Stabbing Planes

Noah Fleming
Department of Computer Science
University of Toronto
Joint work with Paul Beame, Russell Impagliazzo, Antonina Kolokolova,
Denis Pankratov, Toniann Pitassi, and Robert Robere

Search

SAT - NP hard

- Intractable in worst-case

SAT Solvers

- Solve real world instances with millions of variables
- often run in near-linear time!
- Used in model checking, planning, bioinformatics, etc.

Search

SAT - NP hard

- Intractable in worst-case

SAT Solvers

- Solve real world instances with millions of variables
- Often run in near-linear time!
- Used in model checking, planning, bioinformatics, etc.

Conflict-Driven Clause Learning (CDCL)

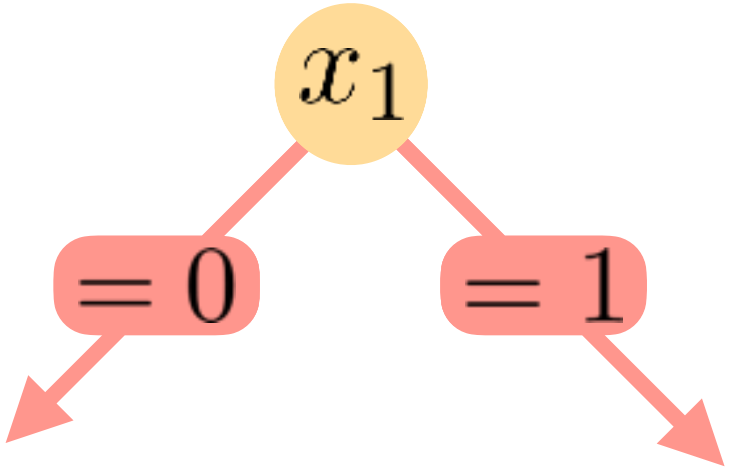
- Basis of state-of-the-art solvers
- Based on *Davis–Putnam–Logemann–Loveland (DPLL)* algorithm [DP60, DLL62] augmented with fine-tuned heuristics

DPLL

$$\mathcal{F} = (x_1 \vee x_2) \wedge (\bar{x}_1 \vee x_2) \wedge (x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee \bar{x}_2)$$

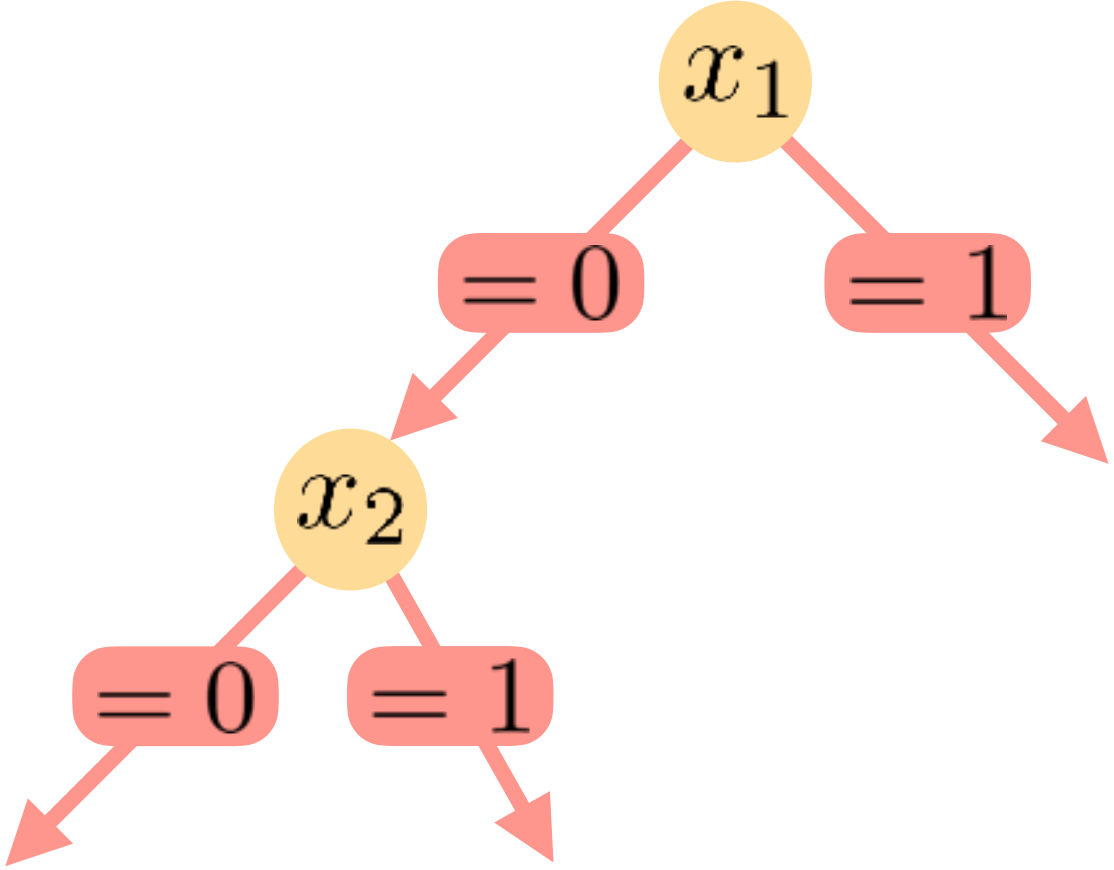
DPLL

$$\mathcal{F} = (x_1 \vee x_2) \wedge (\bar{x}_1 \vee x_2) \wedge (x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee \bar{x}_2)$$



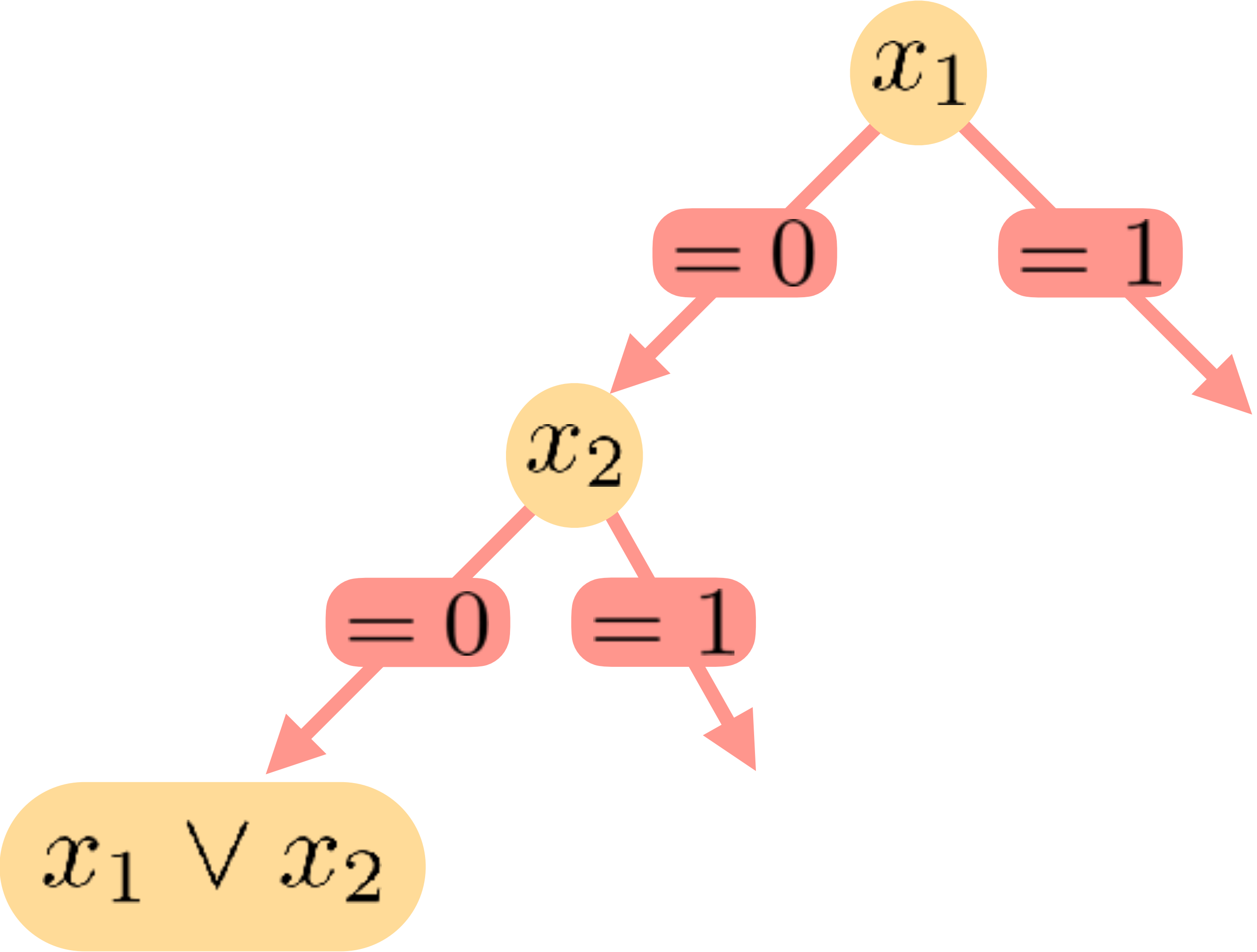
DPLL

$$\mathcal{F} = (x_1 \vee x_2) \wedge (\bar{x}_1 \vee x_2) \wedge (x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee \bar{x}_2)$$



