetic

QF_LRA

- Real valued variables, Constants

Quantifier-Free Linear Real Arithmetic

QF_LRA

- Real valued variables, Constants
- Addition between them

Quantifier-Free Linear Real Arithmetic

QF_LRA

- Real valued variables, Constants
- Addition between them
- Inequalities (focus on $\leq$, $\geq$)

Quantifier-Free Linear Real Arithmetic

QF_LRA

- Real valued variables, Constants
- Addition between them
- Inequalities (focus on $\leq$, $\geq$)
- and their Boolean combination

Quantifier-Free Linear Real Arithmetic

QF_LRA

- Real valued variables, Constants
- Addition between them
- Inequalities (focus on $\leq$, $\geq$)
- and their Boolean combination

Multiplication only with constants as syntactic sugar for addition…

Quantifier-Free Linear Real Arithmetic

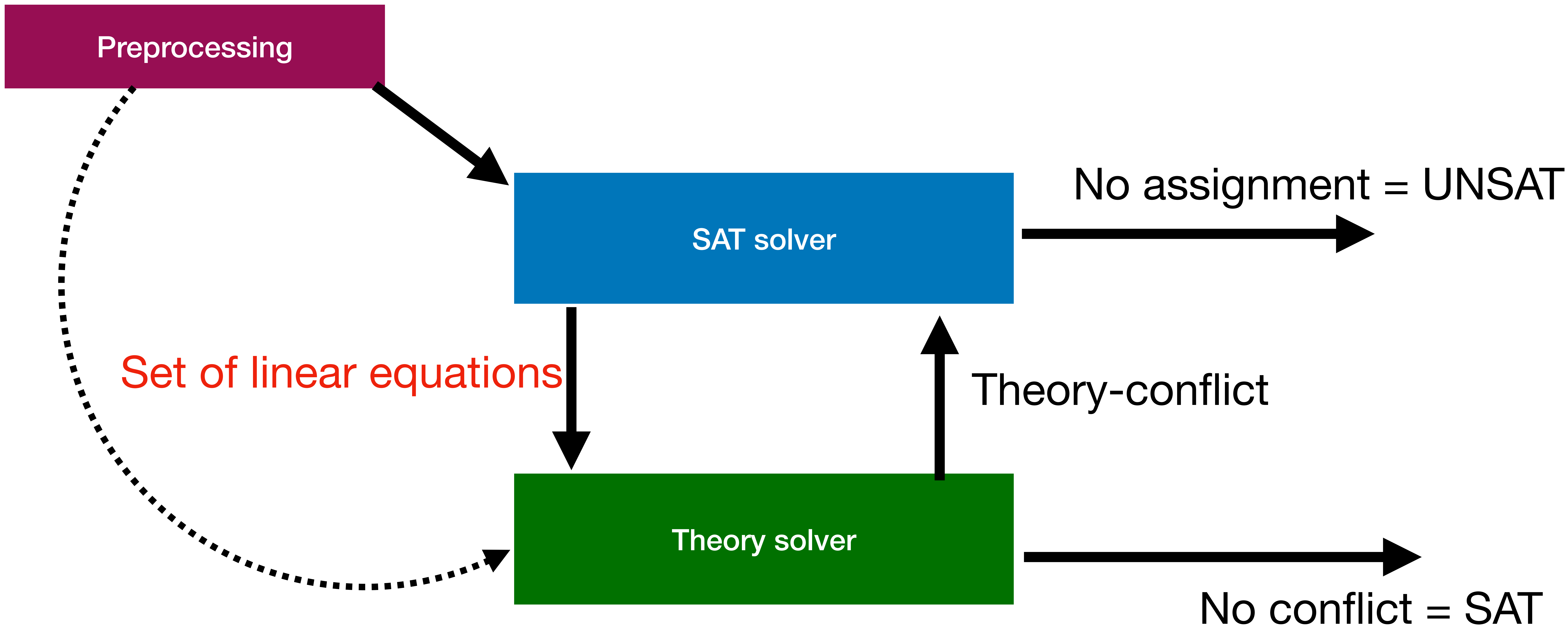
QF_LRA

- Real valued variables, Constants
- Addition between them
- Inequalities (focus on $\leq$, $\geq$)
- and their Boolean combination

Multiplication only with constants as syntactic sugar for addition…

Adds arbitrary Boolean combination of constraints to linear programming, but removes the objective (requires a constant objective)

DPLL(T) for QF_LRA



How to solve Linear Programs

How to solve Linear Programs

- Using an off-the-shelf LP solver:
  GLPK, GLOP, LPsolve, Soplex, Gurobi, Mosek, CPlex, Cardinal Opt, ….

How to solve Linear Programs

- Using an off-the-shelf LP solver:
  GLPK, GLOP, LPsolve, Soplex, Gurobi, Mosek, CPlex, Cardinal Opt, ….

How to solve Linear Programs

- Using an off-the-shelf LP solver:
  GLPK, GLOP, LPsolve, Soplex, Gurobi, Mosek, CPlex, Cardinal Opt, ….

- Simplex algorithm (1920s); worst-case exponential runtime

How to solve Linear Programs

- Using an off-the-shelf LP solver:
  GLPK, GLOP, LPsolve, Soplex, Gurobi, Mosek, CPlex, Cardinal Opt, ….

- Simplex algorithm (1920s); worst-case exponential runtime

- Interior point or barrier methods (1979); polynomial runtime

How to solve Linear Programs

- Using an off-the-shelf LP solver:
  GLPK, GLOP, LPsolve, Soplex, Gurobi, Mosek, CPlex, Cardinal Opt, ….

- Simplex algorithm (1920s); worst-case exponential runtime
- Interior point or barrier methods (1979); polynomial runtime
- (Variations of) simplex still practically superior in many cases

How to solve Linear Programs

- Using an off-the-shelf LP solver:
  GLPK, GLOP, LPsolve, Soplex, Gurobi, Mosek, CPlex, Cardinal Opt, ….

- Simplex algorithm (1920s); worst-case exponential runtime
- Interior point or barrier methods (1979); polynomial runtime
- (Variations of) simplex still practically superior in many cases
- For precise arithmetic and SMT-compliant theory solving, simplex is standard

Summary

- Linear programming & QF_LRA
- Generalized Simplex
- Incrementality & Backtracking for Generalized Simplex
- Branch & Bound for QF_LIA
- Difference logic

Generalized Simplex

- Consider the feasibility of linear equations of the following form

$$A \cdot \begin{bmatrix} x_1 \\ \dots \\ x_n \\ s_1 \\ \dots \\ s_m \end{bmatrix} = 0, \quad \bigwedge_{1 \le i \le m} l_i \le s_i \le u_i$$

i.e.,
variables $x_1, \dots, x_n$, unbounded, and additional variables $s_1, \dots, s_m$ bounded

Generalized Simplex

- Consider the feasibility of linear equations of the following form

$$A \cdot \begin{bmatrix} x_1 \\ \dots \\ x_n \\ s_1 \\ \dots \\ s_m \end{bmatrix} = 0, \quad \bigwedge_{1 \leq i \leq m} l_i \leq s_i \leq u_i$$

Simplex for optimization starts with a feasible point while our goal is to find a feasible point
(but is similar in almost all other aspects)

i.e.,
variables $x_1, \ldots, x_n$, unbounded, and additional variables $s_1, \ldots, s_m$ bounded

Quiztime

Quiztime

Bring $x + y \geq 2$ into generalized form

Example

$$x_1 + x_2 \geq 2$$
$$2x_1 - x_2 \geq 0$$
$$-x_1 - 2x_2 \geq 1$$

Example

$$\begin{aligned} x_1 + x_2 &\geq 2 \\ 2x_1 - x_2 &\geq 0 \\ -x_1 - 2x_2 &\geq 1 \end{aligned} \Rightarrow \begin{bmatrix} 1 & 1 \\ 2 & -1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \geq \begin{bmatrix} 2 \\ 0 \\ 1 \end{bmatrix}$$

Example

$$x_1 + x_2 \geq 2$$
$$2x_1 - x_2 \geq 0$$
$$-x_1 - 2x_2 \geq 1$$

$$\Rightarrow \begin{bmatrix} 1 & 1 \\ 2 & -1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \geq \begin{bmatrix} 2 \\ 0 \\ 1 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} 1 & 1 & -1 & 0 & 0 \\ 2 & -1 & 0 & -1 & 0 \\ -1 & 2 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Example

$$x_1 + x_2 \geq 2$$
$$2x_1 - x_2 \geq 0$$
$$-x_1 - 2x_2 \geq 1$$

$$\Rightarrow \begin{bmatrix} 1 & 1 \\ 2 & -1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \geq \begin{bmatrix} 2 \\ 0 \\ 1 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} 1 & 1 & -1 & 0 & 0 \\ 2 & -1 & 0 & -1 & 0 \\ -1 & 2 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$ and $s_1 \geq 2 \wedge s_2 \geq 0 \wedge s_3 \geq 1$

Tableaux-Notation

$$\begin{bmatrix} 1 & 1 & -1 & 0 & 0 \\ 2 & -1 & 0 & -1 & 0 \\ -1 & 2 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$s_1 \geq 2 \wedge s_2 \geq 0 \wedge s_3 \geq 1$$

Tableaux-Notation

$$\left[\begin{array}{cc|ccc} 1 & 1 & -1 & 0 & 0 \\ 2 & -1 & 0 & -1 & 0 \\ -1 & 2 & 0 & 0 & -1 \end{array}\right] \begin{bmatrix} x_1 \\ x_2 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$s_1 \geq 2 \land s_2 \geq 0 \land s_3 \geq 1$$

Tableaux-Notation

$$
\begin{array}{ccccc}
x_1 & x_2 & s_1 & s_2 & s_3 \\
\end{array}
$$

$$
\left[\begin{array}{cc|ccc}
1 & 1 & -1 & 0 & 0 \\
2 & -1 & 0 & -1 & 0 \\
-1 & 2 & 0 & 0 & -1
\end{array}\right]
\begin{bmatrix} x_1 \\ x_2 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix}
=
\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}
$$

$$
s_1 \geq 2 \land s_2 \geq 0 \land s_3 \geq 1
$$

Tableaux-Notation

$$\begin{array}{ccccc} x_1 & x_2 & s_1 & s_2 & s_3 \end{array}$$

$$\left[\begin{array}{cc|ccc} 1 & 1 & -1 & 0 & 0 \\ 2 & -1 & 0 & -1 & 0 \\ -1 & 2 & 0 & 0 & -1 \end{array}\right] \begin{bmatrix} x_1 \\ x_2 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix}$$

$$s_1 \geq 2 \wedge s_2 \geq 0 \wedge s_3 \geq 1$$

Tableaux-Notation

$$
\begin{array}{ccccc}
x_1 & x_2 & s_1 & s_2 & s_3 \\
\end{array}
$$

$$
\left[\begin{array}{cc|ccc}
1 & 1 & -1 & 0 & 0 \\
2 & -1 & 0 & -1 & 0 \\
-1 & 2 & 0 & 0 & -1
\end{array}\right]
$$

$$s_1 \geq 2 \wedge s_2 \geq 0 \wedge s_3 \geq 1$$

Tableaux-Notation

$$\begin{array}{c} \begin{array}{ccccc} x_1 & x_2 & s_1 & s_2 & s_3 \end{array} \\ \left[\begin{array}{cc|ccc} 1 & 1 & -1 & 0 & 0 \\ 2 & -1 & 0 & -1 & 0 \\ -1 & 2 & 0 & 0 & -1 \end{array}\right] \end{array}$$

$$s_1 \geq 2 \wedge s_2 \geq 0 \wedge s_3 \geq 1$$

Tableaux-Notation

$$\begin{array}{c|c}
 & \begin{array}{ccccc} x_1 & x_2 & s_1 & s_2 & s_3 \end{array} \\
\begin{array}{c} s_1 \\ s_2 \\ s_3 \end{array} & \left[\begin{array}{cc|ccc} 1 & 1 & -1 & 0 & 0 \\ 2 & -1 & 0 & -1 & 0 \\ -1 & 2 & 0 & 0 & -1 \end{array}\right]
\end{array}$$

$$s_1 \geq 2 \wedge s_2 \geq 0 \wedge s_3 \geq 1$$

Tableaux-Notation

$$
\begin{array}{cc}
 & \begin{array}{cc|ccc} x_1 & x_2 & & & \end{array} \\
\begin{array}{c} s_1 \\ s_2 \\ s_3 \end{array} & \left[\begin{array}{cc|ccc} 1 & 1 & -1 & 0 & 0 \\ 2 & -1 & 0 & -1 & 0 \\ -1 & 2 & 0 & 0 & -1 \end{array}\right]
\end{array}
$$

$$s_1 \geq 2 \land s_2 \geq 0 \land s_3 \geq 1$$

Tableaux-Notation

$$\begin{array}{c} \\ s_1 \\ s_2 \\ s_3 \end{array} \begin{array}{c} \begin{array}{cc} x_1 & x_2 \end{array} \\ \begin{bmatrix} 1 & 1 \\ 2 & -1 \\ -1 & 2 \end{bmatrix} \end{array}$$

$$s_1 \geq 2 \land s_2 \geq 0 \land s_3 \geq 1$$

Tableaux-Notation

$$\boxed{x_1 \quad x_2}$$

Nonbasic variables $\mathscr{N}$

$$\boxed{\begin{matrix} s_1 \\ s_2 \\ s_3 \end{matrix}}$$

Basic variables $\mathscr{B}$

$$s_1 \geq 2 \wedge s_2 \geq 0 \wedge s_3 \geq 1$$

Tableaux-Notation

$$\begin{array}{c c} & \begin{array}{cc} x_1 & x_2 \end{array} \\ \begin{array}{c} s_1 \\ s_2 \\ s_3 \end{array} & \begin{bmatrix} 1 & 1 \\ 2 & -1 \\ -1 & 2 \end{bmatrix} \end{array}$$

