Satisfiability Checking

Summary II

Prof. Dr. Erika Ábrahám

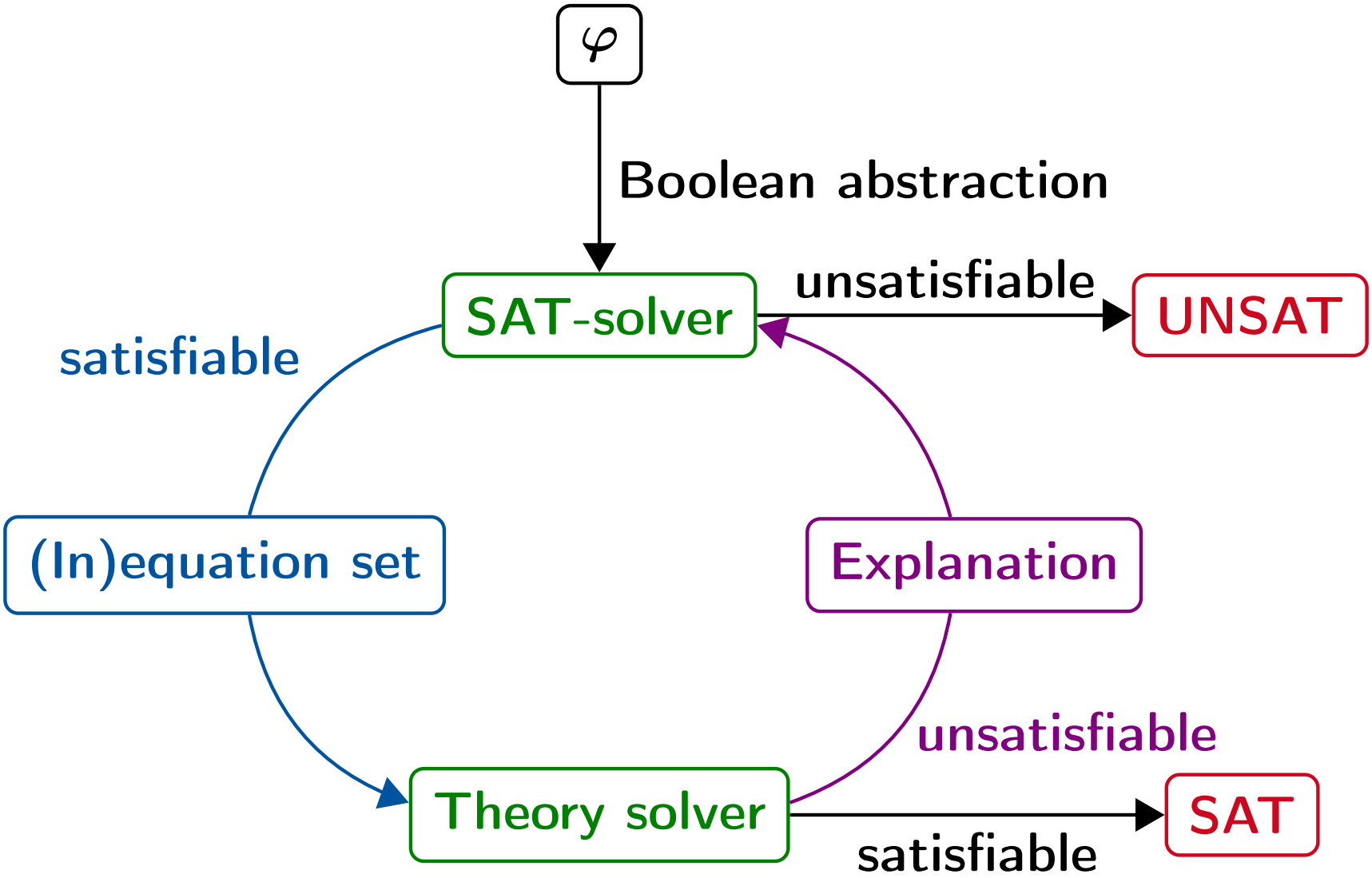
RWTH Aachen University
Informatik 2
LuFG Theory of Hybrid Systems