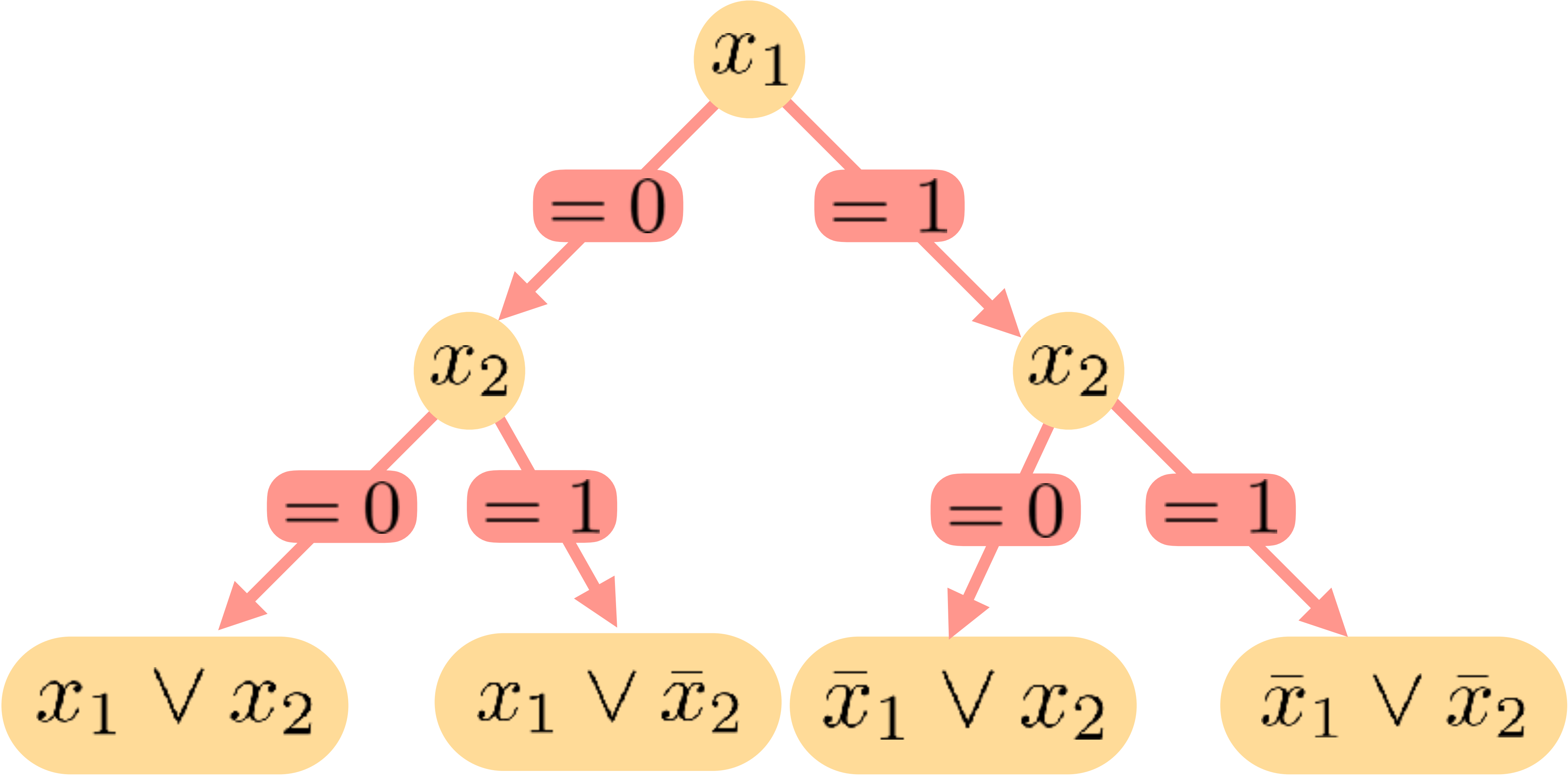
DPLL

$$\mathcal{F} = (x_1 \vee x_2) \wedge (\bar{x}_1 \vee x_2) \wedge (x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee \bar{x}_2)$$



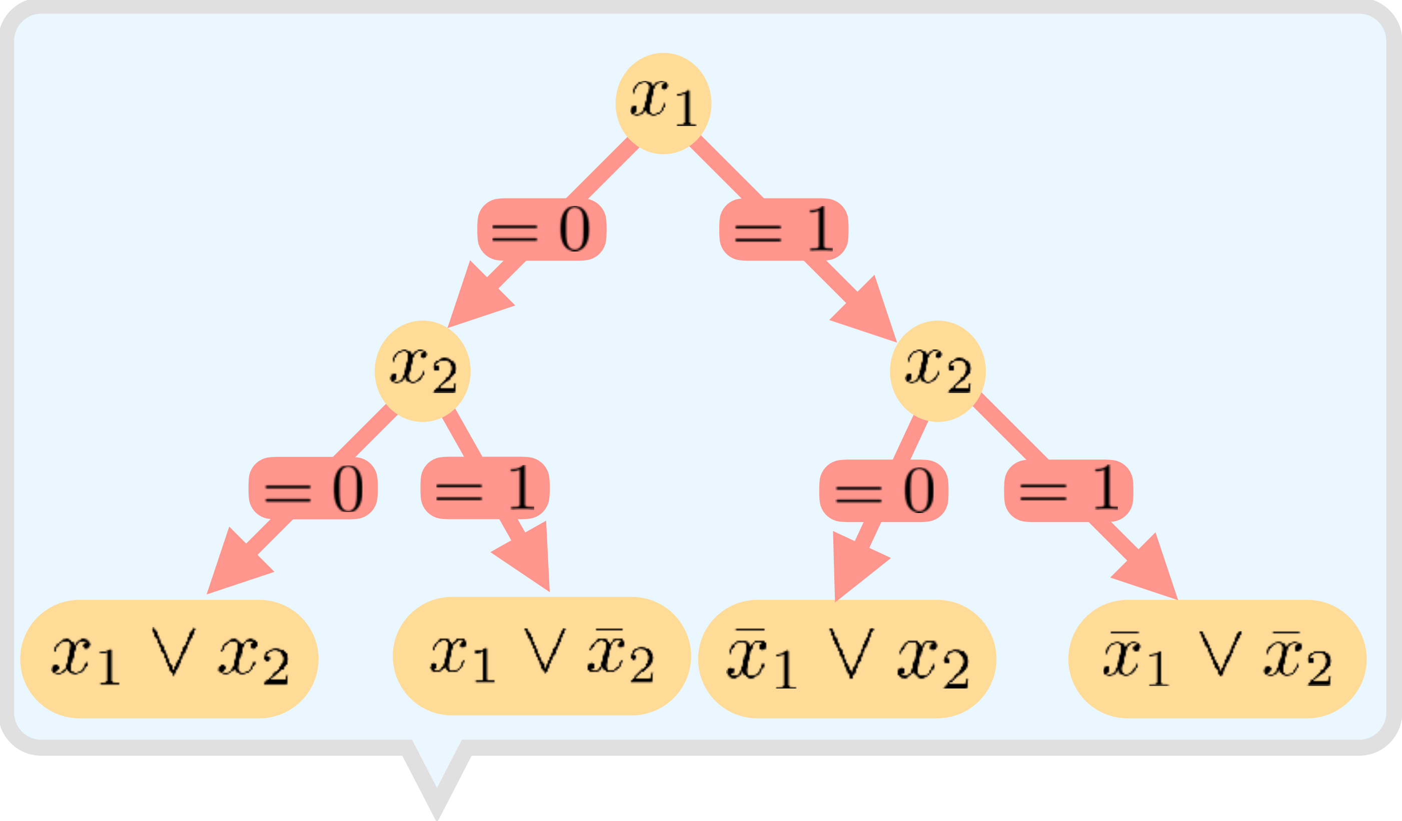
DPLL

$$\mathcal{F} = (x_1 \vee x_2) \wedge (\bar{x}_1 \vee x_2) \wedge (x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee \bar{x}_2)$$