Nonbasic variables $\mathcal{N}$

Basic variables $\mathcal{B}$

$$s_1 \geq 2 \wedge s_2 \geq 0 \wedge s_3 \geq 1$$

Tableaux-Notation

$$\begin{array}{c c} & \begin{array}{cc} x_1 & x_2 \end{array} \\ \begin{array}{c} s_1 \\ s_2 \\ s_3 \end{array} & \begin{bmatrix} 1 & 1 \\ 2 & -1 \\ -1 & 2 \end{bmatrix} \end{array}$$

Nonbasic variables $\mathscr{N}$

Basic variables $\mathscr{B}$

$$s_1 \geq 2 \wedge s_2 \geq 0 \wedge s_3 \geq 1$$

The tableaux represents the following system of linear equations:

$$\bigwedge_{x_i \in \mathscr{B}} \left(x_i = \sum_{x_j \in \mathscr{N}} a_{ij} x_j \right)$$

We call them the **Tableaux-Equations**

The Simplex Algorithm

- Search for a satisfying assignment
- Series of pivot operations guide towards that assignment
- Pivots preserve inequalities

The Simplex Datastructures

The Simplex Datastructures

- The bounds on the variables remain constant

$$\forall x \in \mathscr{B} \cup \mathscr{N} : l_i \leq x_i \leq u_i$$

The Simplex Datastructures

- The bounds on the variables remain constant

$$\forall x \in \mathscr{B} \cup \mathscr{N} : l_i \leq x_i \leq u_i$$

Initially, only bounds on basic variables
but we'll make basic variables nonbasic

The Simplex Datastructures

- The bounds on the variables remain constant

$$\forall x \in \mathcal{B} \cup \mathcal{N} : l_i \leq x_i \leq u_i$$

- Maintain a tableaux

Initially, only bounds on basic variables
but we'll make basic variables nonbasic

- Maintain an assignment $\alpha : x \to \mathbb{Q}$

Simplex Datastructures

Initialization

$$s_1 \geq 2 \wedge s_2 \geq 0 \wedge s_3 \geq 1$$

Simplex Datastructures

Initialization

- The bounds on the variables remain constant

$$\forall x \in \mathscr{B} \cup \mathscr{N} : l_i \leq x_i \leq u_i$$

Simplex Datastructures

Initialization

$$\begin{array}{c c} & \begin{array}{cc} x_1 & x_2 \end{array} \\ \begin{array}{c} s_1 \\ s_2 \\ s_3 \end{array} & \begin{bmatrix} 1 & 1 \\ 2 & -1 \\ -1 & 2 \end{bmatrix} \end{array}$$

$$s_1 \geq 2 \land s_2 \geq 0 \land s_3 \geq 1$$

- The bounds on the variables remain constant

$$\forall x \in \mathcal{B} \cup \mathcal{N} : l_i \leq x_i \leq u_i$$

- Construct a tableaux as outlined before:
  - The basic variables are the additional variables
  - The nonbasic variables are the problem variables

Simplex Datastructures
Initialization

$$\begin{array}{c|cc} & x_1 & x_2 \\ s_1 & & \\ s_2 & & \\ s_3 & & \end{array} \quad \begin{array}{cc} & \\ \end{array}$$

$$\begin{array}{cc} & \begin{array}{cc} x_1 & x_2 \end{array} \\ \begin{array}{c} s_1 \\ s_2 \\ s_3 \end{array} & \begin{bmatrix} 1 & 1 \\ 2 & -1 \\ -1 & 2 \end{bmatrix} \end{array}$$

$$s_1 \geq 2 \wedge s_2 \geq 0 \wedge s_3 \geq 1$$

$$\alpha = \begin{pmatrix} x_1 & x_2 & s_1 & s_2 & s_3 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

- The bounds on the variables remain constant

$$\forall x \in \mathscr{B} \cup \mathscr{N} : l_i \leq x_i \leq u_i$$

- Construct a tableaux as outlined before:
    - The basic variables are the additional variables
    - The nonbasic variables are the problem variables
- Construct an assignment $\alpha \colon x \to \mathbb{Q}$ with $\alpha(x) = 0$

The Simplex Invariants

- Simplex maintains two invariants

1. $A\vec{x} = \vec{0}$ (i.e., the tableaux equalities do hold)
2. For each nonbasic variable $x_j \in \mathcal{N}$, it holds that $l_j \leq \alpha(x_j) \leq u_j$

The Simplex Invariants

- Simplex maintains two invariants

1. $A\vec{x} = \vec{0}$ (i.e., the tableaux equalities do hold)

2. For each nonbasic variable $x_j \in \mathcal{N}$, it holds that $l_j \leq \alpha(x_j) \leq u_j$

The bounds on the basic variables may be violated

The Simplex Invariants

- Simplex maintains two invariants

1. $A\vec{x} = \vec{0}$ (i.e., the tableaux equalities do hold)

2. For each nonbasic variable $x_j \in \mathcal{N}$, it holds that $l_j \leq \alpha(x_j) \leq u_j$

The bounds on the basic variables may be violated

Initially, simplex satisfies its invariants!

The Simplex Invariants

- Simplex maintains two invariants

1. $A\vec{x} = \vec{0}$ (i.e., the tableaux equalities do hold)
2. For each nonbasic variable $x_j \in \mathcal{N}$, it holds that $l_j \leq \alpha(x_j) \leq u_j$

The bounds on the basic variables may be violated

If simplex finds an assignment α that respects the bounds on the basic variables, α is a satisfying solution to the LP

Simplex

With a violated bound

$$\begin{array}{cc} & \begin{array}{cc} x_1 & x_2 \end{array} \\ \begin{array}{c} s_1 \\ s_2 \\ s_3 \end{array} & \begin{bmatrix} 1 & 1 \\ 2 & -1 \\ -1 & 2 \end{bmatrix} \end{array}$$

$$s_1 \geq 2 \land s_2 \geq 0 \land s_3 \geq 1$$

$$\alpha = \begin{pmatrix} x_1 & x_2 & s_1 & s_2 & s_3 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Simplex

With a violated bound

$$\begin{array}{c|cc} & x_1 & x_2 \\ \hline s_1 & & \\ s_2 & & \\ s_3 & & \end{array}\quad\begin{array}{c} \\ \begin{bmatrix} 1 & 1 \\ 2 & -1 \\ -1 & 2 \end{bmatrix}\end{array}$$

$$s_1 \geq 2 \wedge s_2 \geq 0 \wedge s_3 \geq 1$$

$$\alpha = \begin{pmatrix} x_1 & x_2 & s_1 & s_2 & s_3 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

- Let x_i violate it's lower bound (the upper bound case is symmetrical)

Simplex

With a violated bound

$$\begin{array}{c} \\ s_1 \\ s_2 \\ s_3 \end{array} \begin{array}{c} \begin{array}{cc} x_1 & x_2 \end{array} \\ \begin{bmatrix} 1 & 1 \\ 2 & -1 \\ -1 & 2 \end{bmatrix} \end{array}$$

$$s_1 \geq 2 \wedge s_2 \geq 0 \wedge s_3 \geq 1$$

$$\alpha = \begin{pmatrix} x_1 & x_2 & s_1 & s_2 & s_3 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

- Let x_i violate it's lower bound (the upper bound case is symmetrical)
- Thus, $\alpha(x_i) < l_i$

Simplex

With a violated bound

$$\begin{array}{c c} & \begin{array}{cc} x_1 & x_2 \end{array} \\ \begin{array}{c} s_1 \\ s_2 \\ s_3 \end{array} & \begin{bmatrix} 1 & 1 \\ 2 & -1 \\ -1 & 2 \end{bmatrix} \end{array}$$

$$s_1 \geq 2 \wedge s_2 \geq 0 \wedge s_3 \geq 1$$

$$\alpha = \begin{pmatrix} x_1 & x_2 & s_1 & s_2 & s_3 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

- Let x_i violate it's lower bound (the upper bound case is symmetrical)
- Thus, $\alpha(x_i) < l_i$
- Recall that x_i must be basic (why?)

Simplex

With a violated bound

$$\begin{array}{c} \\ s_1 \\ s_2 \\ s_3 \end{array} \begin{array}{c} \begin{array}{cc} x_1 & x_2 \end{array} \\ \begin{bmatrix} 1 & 1 \\ 2 & -1 \\ -1 & 2 \end{bmatrix} \end{array}$$

$$s_1 \geq 2 \wedge s_2 \geq 0 \wedge s_3 \geq 1$$

$$\alpha = \begin{pmatrix} x_1 & x_2 & s_1 & s_2 & s_3 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

- Let x_i violate it's lower bound (the upper bound case is symmetrical)
- Thus, $\alpha(x_i) < l_i$
- Recall that x_i must be basic (why?)
- Thus, use the tableaux equality $x_i = \sum_{x_j \in \mathcal{N}} a_{ij} x_j$ is satisfied by $\alpha(x_i) = \sum a_{ij} \cdot \alpha(x_j)$

Simplex

With a violated bound

$$\begin{array}{c} \\ s_1 \\ s_2 \\ s_3 \end{array} \begin{array}{cc} x_1 & x_2 \\ \multicolumn{2}{c}{\begin{bmatrix} 1 & 1 \\ 2 & -1 \\ -1 & 2 \end{bmatrix}} \end{array}$$

$$s_1 \geq 2 \land s_2 \geq 0 \land s_3 \geq 1$$

$$\alpha = \begin{pmatrix} x_1 & x_2 & s_1 & s_2 & s_3 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

- Let x_i violate it's lower bound (the upper bound case is symmetrical)
- Thus, $\alpha(x_i) < l_i$
- Recall that x_i must be basic (why?)
- Thus, use the tableaux equality $x_i = \sum_{x_j \in \mathcal{N}} a_{ij} x_j$ is satisfied by $\alpha(x_i) = \sum a_{ij} \cdot \alpha(x_j)$
- To increase $\alpha(x_i)$ we must either:

Simplex

With a violated bound

$$\begin{array}{c c} & \begin{array}{cc} x_1 & x_2 \end{array} \\ \begin{array}{c} s_1 \\ s_2 \\ s_3 \end{array} & \begin{bmatrix} 1 & 1 \\ 2 & -1 \\ -1 & 2 \end{bmatrix} \end{array}$$

$$s_1 \geq 2 \wedge s_2 \geq 0 \wedge s_3 \geq 1$$

$$\alpha = \begin{pmatrix} x_1 & x_2 & s_1 & s_2 & s_3 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

- Let x_i violate it's lower bound (the upper bound case is symmetrical)
- Thus, $\alpha(x_i) < l_i$
- Recall that x_i must be basic (why?)
- Thus, use the tableaux equality $x_i = \sum_{x_j \in \mathcal{N}} a_{ij} x_j$ is satisfied by $\alpha(x_i) = \sum a_{ij} \cdot \alpha(x_j)$
- To increase $\alpha(x_i)$ we must either:
    - increase some x_j with $a_{i,j} > 0$ and $\alpha(x_j) < u_j$