WS 19/20

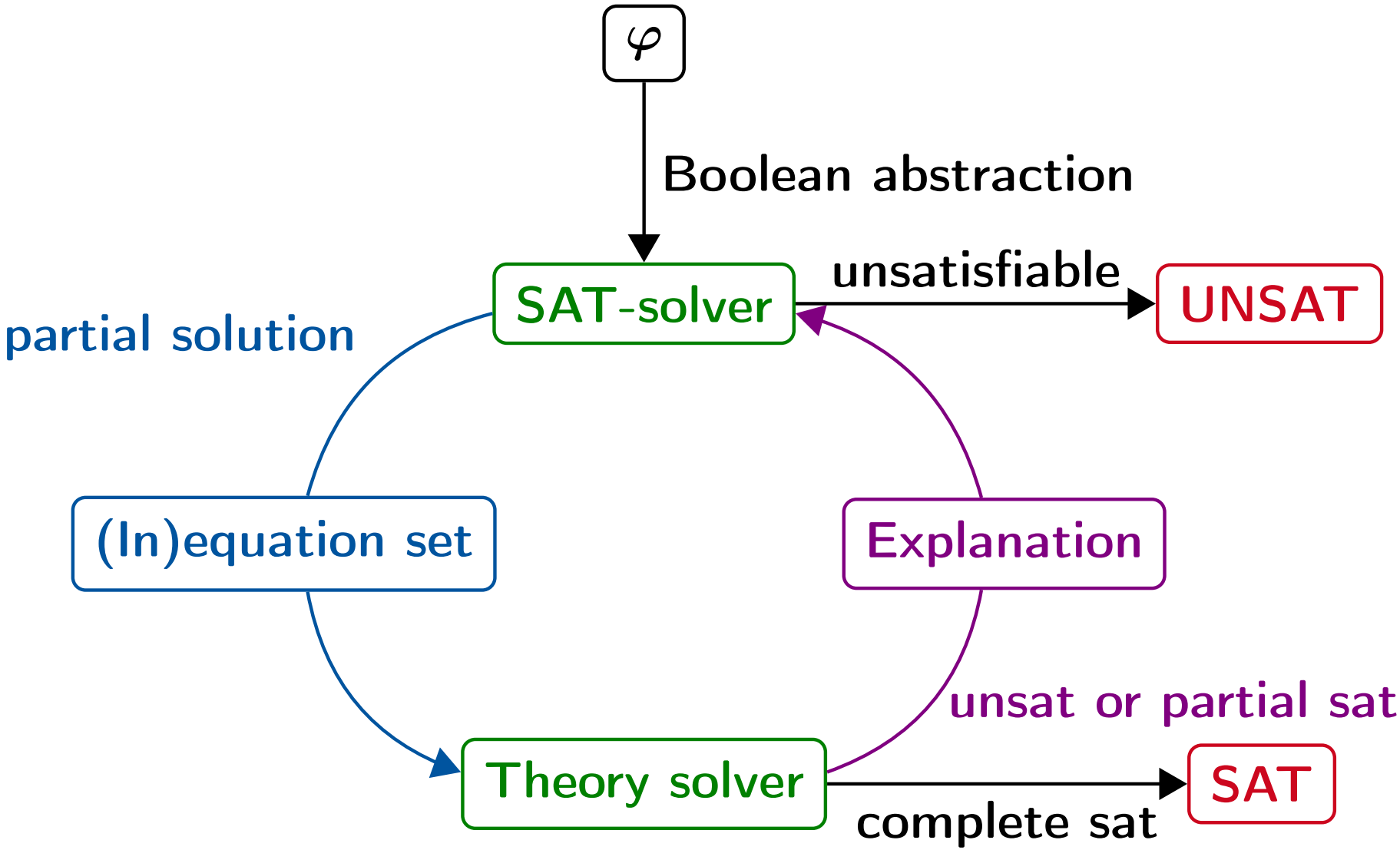
Outline

Outline

Full lazy SMT-solving



Less lazy SMT-solving



Requirements on the theory solver

1. Incrementality: In less lazy solving we incrementally extend a set of constraints, whose satisfiability check should be carried out by the theory solver. The theory solver should make use of the previous satisfiability check for the analysis of the extended set.
2. (Preferably minimal) infeasible subsets: If the constraint set is infeasible then compute a reason for unsatisfaction.
3. Backtracking: The theory solver should be able to remove constraints (in inverse chronological order).

Outline

Conjunctions of equalities: Transitive closure

$$\varphi^E : \quad x_1 = x_2 \wedge x_2 = x_3 \wedge x_4 = x_5 \wedge x_5 \neq x_1$$

Transitive closure:

For each equality, merge the classes of the respective variables!

Equivalence class 1 Equivalence class 2

Conjunctions of equalities: Transitive closure

$$\varphi^E : \quad x_1 = x_2 \wedge x_2 = x_3 \wedge x_4 = x_5 \wedge x_5 \neq x_1$$

Transitive closure:

For each equality, merge the classes of the respective variables!

Equivalence class 1 Equivalence class 2

How to achieve incrementality? How to compute infeasible subsets?

Conjunction of equalities: Algorithm

Input: A conjunction φ of equalities and disequalities without UF

Algorithm

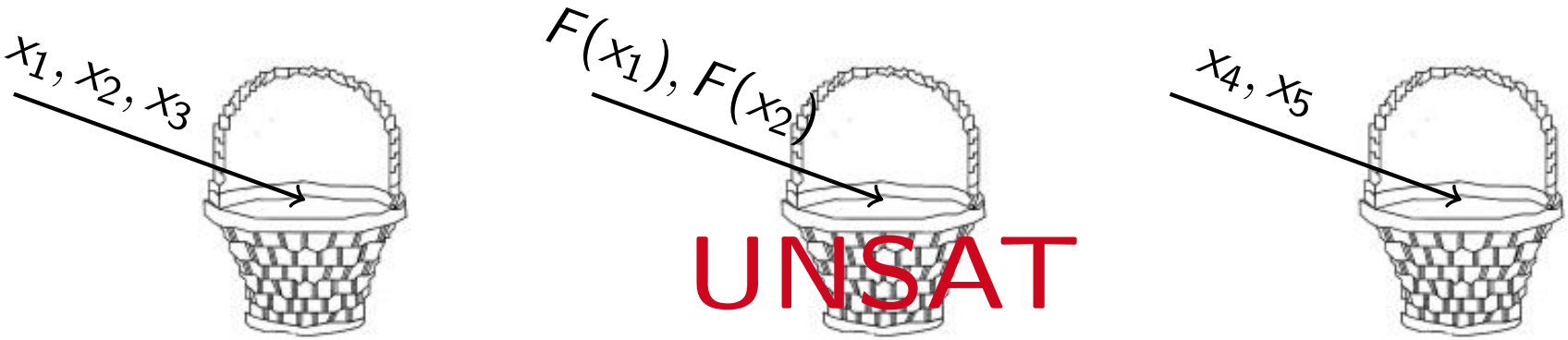
1. Define an equivalence class for each variable in φ.
2. For each equality $x = y$ in φ: merge the equivalence classes of x and y.
3. For each disequality $x \neq y$ in φ:
   if x is in the same class as y, return 'UNSAT'.
4. Return 'SAT'.