DPLL

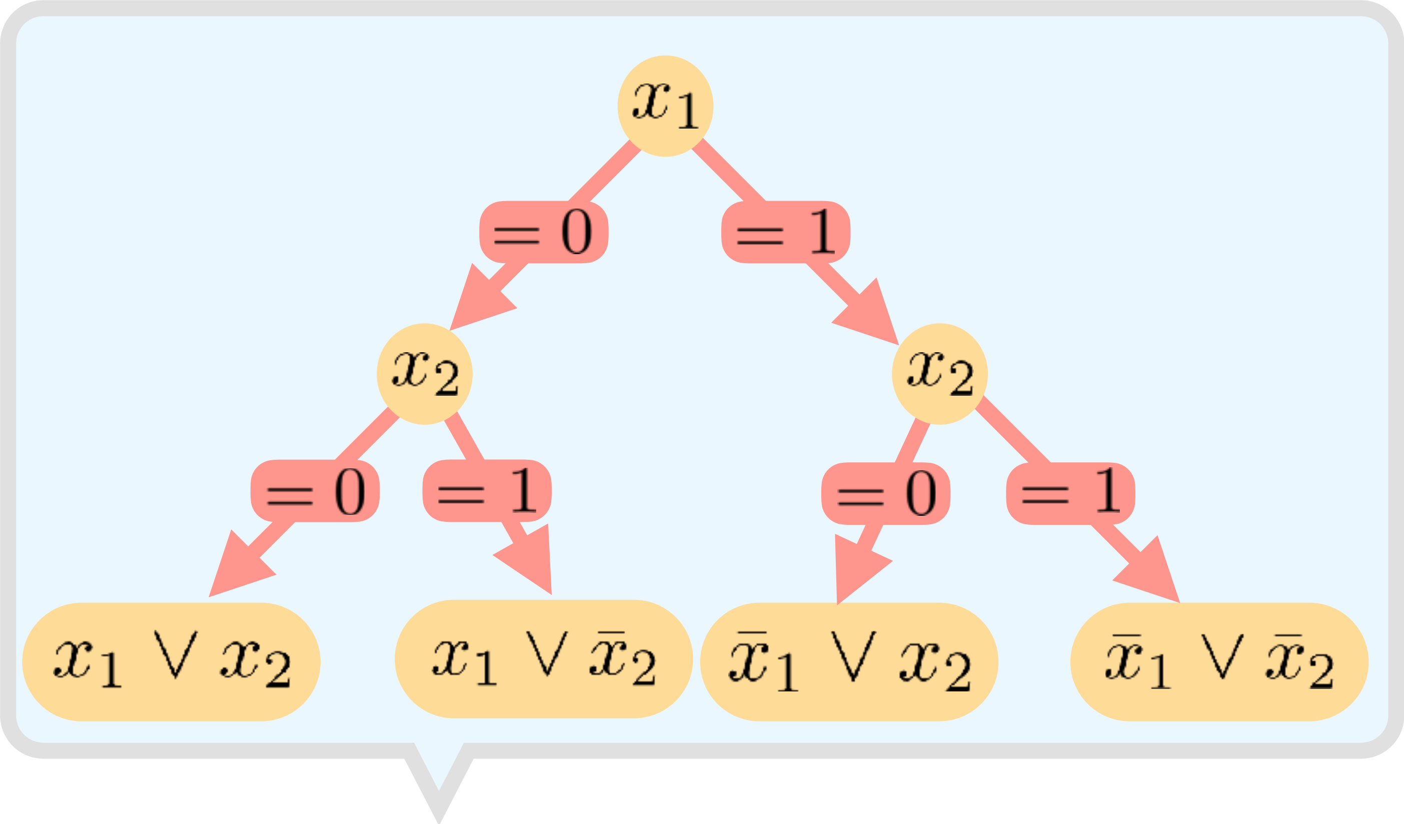
$$\mathcal{F} = (x_1 \lor x_2) \land (\bar{x}_1 \lor x_2) \land (x_1 \lor \bar{x}_2) \land (\bar{x}_1 \lor \bar{x}_2)$$



Proof of unsatisfiability!

DPLL

$$\mathcal{F} = (x_1 \vee x_2) \wedge (\bar{x}_1 \vee x_2) \wedge (x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee \bar{x}_2)$$



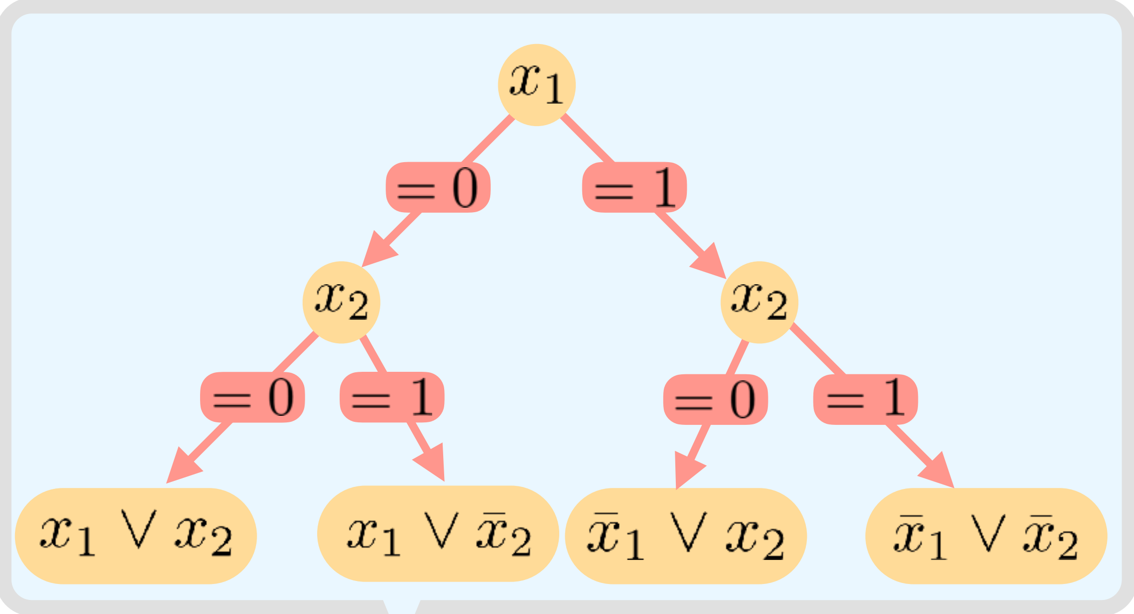
Proof of unsatisfiability!

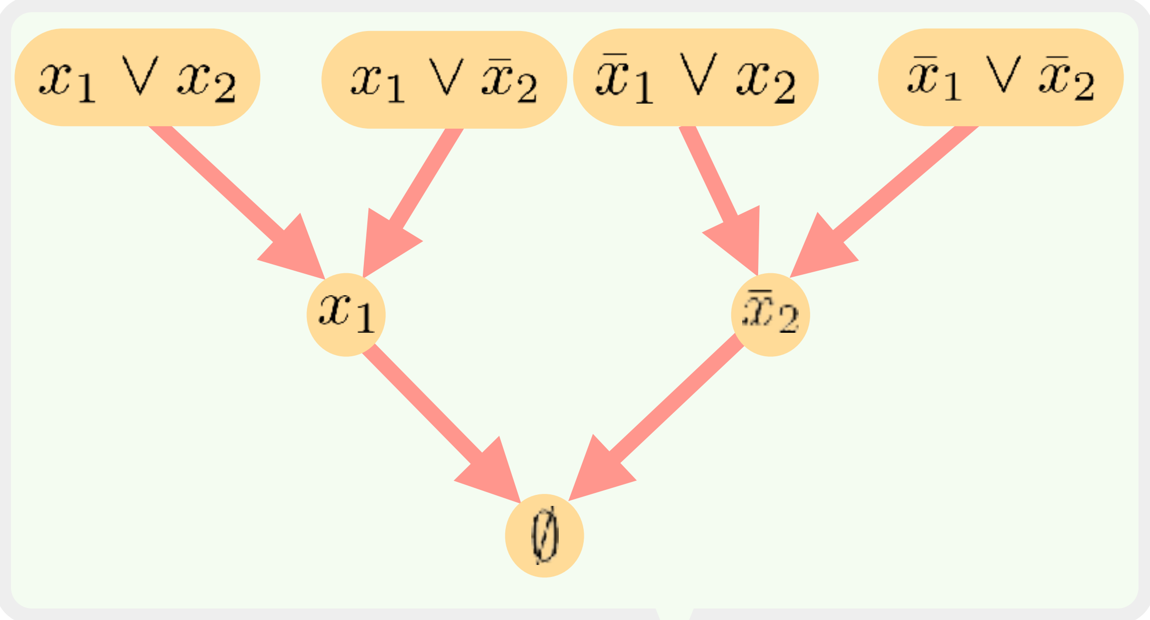
SAT solvers run on unsatisfiable CNFs output proofs.