Simplex
With a violated bound

$$\begin{array}{c|cc} & x_1 & x_2 \\ \hline s_1 & 1 & 1 \\ s_2 & 2 & -1 \\ s_3 & -1 & 2 \end{array}$$

$$s_1 \geq 2 \land s_2 \geq 0 \land s_3 \geq 1$$

$$\alpha = \begin{pmatrix} x_1 & x_2 & s_1 & s_2 & s_3 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

- Let x_i violate it's lower bound (the upper bound case is symmetrical)
- Thus, $\alpha(x_i) < l_i$
- Recall that x_i must be basic (why?)
- Thus, use the tableaux equality $x_i = \sum_{x_j \in \mathcal{N}} a_{ij} x_j$ is satisfied by $\alpha(x_i) = \sum a_{ij} \cdot \alpha(x_j)$
- To increase $\alpha(x_i)$ we must either:
  - increase some x_j with $a_{i,j} > 0$ and $\alpha(x_j) < u_j$
  - decrease some x_j with $a_{i,j} < 0$ and $\alpha(x_j) > l_j$

Simplex

With a violated bound

$$\begin{array}{c|cc} & x_1 & x_2 \\ \hline s_1 & 1 & 1 \\ s_2 & 2 & -1 \\ s_3 & -1 & 2 \end{array}$$

$$s_1 \geq 2 \wedge s_2 \geq 0 \wedge s_3 \geq 1$$

$$\alpha = \begin{pmatrix} x_1 & x_2 & s_1 & s_2 & s_3 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

- Let x_i violate it's lower bound (the upper bound case is symmetrical)
- Thus, $\alpha(x_i) < l_i$
- Recall that x_i must be basic (why?)
- Thus, use the tableaux equality $x_i = \sum_{x_j \in \mathcal{N}} a_{ij} x_j$ is satisfied by $\alpha(x_i) = \sum a_{ij} \cdot \alpha(x_j)$
- To increase $\alpha(x_i)$ we must either:
  - increase some x_j with $a_{i,j} > 0$ and $\alpha(x_j) < u_j$
  - decrease some x_j with $a_{i,j} < 0$ and $\alpha(x_j) > l_j$

Such x_j is **suitable**

Simplex

Update with a violated bound

Simplex

Update with a violated bound

- Thus, $\alpha(x_i) < l_i$ and $x_i = \sum_{x_j \in \mathcal{N}} a_{ij} x_j$

Simplex

Update with a violated bound

- Thus, $\alpha(x_i) < l_i$ and $x_i = \sum_{x_j \in \mathcal{N}} a_{ij} x_j$
- To increase x_i we increase some x_j with $a_{i,j} > 0$ and $\alpha(x_j) < u_j$ (decreasing is symmetric)

Simplex

Update with a violated bound

- Thus, $\alpha(x_i) < l_i$ and $x_i = \sum_{x_j \in \mathcal{N}} a_{ij} x_j$
- To increase x_i we increase some x_j with $a_{i,j} > 0$ and $\alpha(x_j) < u_j$ (decreasing is symmetric)
- How much: Just enough to ensure the bound

Simplex

Update with a violated bound

- Thus, $\alpha(x_i) < l_i$ and $x_i = \sum_{x_j \in \mathcal{N}} a_{ij} x_j$

- To increase x_i we increase some x_j with $a_{i,j} > 0$ and $\alpha(x_j) < u_j$ (decreasing is symmetric)

- How much: Just enough to ensure the bound

- We increase $\alpha(x_j)$ by $\theta = \dfrac{l_i - \alpha(x_i)}{a_{ij}}$

Simplex

Update with a violated bound

- Thus, $\alpha(x_i) < l_i$ and $x_i = \sum_{x_j \in \mathcal{N}} a_{ij} x_j$
- To increase x_i we increase some x_j with $a_{i,j} > 0$ and $\alpha(x_j) < u_j$ (decreasing is symmetric)
- How much: Just enough to ensure the bound

x_i will now satisfy the bound

- We increase $\alpha(x_j)$ by $\theta = \dfrac{l_i - \alpha(x_i)}{a_{ij}}$

Simplex

Update with a violated bound

- Thus, $\alpha(x_i) < l_i$ and $x_i = \sum_{x_j \in \mathcal{N}} a_{ij} x_j$
- To increase x_i we increase some x_j with $a_{i,j} > 0$ and $\alpha(x_j) < u_j$ (decreasing is symmetric)
- How much: Just enough to ensure the bound
- We increase $\alpha(x_j)$ by $\theta = \dfrac{l_i - \alpha(x_i)}{a_{ij}}$

x_i will now satisfy the bound

x_j may now violate the bound

Pivot Operation

Intuition to preserve invariants

Pivot Operation

Intuition to preserve invariants

- If nonbasic variable x_j now violates its bound, make it basic

Pivot Operation

Intuition to preserve invariants

- If nonbasic variable x_j now violates its bound, make it basic
- Every basic variable is a sum of nonbasic variables…

Pivot Operation

Intuition to preserve invariants

- If nonbasic variable x_j now violates its bound, make it basic
- Every basic variable is a sum of nonbasic variables…
- So basic variable x_i must become nonbasic— ok: it satisfies its bound

Pivot Operation

Intuition to preserve invariants

- If nonbasic variable x_j now violates its bound, make it basic
- Every basic variable is a sum of nonbasic variables…
- So basic variable x_i must become nonbasic— ok: it satisfies its bound
- This leads to the pivot operation:

Pivot Operation

Intuition to preserve invariants

- If nonbasic variable x_j now violates its bound, make it basic
- Every basic variable is a sum of nonbasic variables…
- So basic variable x_i must become nonbasic— ok: it satisfies its bound
- This leads to the pivot operation:
- Rewrite $x_i = a_{ij}x_j + \sum_{j \neq k, k \in \mathcal{N}} a_{ik}x_k$ to $x_j = \dfrac{x_i - \sum \dots}{a_{ij}}$

Pivot Operation

Intuition to preserve invariants

- If nonbasic variable x_j now violates its bound, make it basic
- Every basic variable is a sum of nonbasic variables…
- So basic variable x_i must become nonbasic— ok: it satisfies its bound
- This leads to the pivot operation:
  - Rewrite $x_i = a_{ij}x_j + \sum_{j \neq k, k \in \mathcal{N}} a_{ik}x_k$ to $x_j = \dfrac{x_i - \sum \dots}{a_{ij}}$
  - Use this to substitute all occurrences of x_j in the tableaux

Pivot

$$\begin{array}{c c} & \begin{array}{cc} x_1 & x_2 \end{array} \\ \begin{array}{c} s_1 \\ s_2 \\ s_3 \end{array} & \begin{bmatrix} 1 & 1 \\ 2 & -1 \\ -1 & 2 \end{bmatrix} \end{array}$$

$$s_1 \geq 2 \wedge s_2 \geq 0 \wedge s_3 \geq 1$$

$$\alpha = \begin{pmatrix} x_1 & x_2 & s_1 & s_2 & s_3 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Pivot

$$\begin{array}{c} \\ s_1 \\ s_2 \\ s_3 \end{array} \begin{array}{c} \begin{array}{cc} x_1 & x_2 \end{array} \\ \begin{bmatrix} 1 & 1 \\ 2 & -1 \\ -1 & 2 \end{bmatrix} \end{array}$$

$$s_1 \geq 2 \wedge s_2 \geq 0 \wedge s_3 \geq 1$$

$$\alpha = \begin{pmatrix} x_1 & x_2 & s_1 & s_2 & s_3 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

- s_1 violates its bound, x_1 is suitable.

Pivot

$$\begin{array}{c c} & \begin{array}{cc} x_1 & x_2 \end{array} \\ \begin{array}{c} s_1 \\ s_2 \\ s_3 \end{array} & \begin{bmatrix} 1 & 1 \\ 2 & -1 \\ -1 & 2 \end{bmatrix} \end{array}$$

$$s_1 \geq 2 \wedge s_2 \geq 0 \wedge s_3 \geq 1$$

$$\alpha = \begin{pmatrix} x_1 & x_2 & s_1 & s_2 & s_3 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

- s_1 violates its bound, x_1 is suitable.
- Rewrite $s_1 = x_1 + x_2$ into $x_1 = s_1 - x_2$, use as a substitution rule

Pivot

$$\begin{array}{cc} & \begin{array}{cc} x_1 & x_2 \end{array} \\ \begin{array}{c} s_1 \\ s_2 \\ s_3 \end{array} & \begin{bmatrix} 1 & 1 \\ 2 & -1 \\ -1 & 2 \end{bmatrix} \end{array}$$

$$s_1 \geq 2 \wedge s_2 \geq 0 \wedge s_3 \geq 1$$

$$\alpha = \begin{pmatrix} x_1 & x_2 & s_1 & s_2 & s_3 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

- s_1 violates its bound, x_1 is suitable.
- Rewrite $s_1 = x_1 + x_2$ into $x_1 = s_1 - x_2$, use as a substitution rule
- $s_2 = 2x_1 - 1x_2$ becomes $s_2 = 2s_1 - 3x_2$ and
  $s_3 = -1x_1 + 2x_2$ becomes $s_3 = -s_1 + 3x_2$

Pivot

$$\begin{array}{c|cc} & x_1 & x_2 \\ \hline s_1 & 1 & 1 \\ s_2 & 2 & -1 \\ s_3 & -1 & 2 \end{array} \quad \text{i.e.} \quad \begin{array}{c} \\ s_1 \\ s_2 \\ s_3 \end{array} \begin{array}{c} \begin{array}{cc} x_1 & x_2 \end{array} \\ \begin{bmatrix} 1 & 1 \\ 2 & -1 \\ -1 & 2 \end{bmatrix} \end{array}$$

$$s_1 \geq 2 \wedge s_2 \geq 0 \wedge s_3 \geq 1$$

$$\alpha = \begin{pmatrix} x_1 & x_2 & s_1 & s_2 & s_3 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

- s_1 violates its bound, x_1 is suitable.
- Rewrite $s_1 = x_1 + x_2$ into $x_1 = s_1 - x_2$, use as a substitution rule
- $s_2 = 2x_1 - 1x_2$ becomes $s_2 = 2s_1 - 3x_2$ and
  $s_3 = -1x_1 + 2x_2$ becomes $s_3 = -s_1 + 3x_2$

$$\begin{array}{c} \\ s_1 \\ s_2 \\ s_3 \end{array} \begin{array}{c} \begin{array}{cc} x_1 & x_2 \end{array} \\ \begin{bmatrix} 1 & 1 \\ 2 & -1 \\ -1 & 2 \end{bmatrix} \end{array} \quad \text{becomes} \quad \begin{array}{c} \\ x_1 \\ s_2 \\ s_3 \end{array} \begin{array}{c} \begin{array}{cc} s_1 & x_2 \end{array} \\ \begin{bmatrix} 1 & -1 \\ 2 & -3 \\ -1 & 3 \end{bmatrix} \end{array}$$

Update assignment

By example

$$\begin{array}{c|cc} & x_1 & x_2 \\ \hline s_1 & 1 & 1 \\ s_2 & 2 & -1 \\ s_3 & -1 & 2 \end{array}$$

$$s_1 \geq 2 \wedge s_2 \geq 0 \wedge s_3 \geq 1$$

$$\alpha = \begin{pmatrix} x_1 & x_2 & s_1 & s_2 & s_3 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\begin{array}{c|cc} & x_1 & x_2 \\ \hline s_1 & 1 & 1 \\ s_2 & 2 & -1 \\ s_3 & -1 & 2 \end{array}$$

becomes

$$\begin{array}{c|cc} & s_1 & x_2 \\ \hline x_1 & 1 & -1 \\ s_2 & 2 & -3 \\ s_3 & -1 & 3 \end{array}$$

Update assignment

By example

$$\begin{array}{c c} & \begin{array}{cc} x_1 & x_2 \end{array} \\ \begin{array}{c} s_1 \\ s_2 \\ s_3 \end{array} & \begin{bmatrix} 1 & 1 \\ 2 & -1 \\ -1 & 2 \end{bmatrix} \end{array}$$

$$s_1 \geq 2 \wedge s_2 \geq 0 \wedge s_3 \geq 1$$

$$\alpha = \begin{pmatrix} x_1 & x_2 & s_1 & s_2 & s_3 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

- s_1 violates its bound, x_1 is suitable.