Uninterpreted functions: Congruence closure

$$\varphi^E : \quad x_1 = x_2 \wedge x_2 = x_3 \wedge x_4 = x_5 \wedge x_5 \neq x_1 \wedge F(x_1) \neq F(x_2)$$

Congruence closure:

If all the arguments of two function applications are in the same class, merge the classes of the applications!



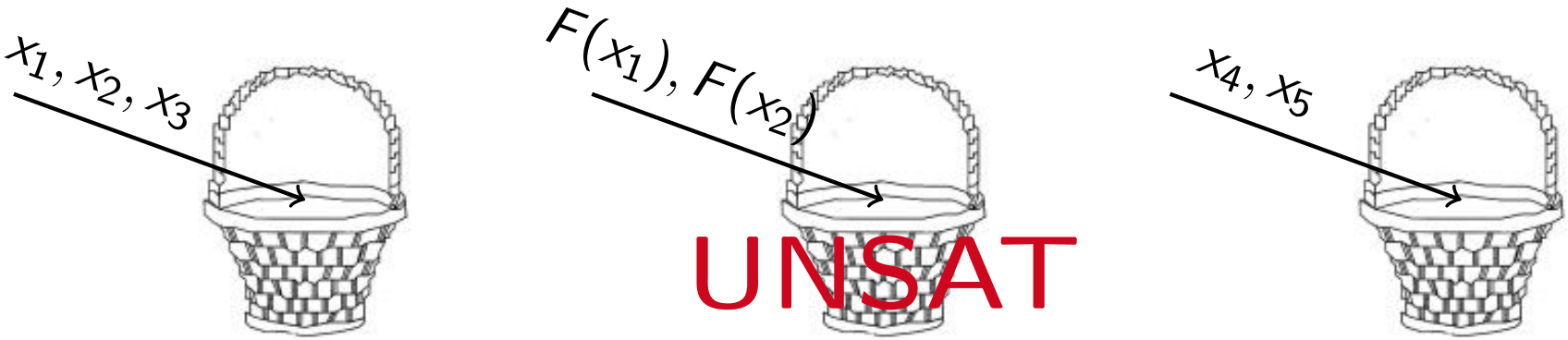
Equivalence class 1 Equivalence class 2 Equivalence class 3

Uninterpreted functions: Congruence closure

$$\varphi^E : \quad x_1 = x_2 \wedge x_2 = x_3 \wedge x_4 = x_5 \wedge x_5 \neq x_1 \wedge F(x_1) \neq F(x_2)$$

Congruence closure:

If all the arguments of two function applications are in the same class, merge the classes of the applications!



Equivalence class 1 Equivalence class 2 Equivalence class 3

How to achieve incrementality? How to compute infeasible subsets?

Conjunction of equalities with UF: Algorithm

Input: A conjunction φ of equalities and disequalities with UFs of type $D \to D$

Output: Satisfiability of φ

Algorithm

1. $\mathcal{C} := \{\{t\} \mid t \text{ occurs as subexpression in an (in)equation in } \varphi\}$;
2. for each equality $t = t'$ in φ with $[t] \neq [t']$
   $\mathcal{C} := (\mathcal{C} \setminus \{[t], [t']\}) \cup \{[t] \cup [t']\}$;
   while exists $F(t), F(t')$ in φ with $[t] = [t']$ and $[F(t)] \neq [F(t')]$
   $\mathcal{C} := (\mathcal{C} \setminus \{[F(t)], [F(t')]\}) \cup \{[F(t)] \cup [F(t')]\}$;
3. for each inequality $t \neq t'$ in φ
   if $[t] = [t']$ return "UNSAT";
4. return "SAT";

($[t] \in \mathcal{C}$ denotes the unique set in $\mathcal{C}$ that contains t)

Outline

Gaussian and Fourier-Motzkin variable elimination

- Goal: decide satisfiability of conjunctions of linear constraints over the reals (n variables, k inequations, l equations)

$$\bigwedge_{1 \leq i \leq k} \sum_{1 \leq j \leq n} a_{ij} x_j \leq b_i \wedge \bigwedge_{1 \leq i \leq l} \sum_{1 \leq j \leq n} c_{ij} x_j = d_i$$

- Eliminate variable x_n:

  Gauss: If there exists an equation $\sum_{1 \leq j \leq n} c_{ij} x_j = d_i$ with $c_{in} \neq 0$ then remove this equation and replace x_n by $\frac{d_i}{c_{in}} - \sum_{j=1}^{n-1} \frac{c_{ij}}{c_{in}} x_j$ in all remaining constraints.