- Proof complexity analysis applies to SAT solvers
- proof size = runtime of ideal implementation of search algorithm

DPLL

$$\mathcal{F} = (x_1 \vee x_2) \wedge (\bar{x}_1 \vee x_2) \wedge (x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee \bar{x}_2)$$





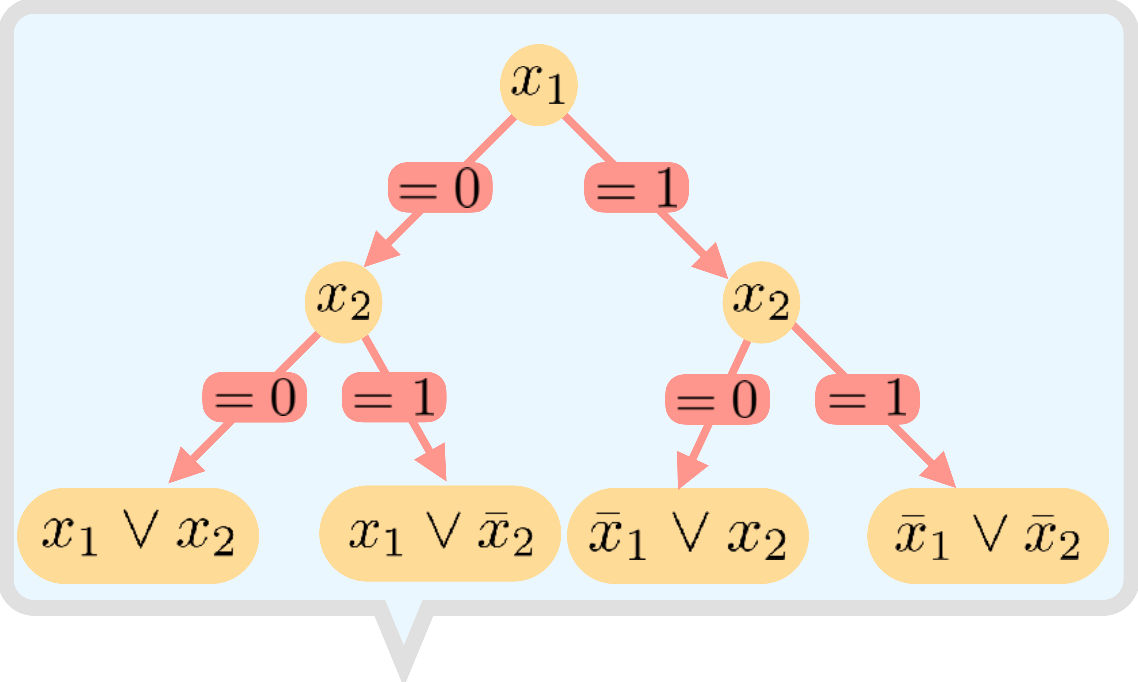
DPLL proofs = *tree-like* Resolution proofs

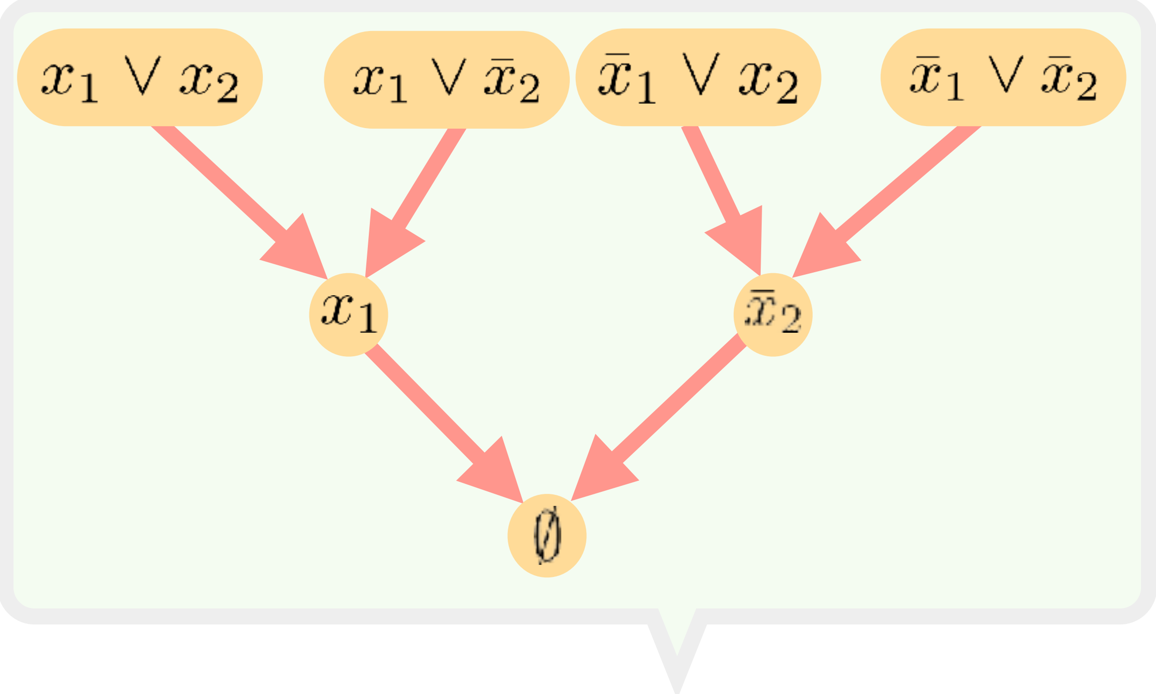
Query based

Rule based

DPLL

$$\mathcal{F} = (x_1 \vee x_2) \wedge (\bar{x}_1 \vee x_2) \wedge (x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee \bar{x}_2)$$





DPLL proofs = *tree-like* Resolution proofs

Query based

Rule based

CDCL augments DPLL with heuristics, clause learning, restarts, etc.

- Proofs still captured by Resolution
  - Weak proof system, cannot count

PseudoBoolean SAT Solvers

Reason about integer-linear inequalities rather than clauses

Most are based on Cutting Planes proof system

- stronger proof system than Resolution

Worse performance than state-of-the-art solvers based on DPLL

Puzzle

Why is can’t we develop good search algorithms based on stronger proof systems?

Many strong proof systems for which we can theoretically find proofs quickly.

- e.g. *polynomial calculus*

Best SAT algorithms based on DPLL

- DPLL can’t even count, no gaussian elimination!