$$\alpha = \begin{pmatrix} x_1 & x_2 & s_1 & s_2 & s_3 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\begin{array}{c c} & \begin{array}{cc} x_1 & x_2 \end{array} \\ \begin{array}{c} s_1 \\ s_2 \\ s_3 \end{array} & \begin{bmatrix} 1 & 1 \\ 2 & -1 \\ -1 & 2 \end{bmatrix} \end{array} \quad \text{becomes} \quad \begin{array}{c c} & \begin{array}{cc} s_1 & x_2 \end{array} \\ \begin{array}{c} x_1 \\ s_2 \\ s_3 \end{array} & \begin{bmatrix} 1 & -1 \\ 2 & -3 \\ -1 & 3 \end{bmatrix} \end{array}$$

Update assignment

By example

$$\begin{array}{c c} & \begin{array}{cc} x_1 & x_2 \end{array} \\ \begin{array}{c} s_1 \\ s_2 \\ s_3 \end{array} & \begin{bmatrix} 1 & 1 \\ 2 & -1 \\ -1 & 2 \end{bmatrix} \end{array}$$

$$s_1 \geq 2 \land s_2 \geq 0 \land s_3 \geq 1$$

$$\alpha = \begin{pmatrix} x_1 & x_2 & s_1 & s_2 & s_3 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

- s_1 violates its bound, x_1 is suitable.
- We must increase s_1 by two, we do this by increasing x_1 by two

$$\alpha = \begin{pmatrix} x_1 & x_2 & s_1 & s_2 & s_3 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\alpha = \begin{pmatrix} x_1 & x_2 & s_1 & s_2 & s_3 \\ 2 & 0 & 2 & 0 & 0 \end{pmatrix}$$

$$\begin{array}{c c} & \begin{array}{cc} x_1 & x_2 \end{array} \\ \begin{array}{c} s_1 \\ s_2 \\ s_3 \end{array} & \begin{bmatrix} 1 & 1 \\ 2 & -1 \\ -1 & 2 \end{bmatrix} \end{array} \quad \text{becomes} \quad \begin{array}{c c} & \begin{array}{cc} s_1 & x_2 \end{array} \\ \begin{array}{c} x_1 \\ s_2 \\ s_3 \end{array} & \begin{bmatrix} 1 & -1 \\ 2 & -3 \\ -1 & 3 \end{bmatrix} \end{array}$$

Update assignment

By example

$$\begin{array}{cc} & \begin{array}{cc} x_1 & x_2 \end{array} \\ \begin{array}{c} s_1 \\ s_2 \\ s_3 \end{array} & \begin{bmatrix} 1 & 1 \\ 2 & -1 \\ -1 & 2 \end{bmatrix} \end{array}$$

$$s_1 \geq 2 \wedge s_2 \geq 0 \wedge s_3 \geq 1$$

$$\alpha = \begin{pmatrix} x_1 & x_2 & s_1 & s_2 & s_3 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

- s_1 violates its bound, x_1 is suitable.
- We must increase s_1 by two, we do this by increasing x_1 by two
- And update all other assignments accordingly

$$\alpha = \begin{pmatrix} x_1 & x_2 & s_1 & s_2 & s_3 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\alpha = \begin{pmatrix} x_1 & x_2 & s_1 & s_2 & s_3 \\ 2 & 0 & 2 & 0 & 0 \end{pmatrix}$$

$$\alpha = \begin{pmatrix} x_1 & x_2 & s_1 & s_2 & s_3 \\ 2 & 0 & 2 & 4 & -2 \end{pmatrix}$$

$$\begin{array}{cc} & \begin{array}{cc} x_1 & x_2 \end{array} \\ \begin{array}{c} s_1 \\ s_2 \\ s_3 \end{array} & \begin{bmatrix} 1 & 1 \\ 2 & -1 \\ -1 & 2 \end{bmatrix} \end{array} \quad \text{becomes} \quad \begin{array}{cc} & \begin{array}{cc} s_1 & x_2 \end{array} \\ \begin{array}{c} x_1 \\ s_2 \\ s_3 \end{array} & \begin{bmatrix} 1 & -1 \\ 2 & -3 \\ -1 & 3 \end{bmatrix} \end{array}$$

Pivot

By example

$$\begin{array}{c|cc} & s_1 & x_2 \\ \hline x_1 & 1 & -1 \\ s_2 & 2 & -3 \\ s_3 & -1 & 3 \end{array}$$

$$s_1 \geq 2 \wedge s_2 \geq 0 \wedge s_3 \geq 1$$

$$\alpha = \begin{pmatrix} x_1 & x_2 & s_1 & s_2 & s_3 \\ 2 & 0 & 2 & 4 & -2 \end{pmatrix}$$

- s_3 violates its bound, x_2 is suitable.
- Rewrite $s_3 = -s_1 + 3x_2$ into $x_2 = \dfrac{s_3 + s_1}{3}$, use as a substitution rule

$$\begin{array}{c|cc} & s_1 & x_2 \\ \hline x_1 & 1 & -1 \\ s_2 & 2 & -3 \\ s_3 & -1 & 3 \end{array} \quad \text{becomes} \quad \begin{array}{c|cc} & s_1 & s_3 \\ \hline x_1 & \frac{2}{3} & \frac{-1}{3} \\ s_2 & 1 & -1 \\ x_2 & \frac{1}{3} & \frac{1}{3} \end{array}$$

About Pivots
(Again)

About Pivots
(Again)

- Pivots are equivalence transformations, describe the same system

About Pivots

(Again)

- Pivots are equivalence transformations, describe the same system
- They just steer us towards updates that make the system satisfiable

About Pivots

(Again)

- Pivots are equivalence transformations, describe the same system
- They just steer us towards updates that make the system satisfiable
- We do not cycle…

About Pivots

(Again)

- Pivots are equivalence transformations, describe the same system
- They just steer us towards updates that make the system satisfiable
- We do not cycle…
  - Assuming we pick variables satisfying the bounds and suitable variables from a global order, we are guaranteed to terminate (Bland’s Rule).

Generalized Simplex

Generalized Simplex

1. Bring system in standard form, initialize

Generalized Simplex

1. Bring system in standard form, initialize
2. Fix a order on all variables

Generalized Simplex

1. Bring system in standard form, initialize
2. Fix a order on all variables
3. While there is a basic variable violating its bounds

Generalized Simplex

1. Bring system in standard form, initialize
2. Fix a order on all variables
3. While there is a basic variable violating its bounds
   A. Find the smallest suitable nonbasic variable or **return** UNSAT

Generalized Simplex

1. Bring system in standard form, initialize
2. Fix a order on all variables
3. While there is a basic variable violating its bounds
    A. Find the smallest suitable nonbasic variable or **return** UNSAT
    B. Pivot and update assignment

Generalized Simplex

1. Bring system in standard form, initialize
2. Fix a order on all variables
3. While there is a basic variable violating its bounds
   A. Find the smallest suitable nonbasic variable or **return** UNSAT
   B. Pivot and update assignment
4. **return** SAT

Beyond Generalized Simplex

- Use some heuristic to pick variables, if we do not terminate, fall back to Bland's Rule.
- Select variables to minimize sum of violations / slack
- Handle strict inequalities using an infinitesimal small constant
- Lots of preprocessing

Summary

- Linear programming & QF_LRA
- Generalized Simplex
- Incrementality & Backtracking for Generalized Simplex
- Branch & Bound for QF_LIA
- Difference logic

Iterative Simplex

Focus on essential aspects

- How to generate unfeasible subsets?
- How to ensure incrementality?

Infeasible Subsets

Infeasible Subsets

- Goal: Find a (sub)set of constraints that are together unsatisfiable.

Infeasible Subsets

- Goal: Find a (sub)set of constraints that are together unsatisfiable.
- Find unsatisfiability -> some basic variable x_i violates its bound

Infeasible Subsets

- Goal: Find a (sub)set of constraints that are together unsatisfiable.
- Find unsatisfiability -> some basic variable x_i violates its bound
- W.l.o.g., assume it exceeds upper bound

Infeasible Subsets

- Goal: Find a (sub)set of constraints that are together unsatisfiable.
- Find unsatisfiability -> some basic variable x_i violates its bound
- W.l.o.g., assume it exceeds upper bound
- Let $x_i = \sum_{x_j \in \mathcal{N}} a_{ij} x_j$ thus $u_i < \alpha(x_i) = \sum_{x_j \in \mathcal{N}} a_{ij} \cdot \alpha(x_j)$

Infeasible Subsets

- Goal: Find a (sub)set of constraints that are together unsatisfiable.
- Find unsatisfiability -> some basic variable x_i violates its bound
- W.l.o.g., assume it exceeds upper bound
- Let $x_i = \sum_{x_j \in \mathcal{N}} a_{ij} x_j$ thus $u_i < \alpha(x_i) = \sum_{x_j \in \mathcal{N}} a_{ij} \cdot \alpha(x_j)$
- UNSAT, so no x_j suitable. Bounds for x_i and all x_j with $a_{ij} \neq 0$ are explanation

Infeasible Subsets

- Goal: Find a (sub)set of constraints that are together unsatisfiable.
- Find unsatisfiability -> some basic variable x_i violates its bound
- W.l.o.g., assume it exceeds upper bound
- Let $x_i = \sum_{x_j \in \mathcal{N}} a_{ij} x_j$ thus $u_i < \alpha(x_i) = \sum_{x_j \in \mathcal{N}} a_{ij} \cdot \alpha(x_j)$
- UNSAT, so no x_j suitable. Bounds for x_i and all x_j with $a_{ij} \neq 0$ are explanation
- Only additional variables have bounds -> these encode constraints.

Incrementality and Backtracking

Incrementality and Backtracking

- Assume an LP with constraints C has been checked and now you want to check $C \cup \{c\}$?

Incrementality and Backtracking

- Assume an LP with constraints C has been checked and now you want to check $C \cup \{c\}$?
- Just adding a row can be difficult as the nonbasic variables may not coincide with the original variables

Incrementality and Backtracking

- Assume an LP with constraints C has been checked and now you want to check $C \cup \{c\}$?
- Just adding a row can be difficult as the nonbasic variables may not coincide with the original variables
- (Assume that all constraints are part of an a-priori known system)

Incrementality and Backtracking

- Assume an LP with constraints C has been checked and now you want to check $C \cup \{c\}$?
- Just adding a row can be difficult as the nonbasic variables may not coincide with the original variables
- (Assume that all constraints are part of an a-priori known system)
- Create a tableaux with all constraints, but only enforce bounds for active constraints

Incrementality and Backtracking

- Assume an LP with constraints C has been checked and now you want to check $C \cup \{c\}$?
- Just adding a row can be difficult as the nonbasic variables may not coincide with the original variables
- (Assume that all constraints are part of an a-priori known system)
- Create a tableaux with all constraints, but only enforce bounds for active constraints
- Adding a constraint: Enforcing the bound.
  Removing a constraint: Restore assignment to last known satisfying assignment

DPLL(T)

Example

Initialize the Simplex tableau with all equalities but without any bounds.

$$
\begin{array}{lclcccl}
p_1 = 0 & \rightarrow & s_1 = & p_1 & & & s_1 = 0 \\
p_2 = 0 & \rightarrow & s_2 = & & p_2 & & s_2 = 0 \\
p_3 = 0 & \rightarrow & s_3 = & & & p_3 & s_3 = 0 \\
p_1 + p_2 + p_3 \geq 100 & \rightarrow & s_4 = & p_1 + & p_2 + & p_3 & s_4 \geq 100 \\
p_1 \geq 5 & \rightarrow & s_5 = & p_1 & & & s_5 \geq 5 \\
p_2 \geq 5 & \rightarrow & s_6 = & & p_2 & & s_6 \geq 5 \\
p_3 \geq 10 & \rightarrow & s_7 = & & & p_3 & s_7 \geq 10 \\
p_1 + 2p_2 + 5p_3 \leq 180 & \rightarrow & s_8 = & p_1 + & 2p_2 + & 5p_3 & s_8 \leq 180 \\
3p_1 + 2p_2 + p_3 \leq 300 & \rightarrow & s_9 = & 3p_1 + & 2p_2 + & p_3 & s_9 \leq 300
\end{array}
$$

	$p_1(0)$	$p_2(0)$	$p_3(0)$
$s_1(0)$	1	0	0
$s_2(0)$	0	1	0
$s_3(0)$	0	0	1
$s_4(0)$	1	1	1
$s_5(0)$	1	0	0
$s_6(0)$	0	1	0
$s_7(0)$	0	0	1
$s_8(0)$	1	2	5
$s_9(0)$	3	2	1