  Fourier-Motzkin: Otherwise partition the inequations according to the coefficients a_{in}:

  - $a_{in} = 0$: no bound on x_n
  - $a_{in} > 0$: upper bound $\beta_i = \frac{b_i}{a_{in}} - \sum_{j=1}^{n-1} \frac{a_{ij}}{a_{in}} x_j$ on x_n
  - $a_{in} < 0$: lower bound $\beta_i = \frac{b_i}{a_{in}} - \sum_{j=1}^{n-1} \frac{a_{ij}}{a_{in}} x_j$ on x_n

  Remove all inequalitites defining a bound on x_n and add for each pair of a lower bound β_l and upper bound β_u the constraint $\beta_l \leq \beta_u$.

Gaussian and Fourier-Motzkin variable elimination

How to achieve incrementality?
How to compute infeasible subsets?

Outline

Satisfiability with simplex

- Problem: Check the satisfiability of a linear inequation system

$$\bigwedge_{i=1}^{m} \sum_{j=1}^{n} a_{ij} x_j \bowtie_i b_i \qquad \bowtie_i \in \{=, \leq, \geq\}$$

  in the real domain

- General form:

$$A\vec{x} = \vec{s} \quad \wedge \quad \bigwedge_{i=1}^{m} s_i \bowtie_i b_i$$

  $s_1, \ldots, s_m$: additional/auxiliary/slack variables

- Tableau:

$$\begin{array}{rc} & \begin{array}{ccc} x_1 & \cdots & x_n \end{array} \\ \text{Basic variables } \mathcal{B} \rightarrow \begin{array}{c} s_1 \\ \cdots \\ s_m \end{array} & \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ & \cdots & \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} \end{array} \leftarrow \text{Non-basic variables } \mathcal{N}$$

Data structures

- Simplex maintains:
  - The tableau,
  - an assignment α to all (original and additional) variables.

  Initially, $\alpha(x_i) = 0$ for $i \in \{1, ..., n + m\}$
- Two invariants are maintained throughout:
  - $A\vec{x} = \vec{s}$
  - All non-basic variables satisfy their bounds

1. If the bounds of all basic variables are satisfied by α, return "satisfiable".
2. Otherwise, find a basic variable x_i that violates its bounds. Suppose that $\alpha(x_i) < l_i$.
3. Find a suitable non-basic variable x_j such that
   - $a_{ij} > 0$ and $\alpha(x_j) < u_j$, or
   - $a_{ij} < 0$ and $\alpha(x_j) > l_j$.
4. If there is no suitable variable, return "unsatisfiable".
5. Otherwise, pivot.

Pivoting

$$
\begin{array}{c} \\ y_1 \\ \dots \\ y_i \\ \dots \\ y_m \end{array}
\begin{array}{c}
\begin{array}{cccccc} y'_1 & \dots & y'_j & \dots & y'_n \end{array} \\
\begin{pmatrix}
a_{11} & \dots & a_{1j} & \dots & a_{1n} \\
\dots & & \dots & & \\
a_{i1} & \dots & a_{ij} & \dots & a_{in} \\
\dots & & \dots & & \\
a_{m1} & \dots & a_{mj} & \dots & a_{mn}
\end{pmatrix}
\end{array}
$$

$$\Downarrow$$

$$
\begin{array}{c} \\ y_1 \\ \dots \\ y'_j \\ \dots \\ y_m \end{array}
\begin{array}{c}
\begin{array}{cccccc} y'_1 & \dots & y_i & \dots & y'_n \end{array} \\
\begin{pmatrix}
a_{11} - \frac{a_{1j}a_{i1}}{a_{ij}} & \dots & \frac{a_{1j}}{a_{ij}} & \dots & a_{1n} - \frac{a_{1j}a_{in}}{a_{ij}} \\
\dots & & \dots & & \\
-\frac{a_{i1}}{a_{ij}} & \dots & \frac{1}{a_{ij}} & \dots & -\frac{a_{in}}{a_{ij}} \\
\dots & & \dots & & \\
a_{m1} - \frac{a_{mj}a_{i1}}{a_{ij}} & \dots & \frac{a_{mj}}{a_{ij}} & \dots & a_{mn} - \frac{a_{mj}a_{in}}{a_{ij}}
\end{pmatrix}
\end{array}
$$

1. $\alpha(y_i) = l_i$ 2. $\alpha(y'_k)$ unchanged for $k \neq j$ 3. rest according to the new matrix

Termination

To achieve completeness, we use Bland's rule:

1. Determine a total order on the variables
2. Choose the first basic variable that violates its bounds, and the first non-basic suitable variable for pivoting.

General simplex with Bland's rule

1. Transform the system into the general form

$$A\vec{x} = \vec{s} \quad \text{and} \quad \bigwedge_{i=1}^{m} l_i \leq s_i \leq u_i \ .$$

2. Construct the tableau with the initial assignment.
3. Determine a fixed order on the variables.
4. If there is no basic variable that violates its bounds, return "satisfiable". Otherwise, let y_i be the first basic variable in the order that violates its bounds.
5. Search for the first suitable non-basic variable y'_j in the order for pivoting with y_i. If there is no such variable, return "unsatisfiable".
6. Perform the pivot operation on y_i and y'_j.
7. Go to step 4.