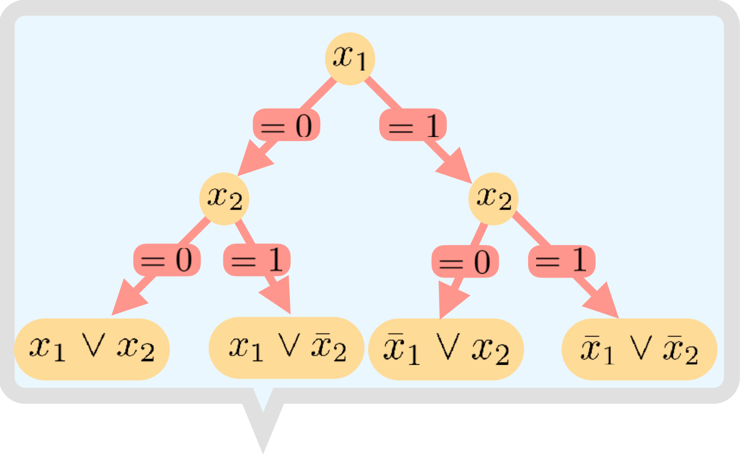
Puzzle

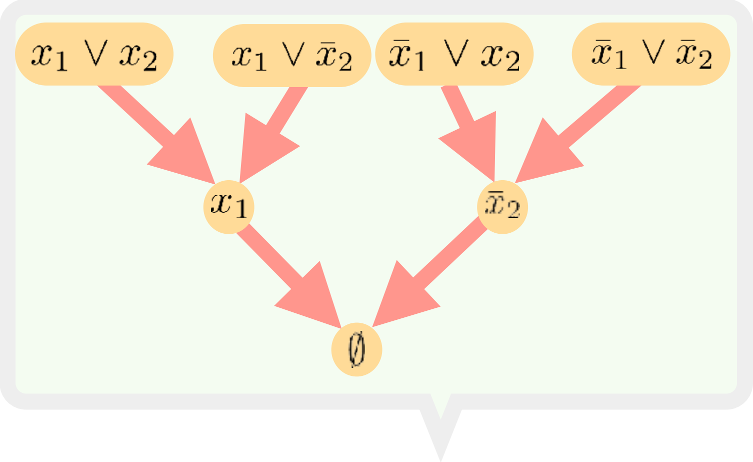
Why is can’t we develop good search algorithms based on stronger proof systems?

Hypothesis:

Querying is more conducive to search algorithms

- Leads to simple divide-and-conquer style algorithms
- DPLL vs tree-like Resolution





Stabbing Planes

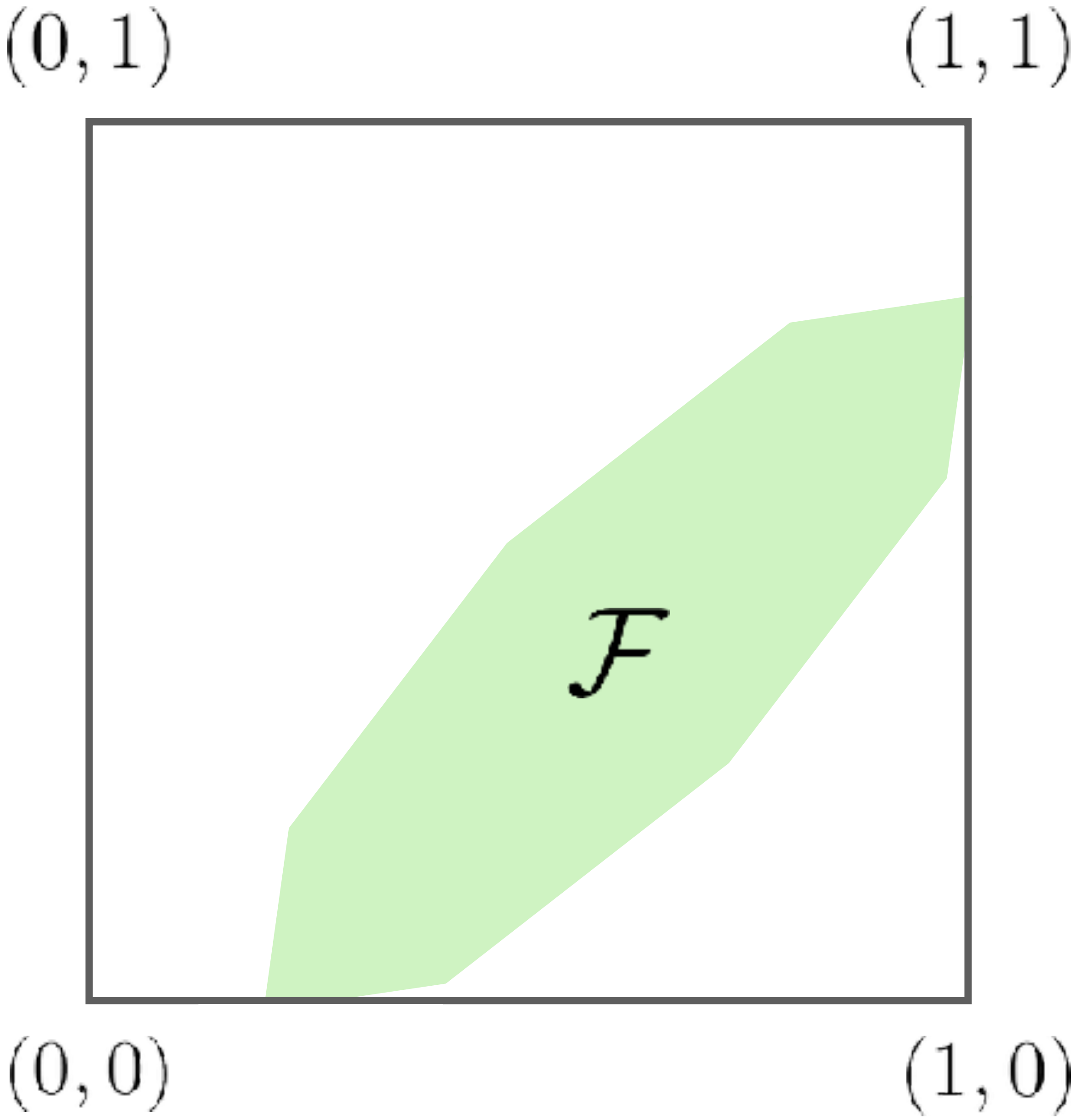
Generalization of DPLL to reason about integer-linear inequalities, formalized as a proof system