DPLL(T)

Example

$$(a_1 \vee a_2 \vee a_3) \wedge a_4 \wedge (a_5 \vee a_6) \wedge a_7 \wedge a_8 \wedge a_9$$

Assume a fixed variable order: $a_1, \ldots, a_9$
Assignment to decision variables: false

$DL0 : a_4 : 1, a_7 : 1, a_8 : 1, a_9 : 1$

DPLL(T)

Example

$$
\begin{array}{lclllll}
p_1 = 0 & \rightarrow & s_1 = & p_1 & & & s_1 = 0 \\
p_2 = 0 & \rightarrow & s_2 = & & p_2 & & s_2 = 0 \\
p_3 = 0 & \rightarrow & s_3 = & & & p_3 & s_3 = 0 \\
p_1 + p_2 + p_3 \geq 100 & \rightarrow & s_4 = & p_1 + & p_2 + & p_3 & s_4 \geq 100 \\
p_1 \geq 5 & \rightarrow & s_5 = & p_1 & & & s_5 \geq 5 \\
p_2 \geq 5 & \rightarrow & s_6 = & & p_2 & & s_6 \geq 5 \\
p_3 \geq 10 & \rightarrow & s_7 = & & & p_3 & s_7 \geq 10 \\
p_1 + 2p_2 + 5p_3 \leq 180 & \rightarrow & s_8 = & p_1 + & 2p_2 + & 5p_3 & s_8 \leq 180 \\
3p_1 + 2p_2 + p_3 \leq 300 & \rightarrow & s_9 = & 3p_1 + & 2p_2 + & p_3 & s_9 \leq 300
\end{array}
$$

Variable order: $s_1 < \ldots < s_9 < p_1 < p_2 < p_3$, the values of the variables are given in parentheses

	$p_1(0)$	$p_2(0)$	$p_3(0)$
$s_1(0)$	1	0	0
$s_2(0)$	0	1	0
$s_3(0)$	0	0	1
$s_4(0)$	1	1	1
$s_5(0)$	1	0	0
$s_6(0)$	0	1	0
$s_7(0)$	0	0	1
$s_8(0)$	1	2	5
$s_9(0)$	3	2	1

DPLL(T)

Example

$$
\begin{array}{lcllll}
p_1 = 0 & \rightarrow & s_1 = & p_1 & & & s_1 = 0 \\
p_2 = 0 & \rightarrow & s_2 = & & p_2 & & s_2 = 0 \\
p_3 = 0 & \rightarrow & s_3 = & & & p_3 & s_3 = 0 \\
p_1 + p_2 + p_3 \geq 100 & \rightarrow & s_4 = & p_1+ & p_2+ & p_3 & s_4 \geq 100 \\
p_1 \geq 5 & \rightarrow & s_5 = & p_1 & & & s_5 \geq 5 \\
p_2 \geq 5 & \rightarrow & s_6 = & & p_2 & & s_6 \geq 5 \\
p_3 \geq 10 & \rightarrow & s_7 = & & & p_3 & s_7 \geq 10 \\
p_1 + 2p_2 + 5p_3 \leq 180 & \rightarrow & s_8 = & p_1+ & 2p_2+ & 5p_3 & s_8 \leq 180 \\
3p_1 + 2p_2 + p_3 \leq 300 & \rightarrow & s_9 = & 3p_1+ & 2p_2+ & p_3 & s_9 \leq 300
\end{array}
$$

Variable order: $s_1 < \ldots < s_9 < p_1 < p_2 < p_3$, the values of the variables are given in parentheses

	$p_1(0)$	$p_2(0)$	$p_3(0)$		$s_4(100)$	$p_2(0)$	$p_3(0)$
$s_1(0)$	1	0	0	$s_1(100)$	1	-1	-1
$s_2(0)$	0	1	0	$s_2(0)$	0	1	0
$s_3(0)$	0	0	1	$s_3(0)$	0	0	1
$s_4(0)$	1	1	1	$p_1(100)$	1	-1	-1
$s_5(0)$	1	0	0	$s_5(100)$	1	-1	-1
$s_6(0)$	0	1	0	$s_6(0)$	0	1	0
$s_7(0)$	0	0	1	$s_7(0)$	0	0	1
$s_8(0)$	1	2	5	$s_8(100)$	1	1	4
$s_9(0)$	3	2	1	$s_9(300)$	3	-1	-2

DPLL(T)

Example

$$
\begin{array}{lclllll}
p_1 = 0 & \rightarrow & s_1 = & p_1 & & & s_1 = 0 \\
p_2 = 0 & \rightarrow & s_2 = & & p_2 & & s_2 = 0 \\
p_3 = 0 & \rightarrow & s_3 = & & & p_3 & s_3 = 0 \\
p_1 + p_2 + p_3 \geq 100 & \rightarrow & s_4 = & p_1 + & p_2 + & p_3 & s_4 \geq 100 \\
p_1 \geq 5 & \rightarrow & s_5 = & p_1 & & & s_5 \geq 5 \\
p_2 \geq 5 & \rightarrow & s_6 = & & p_2 & & s_6 \geq 5 \\
p_3 \geq 10 & \rightarrow & s_7 = & & & p_3 & s_7 \geq 10 \\
p_1 + 2p_2 + 5p_3 \leq 180 & \rightarrow & s_8 = & p_1 + & 2p_2 + & 5p_3 & s_8 \leq 180 \\
3p_1 + 2p_2 + p_3 \leq 300 & \rightarrow & s_9 = & 3p_1 + & 2p_2 + & p_3 & s_9 \leq 300
\end{array}
$$

Variable order: $s_1 < \ldots < s_9 < p_1 < p_2 < p_3$, the values of the variables are given in parentheses

	$p_1(0)$	$p_2(0)$	$p_3(0)$		$s_4(100)$	$p_2(0)$	$p_3(0)$		$s_4(100)$	$p_2(0)$	$s_7(10)$
$s_1(0)$	1	0	0	$s_1(100)$	1	-1	-1	$s_1(90)$	1	-1	-1
$s_2(0)$	0	1	0	$s_2(0)$	0	1	0	$s_2(0)$	0	1	0
$s_3(0)$	0	0	1	$s_3(0)$	0	0	1	$s_3(10)$	0	0	1
$s_4(0)$	1	1	1	$p_1(100)$	1	-1	-1	$p_1(90)$	1	-1	-1
$s_5(0)$	1	0	0	$s_5(100)$	1	-1	-1	$s_5(90)$	1	-1	-1
$s_6(0)$	0	1	0	$s_6(0)$	0	1	0	$s_6(0)$	0	1	0
$s_7(0)$	0	0	1	$s_7(0)$	0	0	1	$p_3(10)$	0	0	1
$s_8(0)$	1	2	5	$s_8(100)$	1	1	4	$s_8(140)$	1	1	4
$s_9(0)$	3	2	1	$s_9(300)$	3	-1	-2	$s_9(280)$	3	-1	-2

DPLL(T)

Example

$$
\begin{array}{lclllll}
p_1 = 0 & \rightarrow & s_1 = & p_1 & & & s_1 = 0 \\
p_2 = 0 & \rightarrow & s_2 = & & p_2 & & s_2 = 0 \\
p_3 = 0 & \rightarrow & s_3 = & & & p_3 & s_3 = 0 \\
p_1 + p_2 + p_3 \geq 100 & \rightarrow & s_4 = & p_1 + & p_2 + & p_3 & s_4 \geq 100 \\
p_1 \geq 5 & \rightarrow & s_5 = & p_1 & & & s_5 \geq 5 \\
p_2 \geq 5 & \rightarrow & s_6 = & & p_2 & & s_6 \geq 5 \\
p_3 \geq 10 & \rightarrow & s_7 = & & & p_3 & s_7 \geq 10 \\
p_1 + 2p_2 + 5p_3 \leq 180 & \rightarrow & s_8 = & p_1 + & 2p_2 + & 5p_3 & s_8 \leq 180 \\
3p_1 + 2p_2 + p_3 \leq 300 & \rightarrow & s_9 = & 3p_1 + & 2p_2 + & p_3 & s_9 \leq 300
\end{array}
$$

Variable order: $s_1 < \ldots < s_9 < p_1 < p_2 < p_3$, the values of the variables are given in parentheses

	$p_1(0)$	$p_2(0)$	$p_3(0)$		$s_4(100)$	$p_2(0)$	$p_3(0)$		$s_4(100)$	$p_2(0)$	$s_7(10)$
$s_1(0)$	1	0	0	$s_1(100)$	1	-1	-1	$s_1(90)$	1	-1	-1
$s_2(0)$	0	1	0	$s_2(0)$	0	1	0	$s_2(0)$	0	1	0
$s_3(0)$	0	0	1	$s_3(0)$	0	0	1	$s_3(10)$	0	0	1
$s_4(0)$	1	1	1	$p_1(100)$	1	-1	-1	$p_1(90)$	1	-1	-1
$s_5(0)$	1	0	0	$s_5(100)$	1	-1	-1	$s_5(90)$	1	-1	-1
$s_6(0)$	0	1	0	$s_6(0)$	0	1	0	$s_6(0)$	0	1	0
$s_7(0)$	0	0	1	$s_7(0)$	0	0	1	$p_3(10)$	0	0	1
$s_8(0)$	1	2	5	$s_8(100)$	1	1	4	$s_8(140)$	1	1	4
$s_9(0)$	3	2	1	$s_9(300)$	3	-1	-2	$s_9(280)$	3	-1	-2

Return partial SAT.

DPLL(T)

Example

$$(a_1 \vee a_2 \vee a_3) \wedge a_4 \wedge (a_5 \vee a_6) \wedge a_7 \wedge a_8 \wedge a_9$$

Assume a fixed variable order: $a_1, \ldots, a_9$

Assignment to decision variables: false

$DL0 : a_4 : 1, a_7 : 1, a_8 : 1, a_9 : 1$

$DL1 : a_1 : 0$

$DL2 : a_2 : 0, a_3 : 1$

DPLL(T)

Example

$p_1 = 0$	$\rightarrow$	$s_1 =$	p_1			$s_1 = 0$
$p_2 = 0$	$\rightarrow$	$s_2 =$		p_2		$s_2 = 0$
$p_3 = 0$	$\rightarrow$	$s_3 =$			p_3	$s_3 = 0$
$p_1 + p_2 + p_3 \geq 100$	$\rightarrow$	$s_4 =$	$p_1 +$	$p_2 +$	p_3	$s_4 \geq 100$
$p_1 \geq 5$	$\rightarrow$	$s_5 =$	p_1			$s_5 \geq 5$
$p_2 \geq 5$	$\rightarrow$	$s_6 =$		p_2		$s_6 \geq 5$
$p_3 \geq 10$	$\rightarrow$	$s_7 =$			p_3	$s_7 \geq 10$
$p_1 + 2p_2 + 5p_3 \leq 180$	$\rightarrow$	$s_8 =$	$p_1 +$	$2p_2 +$	$5p_3$	$s_8 \leq 180$
$3p_1 + 2p_2 + p_3 \leq 300$	$\rightarrow$	$s_9 =$	$3p_1 +$	$2p_2 +$	p_3	$s_9 \leq 300$

DPLL(T)

Example

$$\begin{array}{lcrrrrl}
p_1 = 0 & \rightarrow & s_1 = & p_1 & & & s_1 = 0 \\
p_2 = 0 & \rightarrow & s_2 = & & p_2 & & s_2 = 0 \\
p_3 = 0 & \rightarrow & s_3 = & & & p_3 & s_3 = 0 \\
p_1 + p_2 + p_3 \geq 100 & \rightarrow & s_4 = & p_1 + & p_2 + & p_3 & s_4 \geq 100 \\
p_1 \geq 5 & \rightarrow & s_5 = & p_1 & & & s_5 \geq 5 \\
p_2 \geq 5 & \rightarrow & s_6 = & & p_2 & & s_6 \geq 5 \\
p_3 \geq 10 & \rightarrow & s_7 = & & & p_3 & s_7 \geq 10 \\
p_1 + 2p_2 + 5p_3 \leq 180 & \rightarrow & s_8 = & p_1 + & 2p_2 + & 5p_3 & s_8 \leq 180 \\
3p_1 + 2p_2 + p_3 \leq 300 & \rightarrow & s_9 = & 3p_1 + & 2p_2 + & p_3 & s_9 \leq 300
\end{array}$$