Requirements on the theory solver

- (Minimal) infeasible subsets (to explain infeasibility)
- Incrementality (to add constraints stepwise)
- Backtracking (to mimic backtracking in the SAT solver)

Minimal infeasible subsets in simplex:

- The constraints corresponding to the basic variable of the contradictory row and all non-basic variables with non-zero coefficients in this row are together unsatisfiable.

Incrementality in simplex:

- Add all constraints but without bounds on non-active constraints.
- If a constraint becomes true, activate its bound.

Backtracking in simplex:

- Remove bounds of unassigned constraints

Outline

Integer linear systems

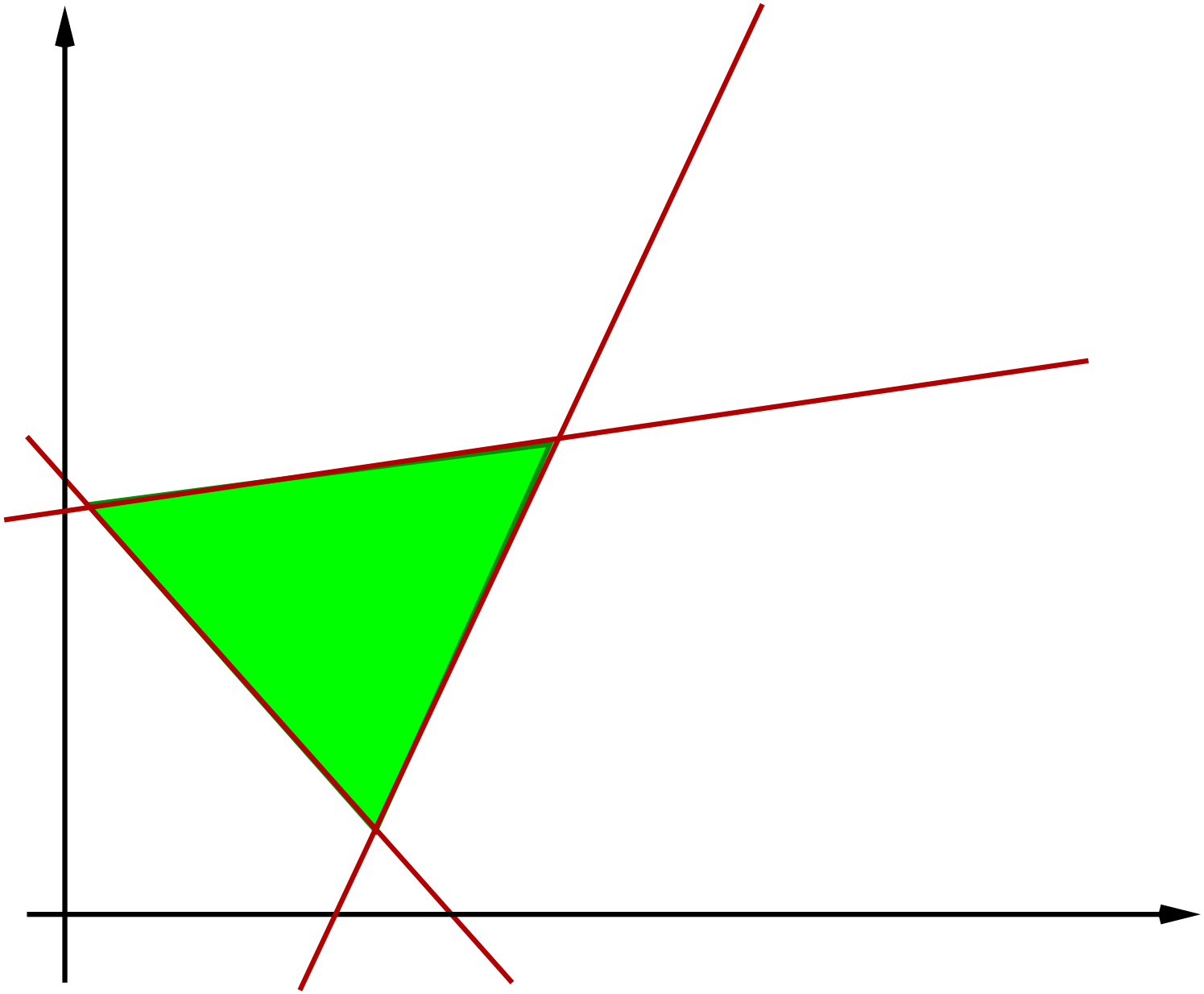
Definition

An integer linear system S is a linear system $Ax = 0$, $\bigwedge_{i=1}^{m} l_i \leq s_i \leq u_i$, with the additional integrality requirement that all variables are of type integer.