Stabbing Planes

$\mathcal{F} = \{\text{Unsatisfiable set of ineteger-linear inequalities}\}$

Variables $x, y \in [0, 1]$

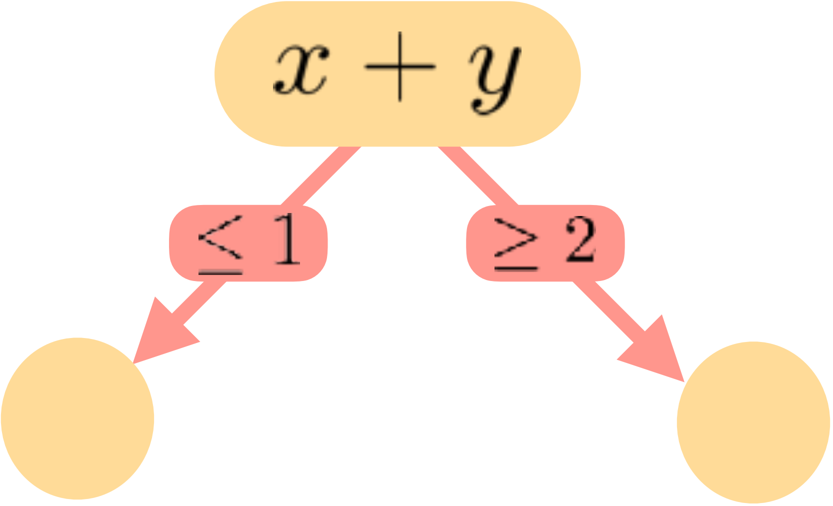
over $\{0, 1\}$ assignments

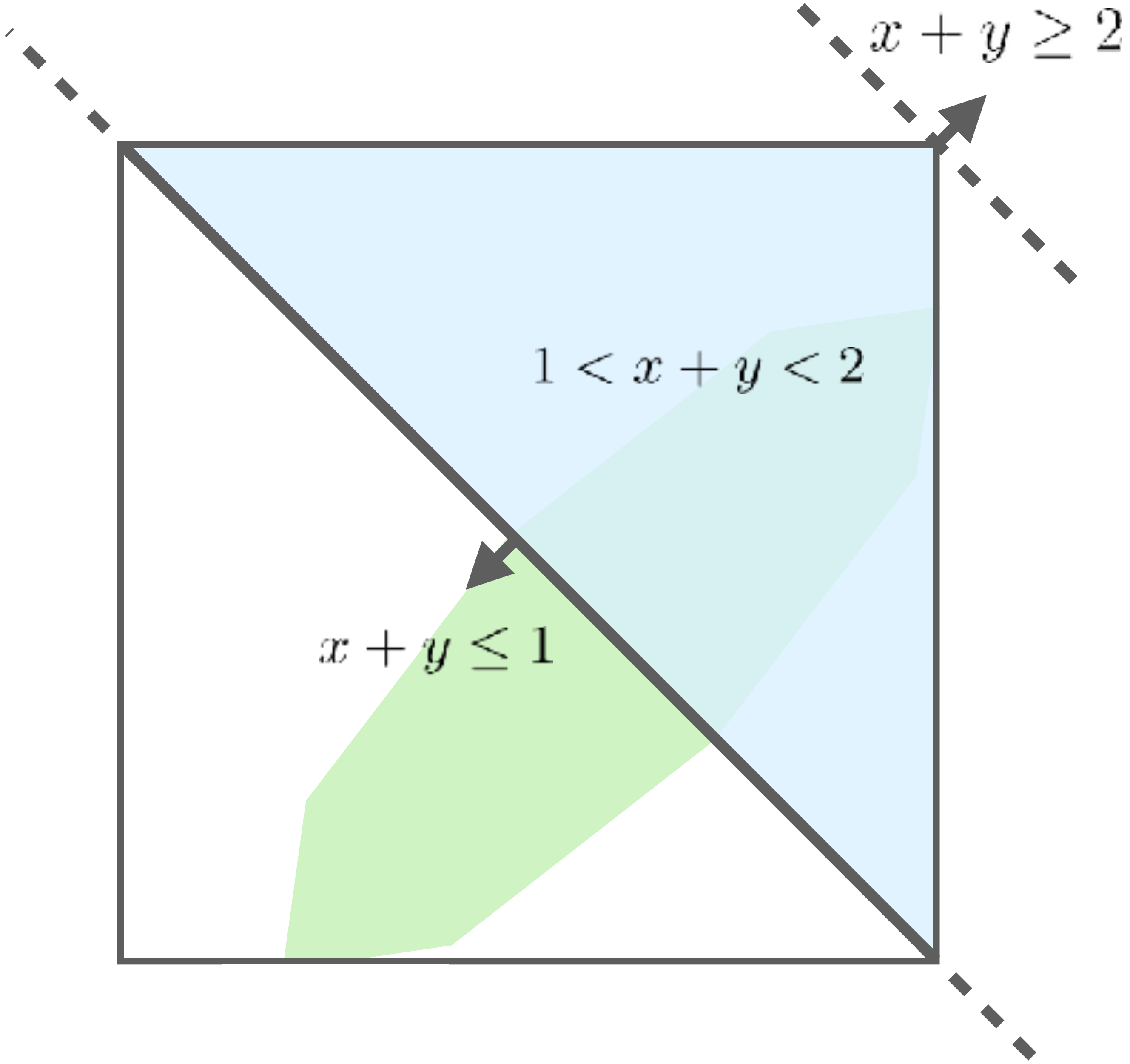


Stabbing Planes

$\mathcal{F} = \{\text{unsatisfiable set of linear inequalities}\}$

Variables $x, y \in [0, 1]$

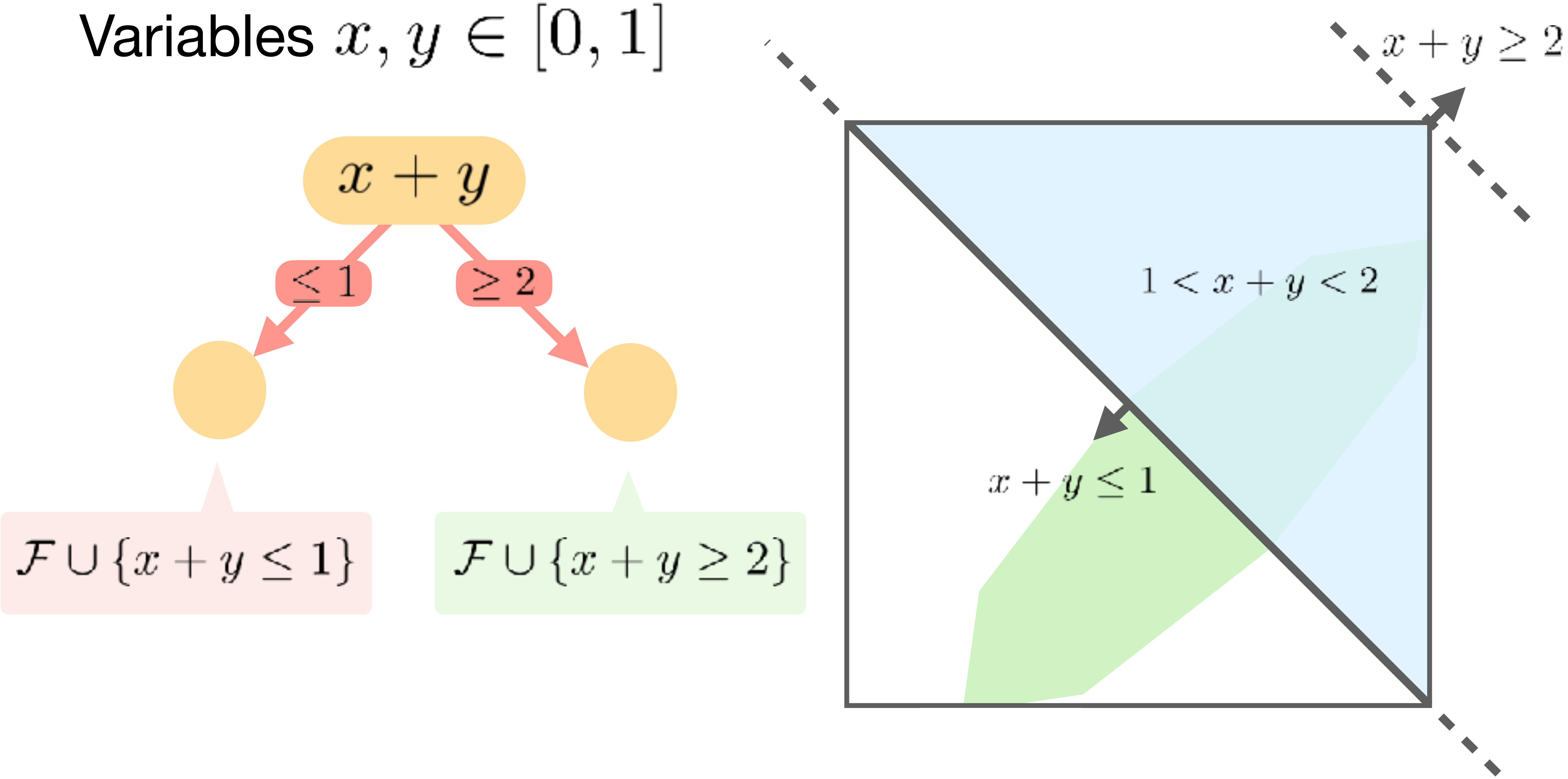




Stabbing Planes