	$s_4(100)$	$p_2(0)$	$s_7(10)$
$s_1(90)$	1	-1	-1
$s_2(0)$	0	1	0
$s_3(10)$	0	0	1
$p_1(90)$	1	-1	-1
$s_5(90)$	1	-1	-1
$s_6(0)$	0	1	0
$p_3(10)$	0	0	1
$s_8(140)$	1	1	4
$s_9(280)$	3	-1	-2

DPLL(T)

Example

$$
\begin{array}{lcl lll l}
p_1 = 0 & \to & s_1 = & p_1 & & & s_1 = 0 \\
p_2 = 0 & \to & s_2 = & & p_2 & & s_2 = 0 \\
p_3 = 0 & \to & s_3 = & & & p_3 & \boxed{s_3 = 0} \\
p_1 + p_2 + p_3 \geq 100 & \to & s_4 = & p_1 + & p_2 + & p_3 & s_4 \geq 100 \\
p_1 \geq 5 & \to & s_5 = & p_1 & & & s_5 \geq 5 \\
p_2 \geq 5 & \to & s_6 = & & p_2 & & s_6 \geq 5 \\
p_3 \geq 10 & \to & s_7 = & & & p_3 & s_7 \geq 10 \\
p_1 + 2p_2 + 5p_3 \leq 180 & \to & s_8 = & p_1 + & 2p_2 + & 5p_3 & s_8 \leq 180 \\
3p_1 + 2p_2 + p_3 \leq 300 & \to & s_9 = & 3p_1 + & 2p_2 + & p_3 & s_9 \leq 300
\end{array}
$$

	$s_4(100)$	$p_2(0)$	$s_7(10)$
$s_1(90)$	1	-1	-1
$s_2(0)$	0	1	0
$s_3(10)$	0	0	1
$p_1(90)$	1	-1	-1
$s_5(90)$	1	-1	-1
$s_6(0)$	0	1	0
$p_3(10)$	0	0	1
$s_8(140)$	1	1	4
$s_9(280)$	3	-1	-2

Conflict: $\underbrace{p_3 = 0}_{a_3} \wedge \underbrace{p_3 \geq 10}_{a_7}$ is not satisfiable.

DPLL(T)

Example

Current assignment:

$DL0 : a_4 : 1, a_7 : 1, a_8 : 1, a_9 : 1$
$DL1 : a_1 : 0$
$DL2 : a_2 : 0, a_3 : 1$

Add clause $(\neg a_3 \vee \neg a_7)$:

$$(a_1 \vee a_2 \vee a_3) \wedge a_4 \wedge (a_5 \vee a_6) \wedge a_7 \wedge a_8 \wedge a_9 \wedge (\neg a_3 \vee \neg a_7)$$

No conflict resolution needed, since the new clause is already asserting. Backtracking removes $DL1$ and $DL2$ first, then propagation is applied.

$DL0 : a_4 : 1, a_7 : 1, a_8 : 1, a_9 : 1, a_3 : 0$
$DL1 : a_1 : 0, a_2 : 1$

DPLL(T)
Example

Backtracking: remove bound $s_3 = 0$, add bound $s_2 = 0$

$p_1 = 0$	$\rightarrow$	$s_1 =$	p_1			$s_1 = 0$
$p_2 = 0$	$\rightarrow$	$s_2 =$		p_2		$s_2 = 0$
$p_3 = 0$	$\rightarrow$	$s_3 =$			p_3	$s_3 = 0$
$p_1 + p_2 + p_3 \geq 100$	$\rightarrow$	$s_4 =$	p_1+	p_2+	p_3	$s_4 \geq 100$
$p_1 \geq 5$	$\rightarrow$	$s_5 =$	p_1			$s_5 \geq 5$
$p_2 \geq 5$	$\rightarrow$	$s_6 =$		p_2		$s_6 \geq 5$
$p_3 \geq 10$	$\rightarrow$	$s_7 =$			p_3	$s_7 \geq 10$
$p_1 + 2p_2 + 5p_3 \leq 180$	$\rightarrow$	$s_8 =$	p_1+	$2p_2+$	$5p_3$	$s_8 \leq 180$
$3p_1 + 2p_2 + p_3 \leq 300$	$\rightarrow$	$s_9 =$	$3p_1+$	$2p_2+$	p_3	$s_9 \leq 300$

	$s_4(100)$	$p_2(0)$	$s_7(10)$
$s_1(90)$	1	-1	-1
$s_2(0)$	0	1	0
$s_3(10)$	0	0	1
$p_1(90)$	1	-1	-1
$s_5(90)$	1	-1	-1
$s_6(0)$	0	1	0
$p_3(10)$	0	0	1
$s_8(140)$	1	1	4
$s_9(280)$	3	-1	-2

Return partial SAT.

DPLL(T)

Example

- And so forth…

- Full example on https://ths.rwth-aachen.de/wp-content/uploads/sites/4/teaching/vorlesung_satchecking/ws14_15/06c_simplex_in_smt_handout.pdf

Integers

Integers

- Consider quantifier-free linear integer arithmetic (QF_LIA) which is like (QF_LRA) but over the integers.

Integers

- Consider quantifier-free linear integer arithmetic (QF_LIA) which is like (QF_LRA) but over the integers.
- The relaxed problem instance contains the same constraints, but assumes that variables are real-valued.

Integers

- Consider quantifier-free linear integer arithmetic (QF_LIA) which is like (QF_LRA) but over the integers.
- The relaxed problem instance contains the same constraints, but assumes that variables are real-valued.
- Solve using simplex.

Integers

- Consider quantifier-free linear integer arithmetic (QF_LIA) which is like (QF_LRA) but over the integers.
- The relaxed problem instance contains the same constraints, but assumes that variables are real-valued.
- Solve using simplex.
    - UNSAT implies original problem is UNSAT.

Integers

- Consider quantifier-free linear integer arithmetic (QF_LIA) which is like (QF_LRA) but over the integers.
- The relaxed problem instance contains the same constraints, but assumes that variables are real-valued.
- Solve using simplex.
    - UNSAT implies original problem is UNSAT.
    - SAT implies…

Integers

- Consider quantifier-free linear integer arithmetic (QF_LIA) which is like (QF_LRA) but over the integers.
- The relaxed problem instance contains the same constraints, but assumes that variables are real-valued.
- Solve using simplex.
  - UNSAT implies original problem is UNSAT.
  - SAT implies…
    - Either all variables are assigned with an integer -> SAT

Integers

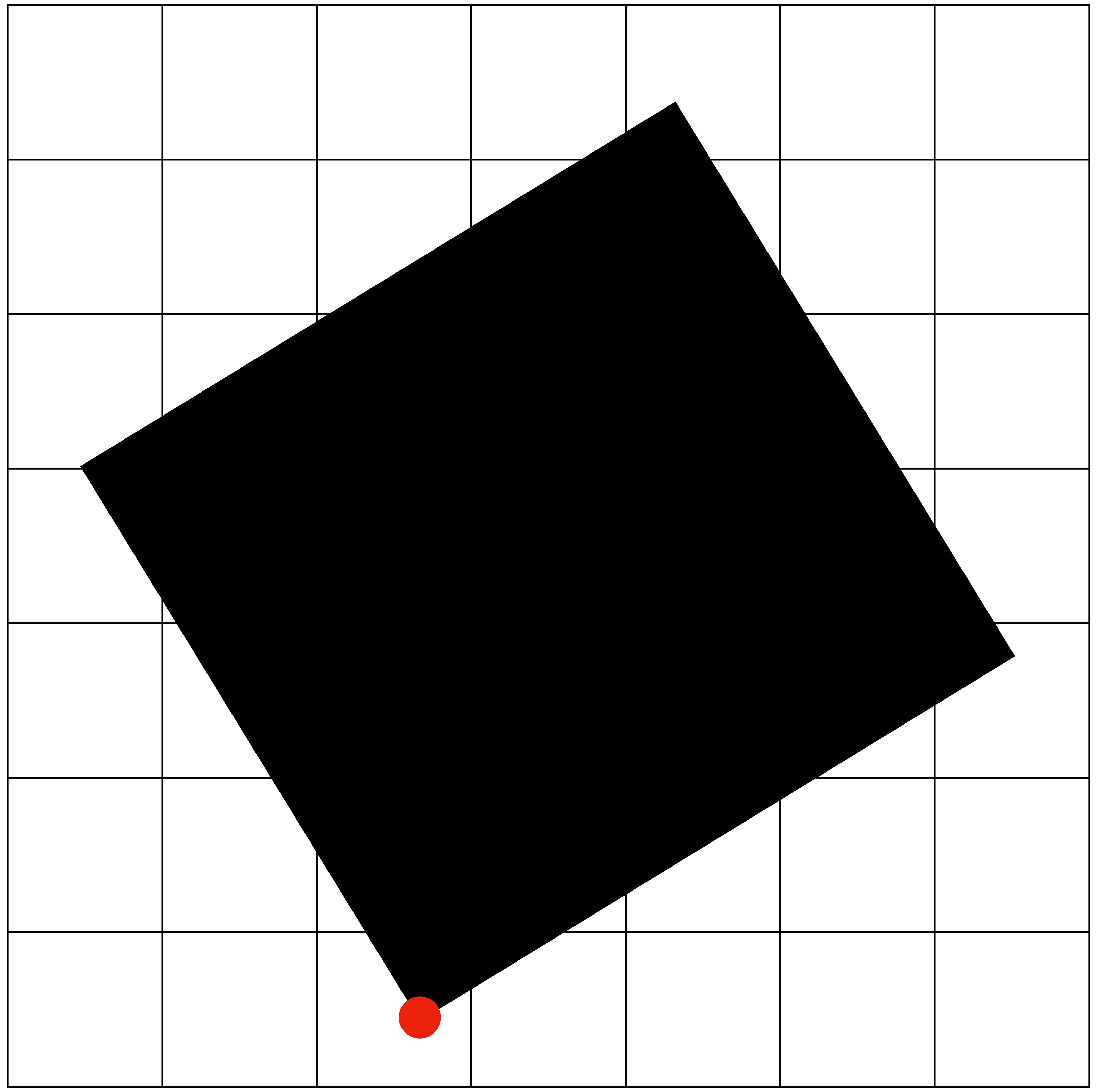
- Consider quantifier-free linear integer arithmetic (QF_LIA) which is like (QF_LRA) but over the integers.
- The relaxed problem instance contains the same constraints, but assumes that variables are real-valued.
- Solve using simplex.
  - UNSAT implies original problem is UNSAT.
  - SAT implies…
    - Either all variables are assigned with an integer -> SAT
    - At least one variable is assigned a real, non-integer value

Summary

- Linear programming & QF_LRA
- Generalized Simplex
- Incrementality & Backtracking for Generalized Simplex
- Branch & Bound for QF_LIA
- Difference logic