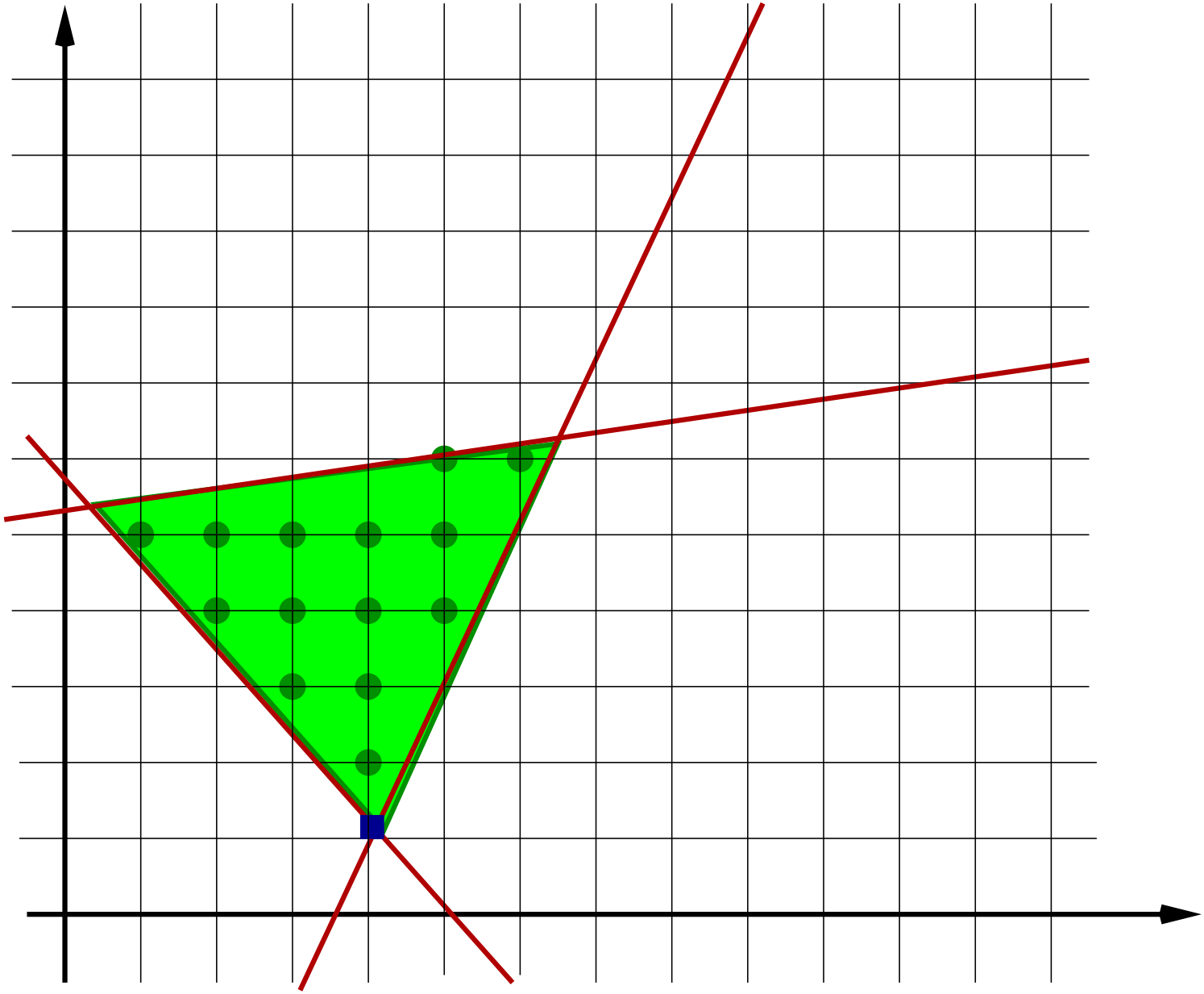
Definition (relaxed system)

Given an integer linear system S, its relaxation relaxed(S) is S without the integrality requirement.