$\mathcal{F} = \{\text{unsatisfiable set of linear inequalities}\}$

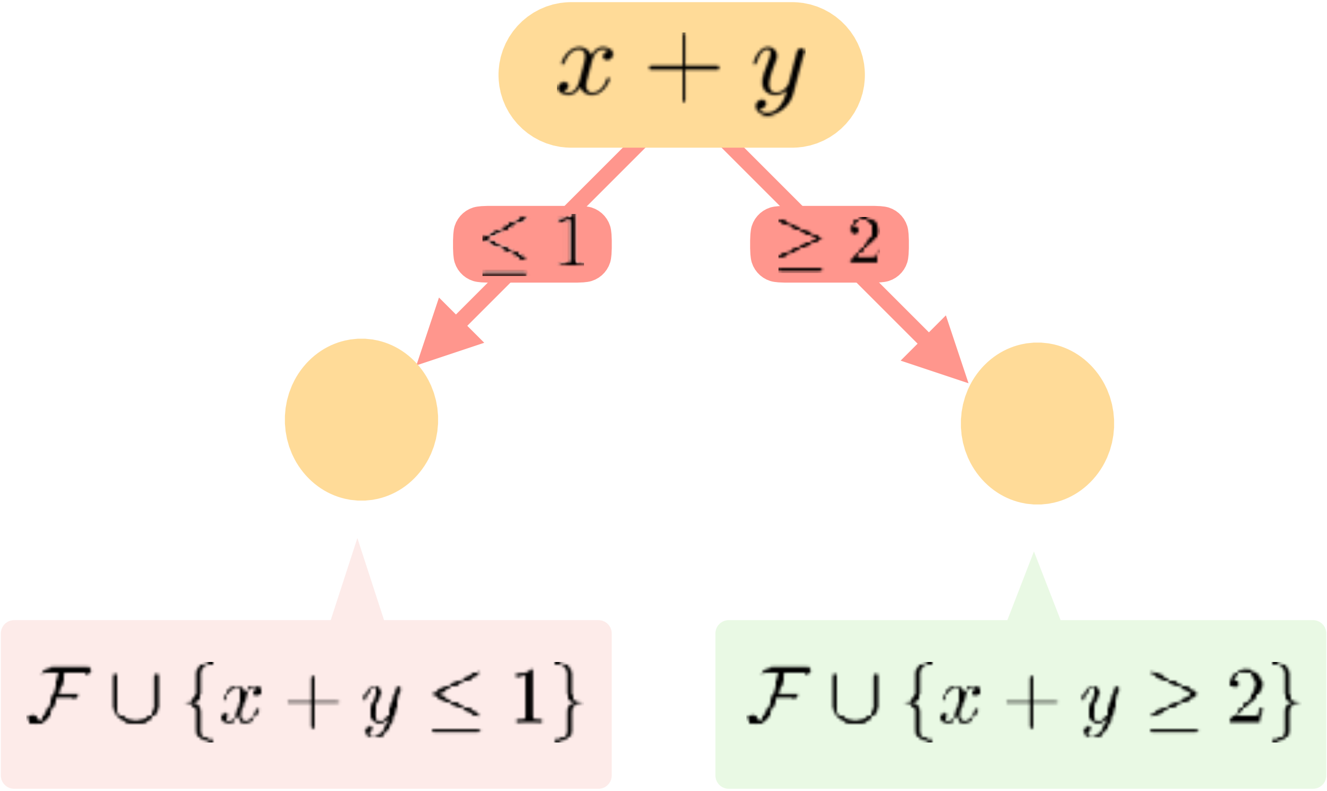
Variables $x, y \in [0, 1]$

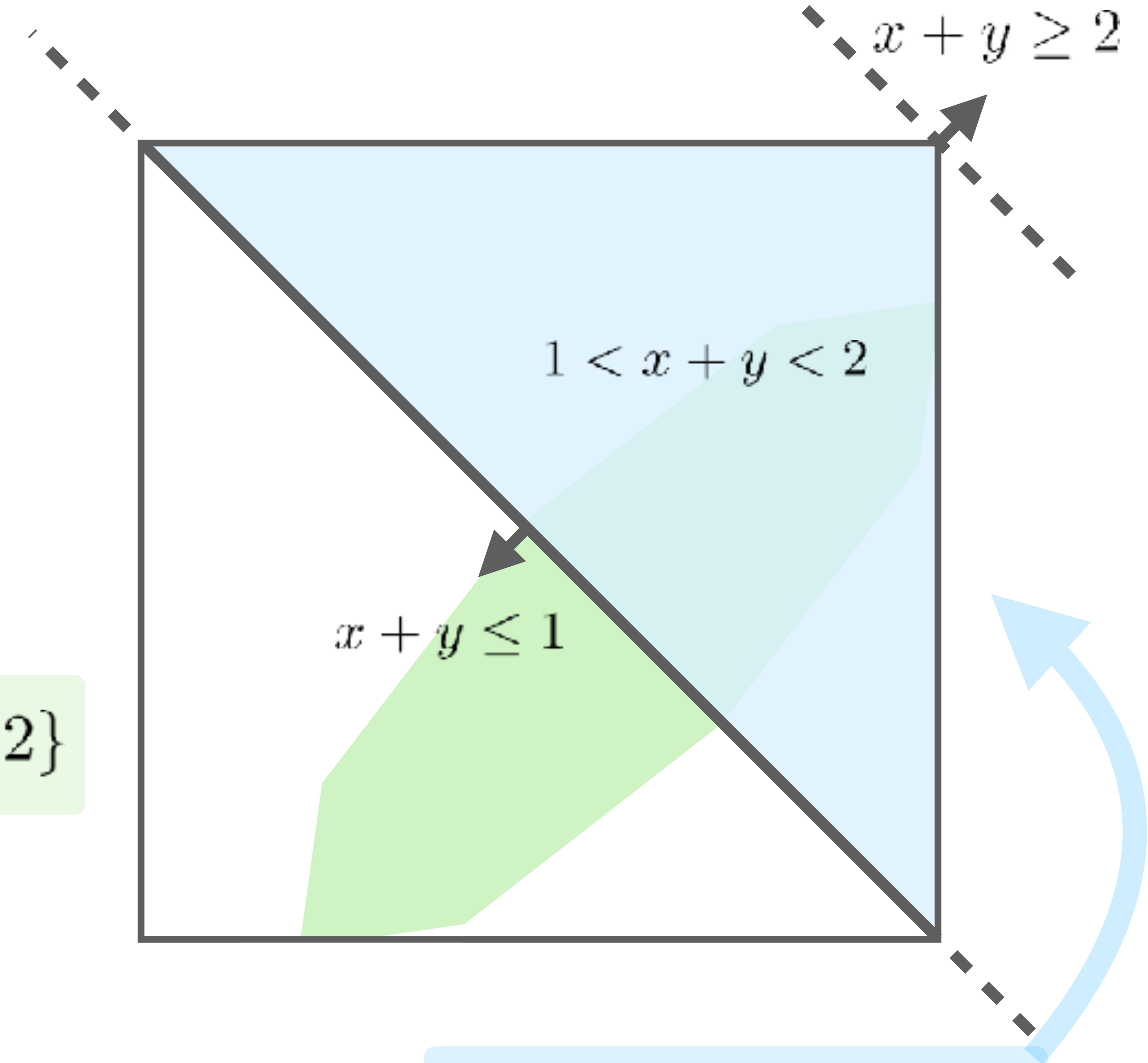


Stabbing Planes

$\mathcal{F} = \{\text{unsatisfiable set of linear inequalities}\}$

Variables $x, y \in [0, 1]$



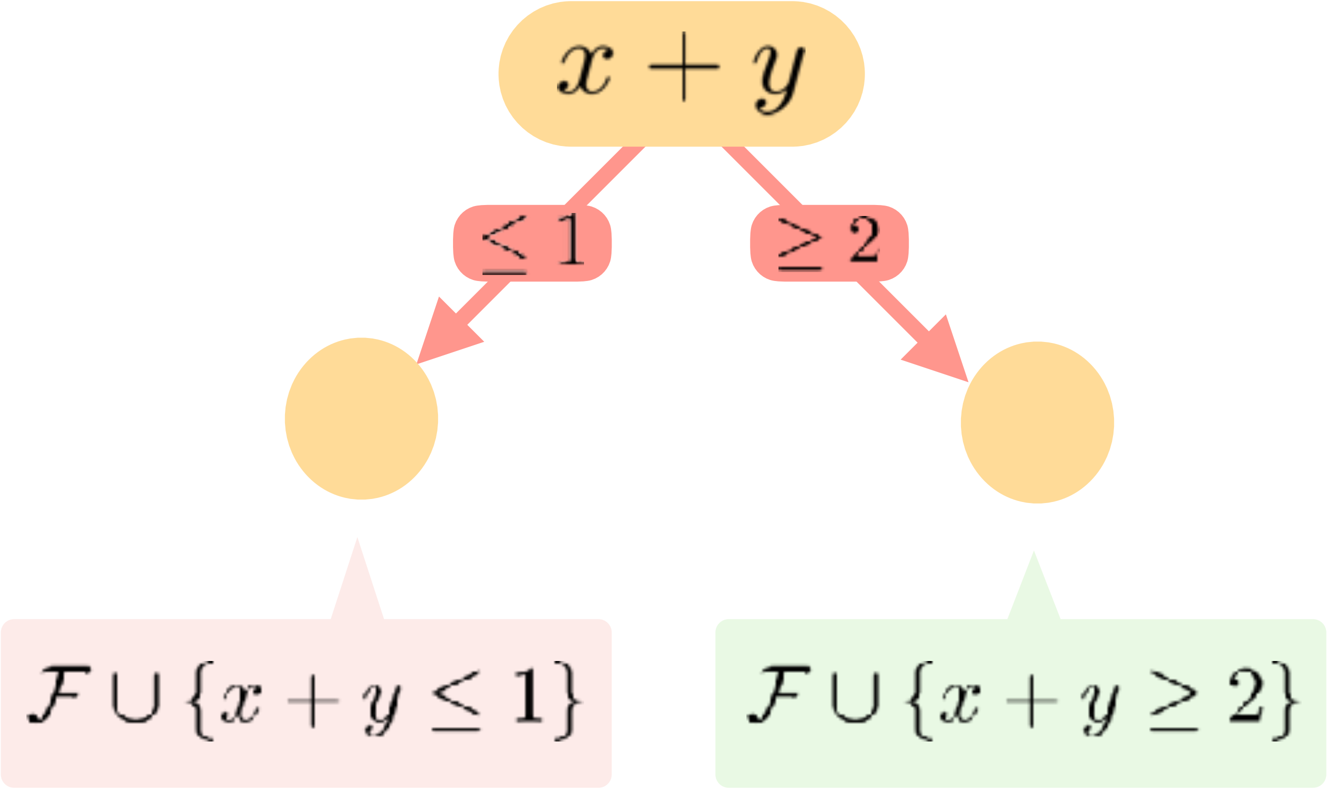


$1 < x + y < 2$ slab removed

Stabbing Planes