Integers

Branch-And-Bound Illustrated

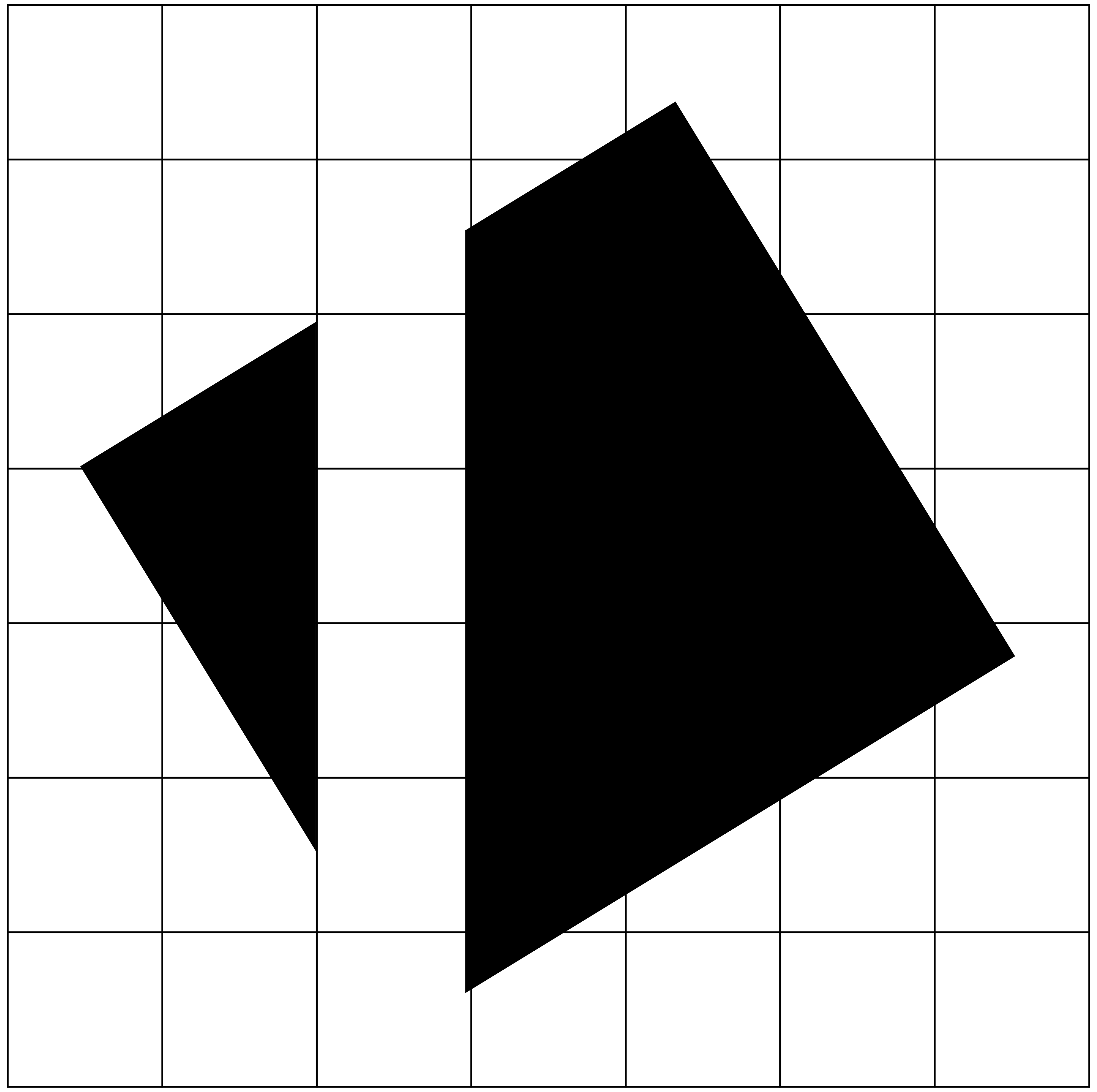


Solve the relaxed problem,

Find a real-valued solution

Integers

Branch-And-Bound Illustrated

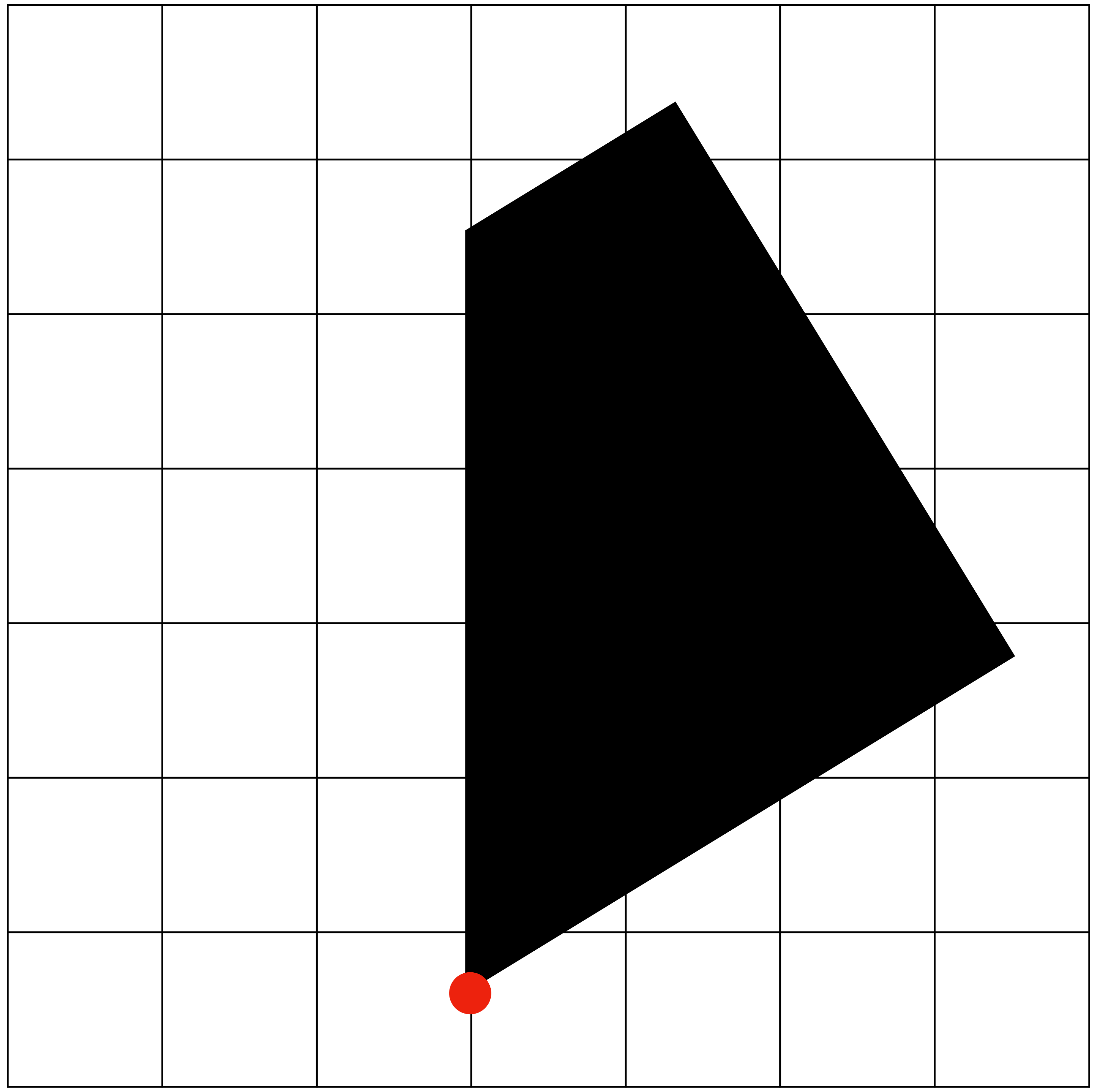


Exclude the solution

No longer conjunction of linear constraints

Integers

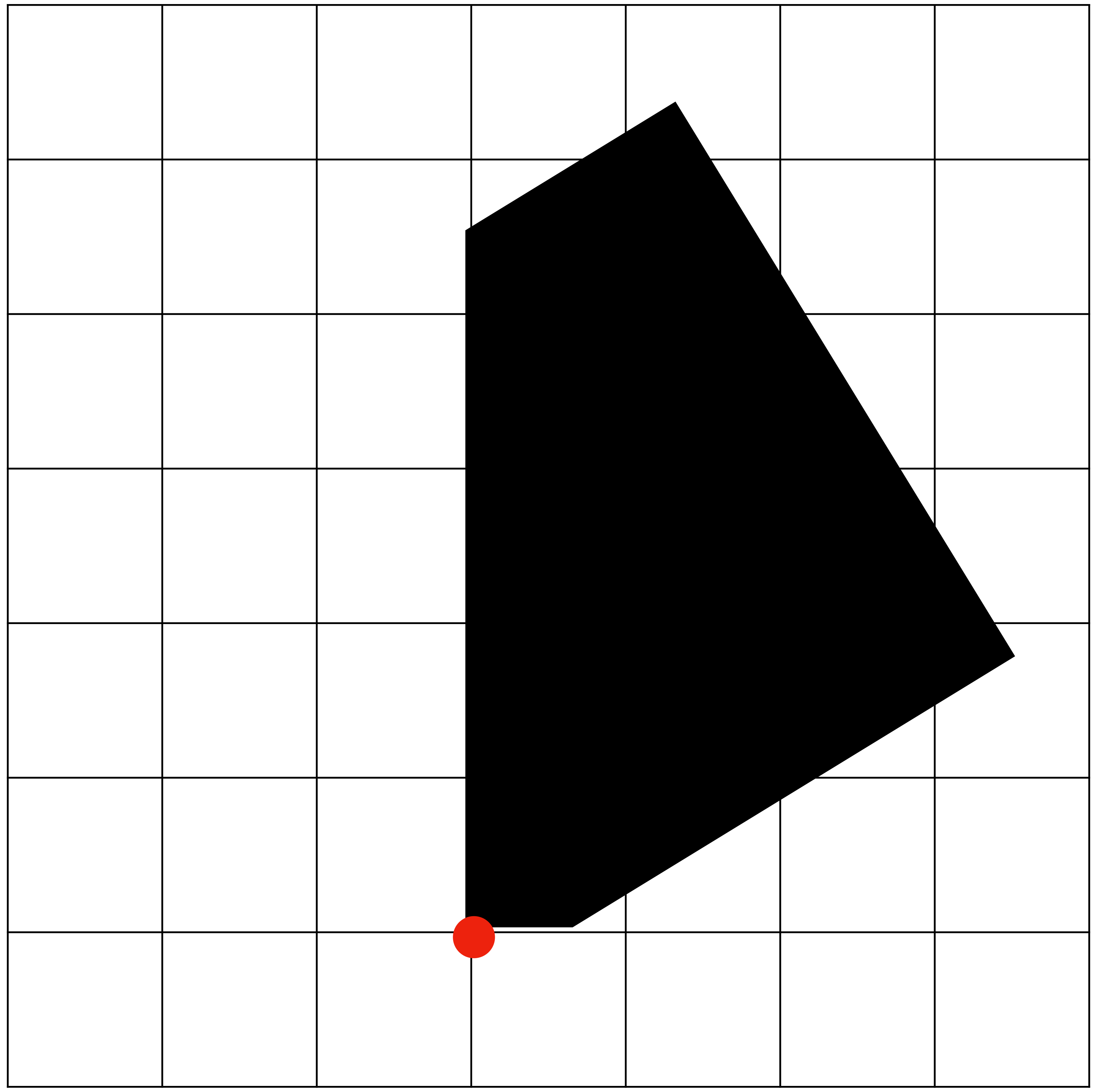
Branch-And-Bound Illustrated



Guess one side, repeat

Integers

Branch-And-Bound Illustrated



Until we find an integer solution

Branch-And-Bound

Recursive algorithm

Branch-And-Bound

Recursive algorithm

Branch-and-bound (constraints C)

Branch-And-Bound

Recursive algorithm

Branch-and-bound (constraints C)

1. Solve relaxed version using Simplex

Branch-And-Bound

Recursive algorithm

Branch-and-bound (constraints C)

1. Solve relaxed version using Simplex

2. **If** assignment α found

Branch-And-Bound

Recursive algorithm

Branch-and-bound (constraints C)

1. Solve relaxed version using Simplex
2. **If** assignment α found
   A. **If** $\alpha(x)$ integral for all variables **return** SAT

Branch-And-Bound

Recursive algorithm

Branch-and-bound (constraints C)

1. Solve relaxed version using Simplex
2. **If** assignment α found
   A. **If** $\alpha(x)$ integral for all variables **return** SAT
   B. Pick variable x with $\alpha(x)$ not integral, $\alpha(x) = d\,.\,mmmm$

Branch-And-Bound

Recursive algorithm

Branch-and-bound (constraints C)

1. Solve relaxed version using Simplex
2. **If** assignment α found
   A. **If** $\alpha(x)$ integral for all variables **return** SAT
   B. Pick variable x with $\alpha(x)$ not integral, $\alpha(x) = d\,.\,mmmm$
   C. **If** Branch-and-bound $(C \cup \{x \leq d\})$ == SAT **return** SAT

Branch-And-Bound

Recursive algorithm

Branch-and-bound (constraints C)

1. Solve relaxed version using Simplex
2. **If** assignment α found
   A. **If** $\alpha(x)$ integral for all variables **return** SAT
   B. Pick variable x with $\alpha(x)$ not integral, $\alpha(x) = d\,.\,mmmm$
   C. **If** Branch-and-bound $(C \cup \{x \leq d\})$ == SAT **return** SAT
   D. **If** Branch-and-bound $(C \cup \{x \geq d + 1\})$ == SAT **return** SAT