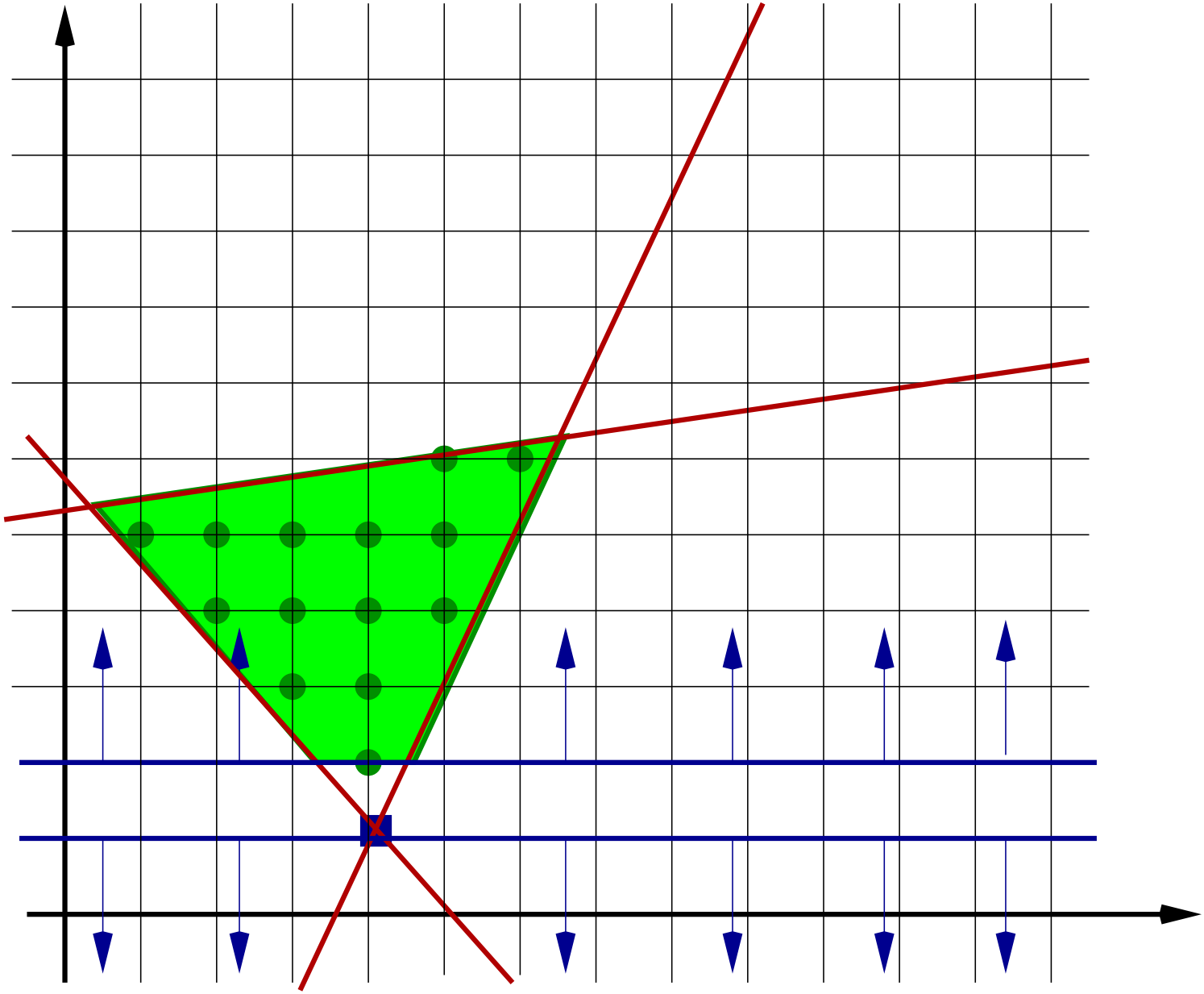
Geometric interpretation



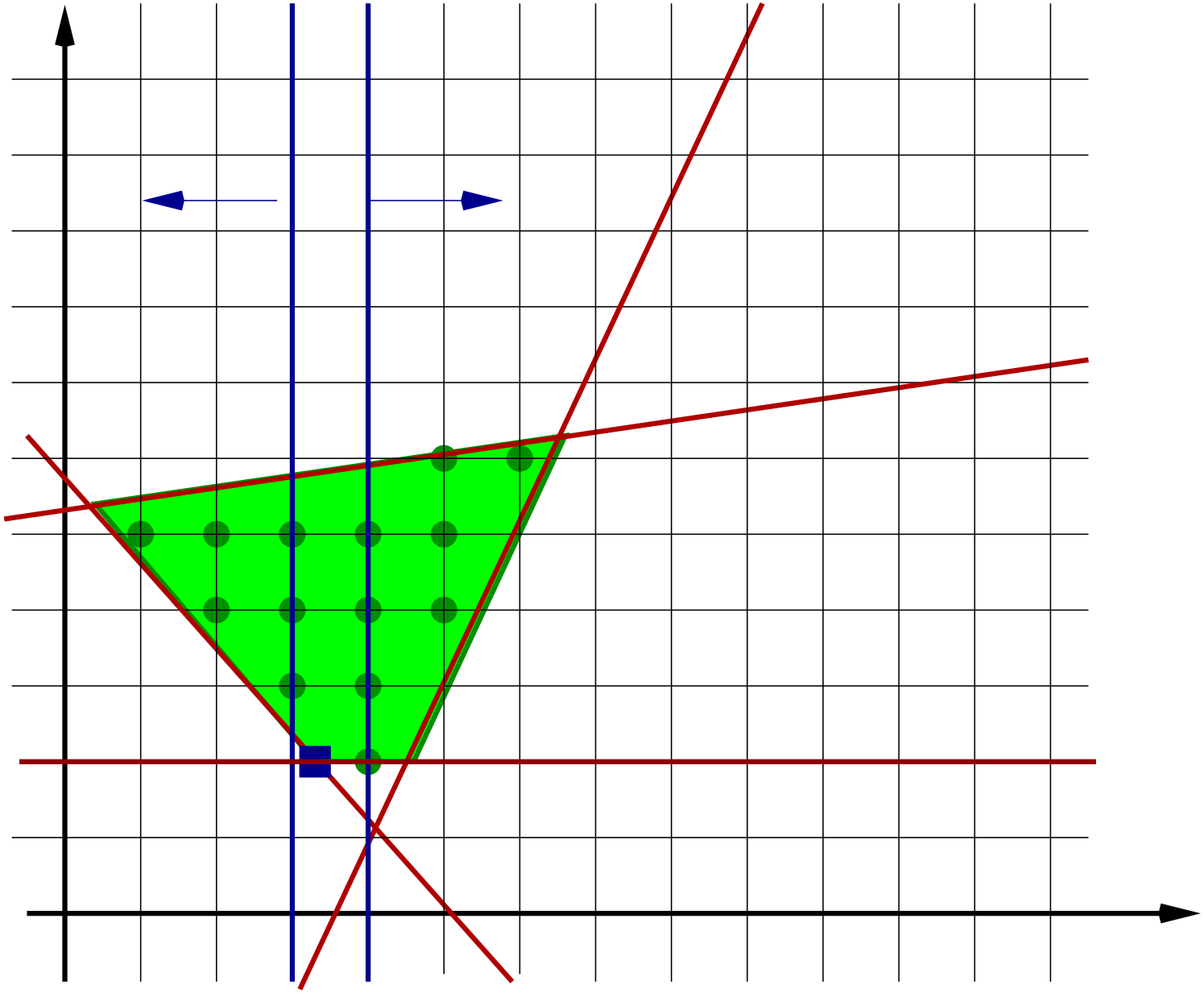
Geometric interpretation



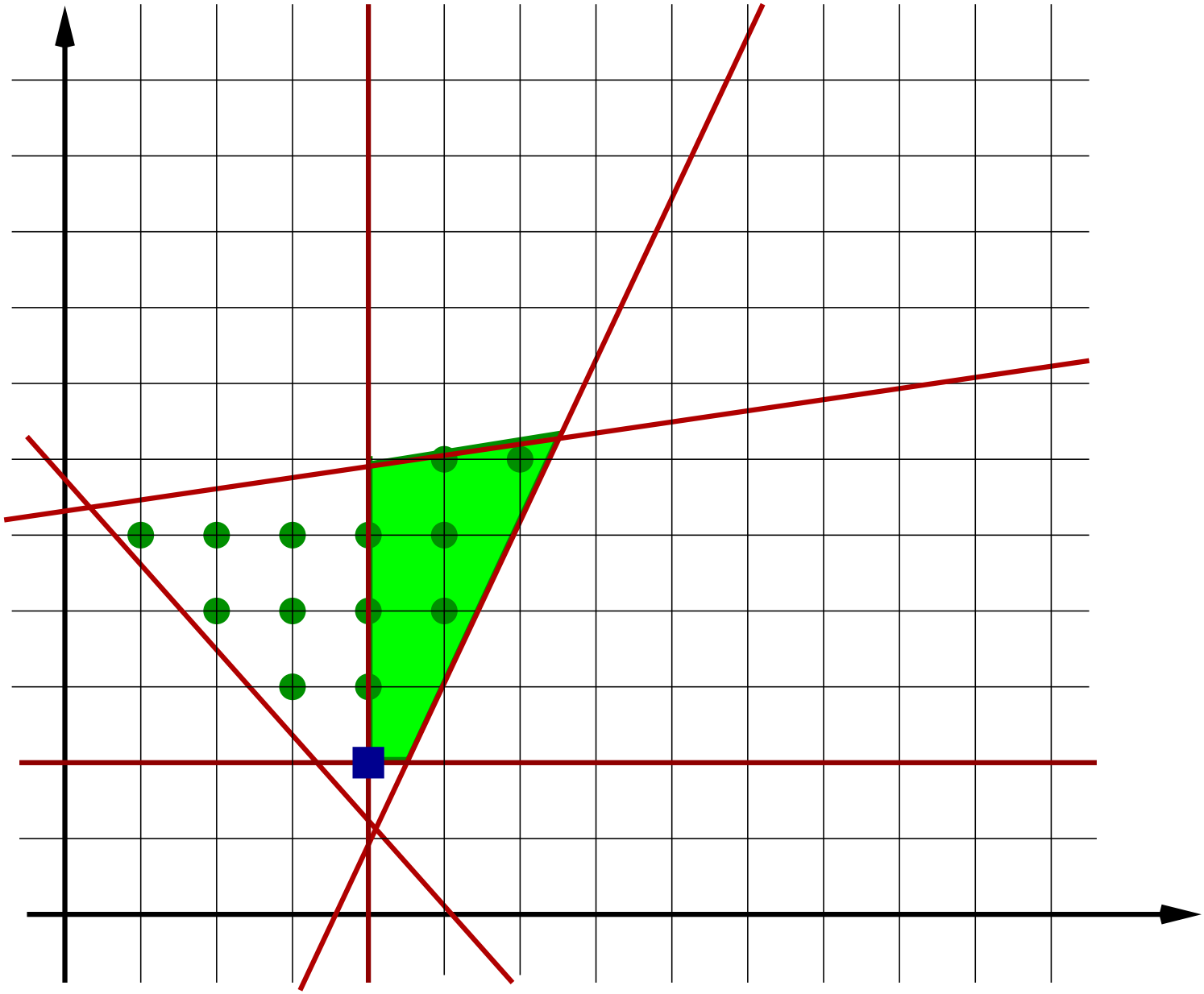
Geometric interpretation



Geometric interpretation



Geometric interpretation



Branch and bound algorithm

Input: An integer linear system S
Output: SAT if S is satisfiable, UNSAT otherwise

```
procedure Branch-and-Bound(S) {
  res = LP(relaxed(S));
  if (res==UNSAT) return UNSAT;
  else if (res is integral) return SAT;
  else {
    Select a variable v that is assigned a non-integral value r;
    if (Branch-and-Bound(S ∪ (v ≤ ⌊r⌋))==SAT) return SAT;
    else if (Branch-and-Bound(S ∪ (v ≥ ⌈r⌉))==SAT) return SAT;
    else return UNSAT;
  }
}
```