Branch-And-Bound

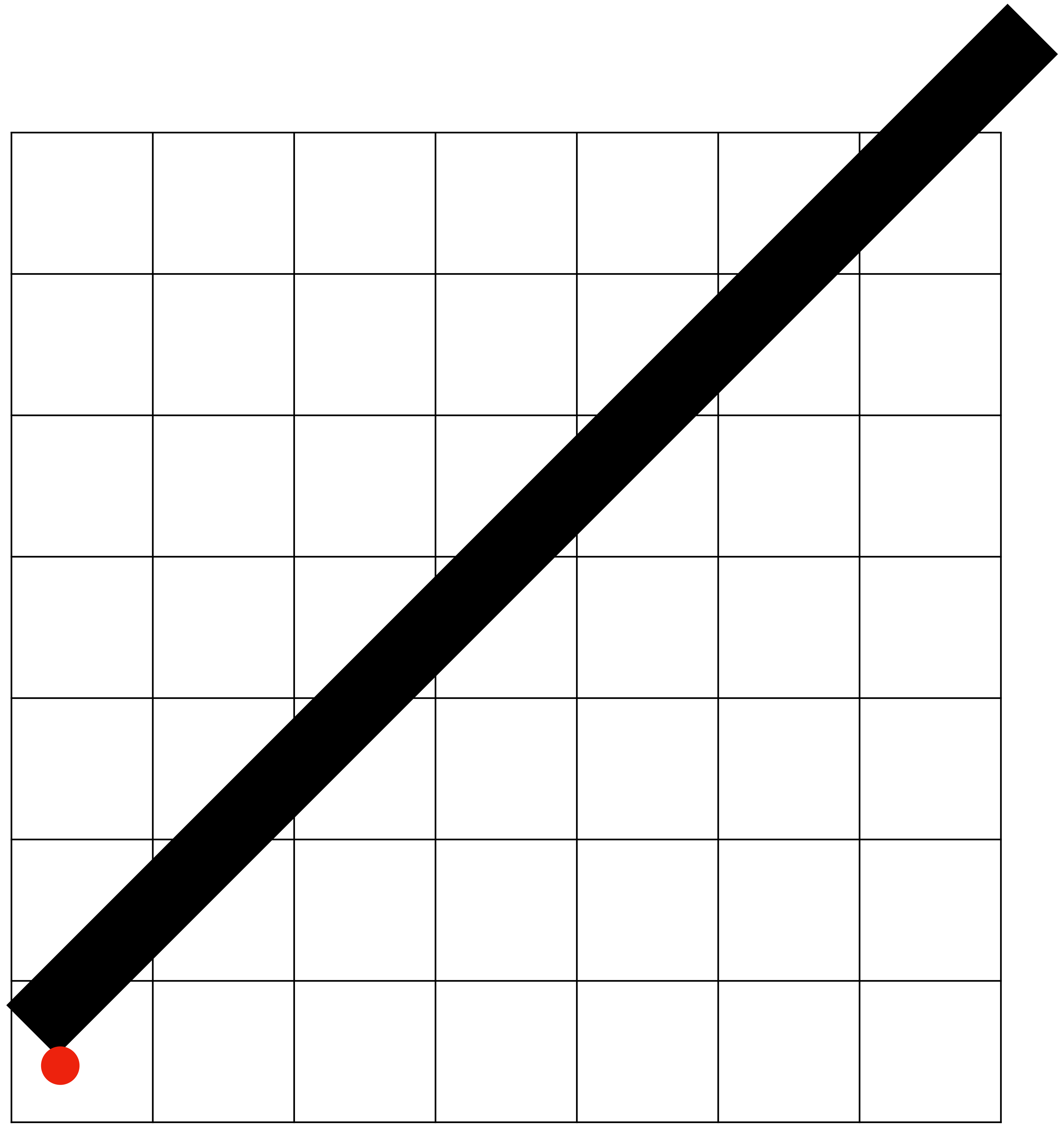
Recursive algorithm

Branch-and-bound (constraints C)

1. Solve relaxed version using Simplex
2. **If** assignment α found
   A. **If** $\alpha(x)$ integral for all variables **return** SAT
   B. Pick variable x with $\alpha(x)$ not integral, $\alpha(x) = d\,.\,mmmm$
   C. **If** Branch-and-bound $(C \cup \{x \leq d\})$ == SAT **return** SAT
   D. **If** Branch-and-bound $(C \cup \{x \geq d + 1\})$ == SAT **return** SAT
3. **return** UNSAT

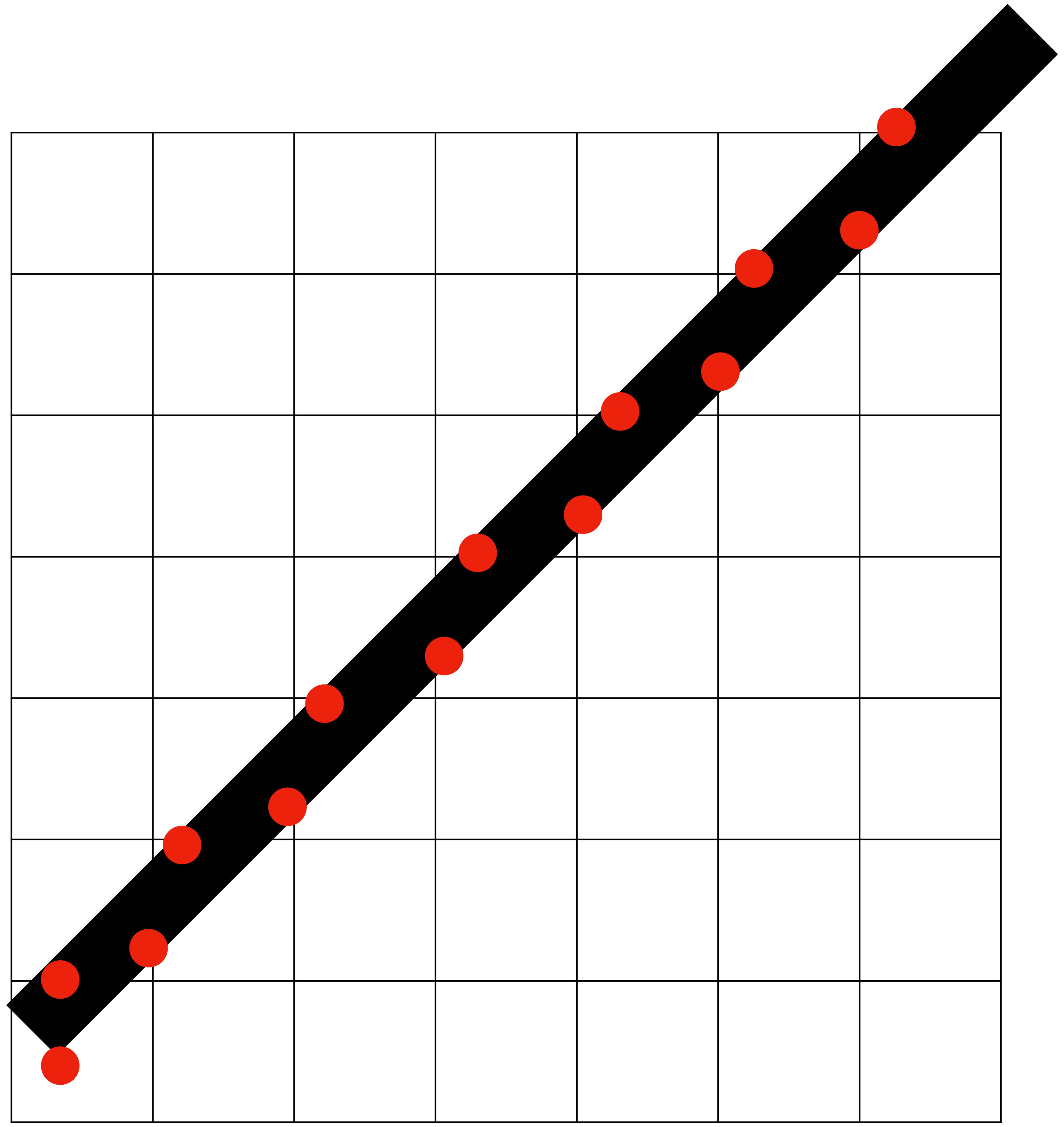
Integers

Branch-And-Bound Illustrated



Integers

Branch-And-Bound Illustrated



May diverge
Need small-model property to bound all integers

QF_LIA

- Many more ways to cut away integer-problems
- Even for a conjunction (feasiblity of Integer LP) the problem is NP-complete
- In less-lazy SMT, solve relaxed version unless it is a complete assignment

Summary

- Linear programming & QF_LRA
- Generalized Simplex
- Incrementality & Backtracking for Generalized Simplex
- Branch & Bound for QF_LIA
- Difference logic

Job-Shop Scheduling

Example

- Finite N jobs consist of a chain of ordered operations
- Finite M machines that execute an operation
- Each operation consists of a machine and a duration; must be executed on machine for duration time without interruption
- Order of operations must be respected

Quiztime

Encoding?!

- Finite N jobs consist of a chain of ordered operations
- Finite M machines that execute an operation
- Each operation consists of a machine and a duration; must be executed on machine for duration time without interruption
- Order of operations must be respected

Solution

- x_p start time of operation
- $x_p \geq 0$
- $\forall p, p'$ s.t. p' in same job and after p : $x_{p'} - x_p \geq$ duration(p)
- $\forall p, p'$ with machine(p) $=$ machine(p'):
  $x_{p'} +$ duration$(p') \leq x_p \vee x_p +$ duration$(p) \leq x_{p'}$

Difference Logic

QF_DL

- Real (or integer) variables x_i, constants c
- $x_i - x_j \leq c$
- Boolean combinations

- Use x_0 special variable for 0 (and add implicit constraints $x_0 \leq x_i$)

Quiztime

- $x_1 - x_2 \leq 1 \land x_2 - x_3 \leq -1 \land x_3 - x_1 \leq -3$
- Satisfiable?

Inequality Graph

- Edges = Variables including the x_0 variable
- Constraints: For every $x - y \leq c$ an edge from x to y with weight c

Inequality Graph

- Edges = Variables including the x_0 variable
- Constraints: For every $x - y \leq c$ an edge from x to y with weight c

There is a negative cycle in the inequality graph iff the conjunction of constraints together is unsatisfiable

Inequality Graph

- Edges = Variables including the x_0 variable
- Constraints: For every $x - y \leq c$ an edge from x to y with weight c

There is a negative cycle in the inequality graph iff the conjunction of constraints together is unsatisfiable

Bellman-Ford (source = x_0) detects negative cycles or finds a satisfying assignment

End of SAT/SMT

End of SAT/SMT

- DPLL/CDCL

End of SAT/SMT

- DPLL/CDCL
- Resolution for proofs of unsatisfiability and for clause learning

End of SAT/SMT

- DPLL/CDCL
- Resolution for proofs of unsatisfiability and for clause learning
- Eager encodings

End of SAT/SMT

- DPLL/CDCL
- Resolution for proofs of unsatisfiability and for clause learning
- Eager encodings
- DPLL(T)

End of SAT/SMT

- DPLL/CDCL
- Resolution for proofs of unsatisfiability and for clause learning
- Eager encodings
- DPLL(T)
- Linear arithmetic & Simplex

End of SAT/SMT

- DPLL/CDCL
- Resolution for proofs of unsatisfiability and for clause learning
- Eager encodings
- DPLL(T)
- Linear arithmetic & Simplex
- Many, many more logics & theory solvers….

End of SAT/SMT

- DPLL/CDCL
- Resolution for proofs of unsatisfiability and for clause learning
- Eager encodings
- DPLL(T)
- Linear arithmetic & Simplex
- Many, many more logics & theory solvers….
- But automated reasoning is more than reasoning about satisfiability….

Summary

- Linear programming & QF_LRA
- Generalized Simplex
- Incrementality & Backtracking
- Branch & Bound for QF_LIA
- Difference logic

Learning goals

You are able to:

- Describe the integration of theory solvers in a (less) lazy DPLL(T) loop
- Explain main ingredients of simplex, execute pivot steps and deduce satisfiability
- Explain and execute branch & bound for integers
- Create inequality graphs to show (un)satisfiability of difference logic

Questions

Make sure you can answer the following questions!

A. What is a linear program? What is an LRA formula?

B. Which data structures does simplex maintain?

C. How does simplex update an variable with a violated bound?

D. What is branch and bound? What problems can it solve?

E. What is difference logic and how can we solve conjunctions?

Be able to construct LRA formulas for problem statements

See you next week!