$\mathcal{F} = \{\text{unsatisfiable set of linear inequalities}\}$

Variables $x, y \in [0, 1]$



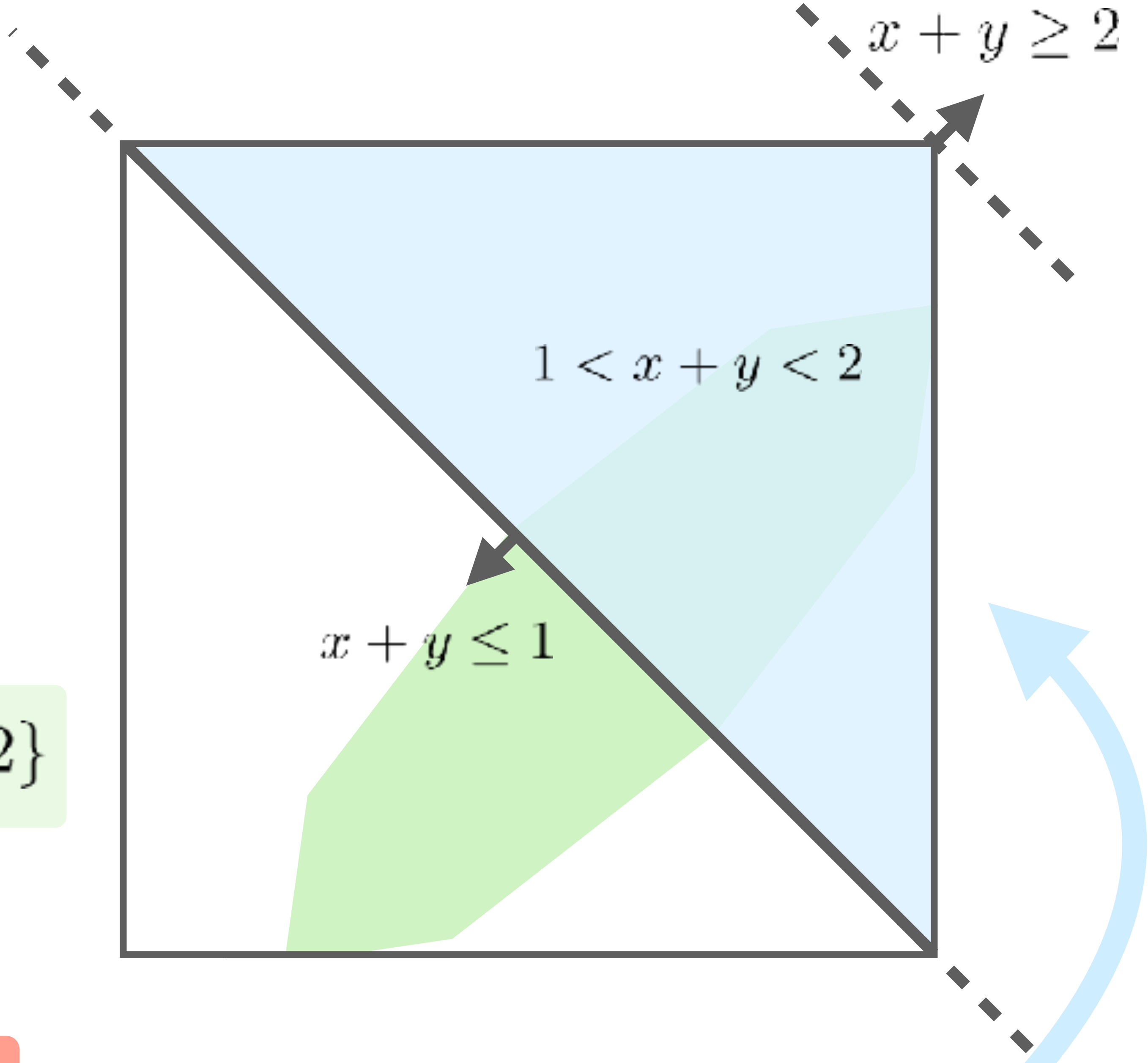
$x + y$

≤ 1

≥ 2

$\mathcal{F} \cup \{x + y \leq 1\}$

$\mathcal{F} \cup \{x + y \geq 2\}$



Can't remove $\{0, 1\}$ points!

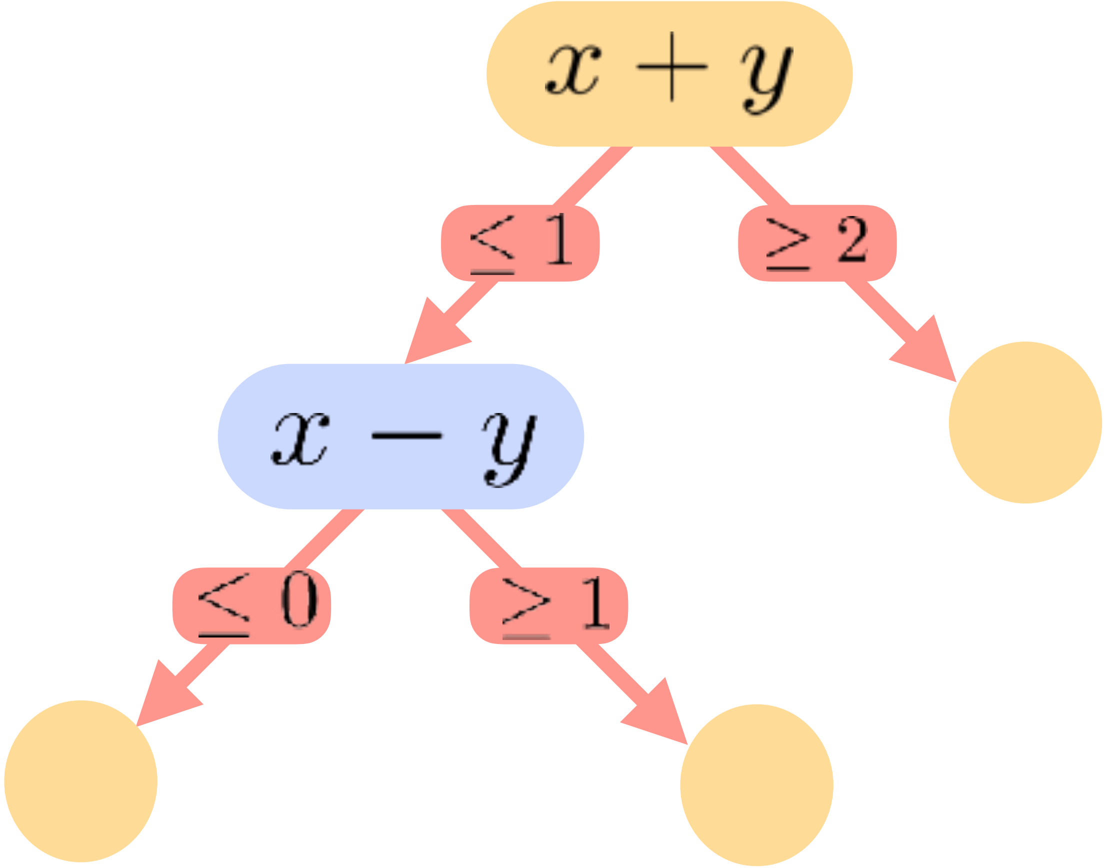
Any $\alpha \in \{0,1\}^n$ satisfies $Ax \geq b$ or $Ax \leq b - 1$ for $A \in \mathbb{Z}^n, b \in \mathbb{Z}$

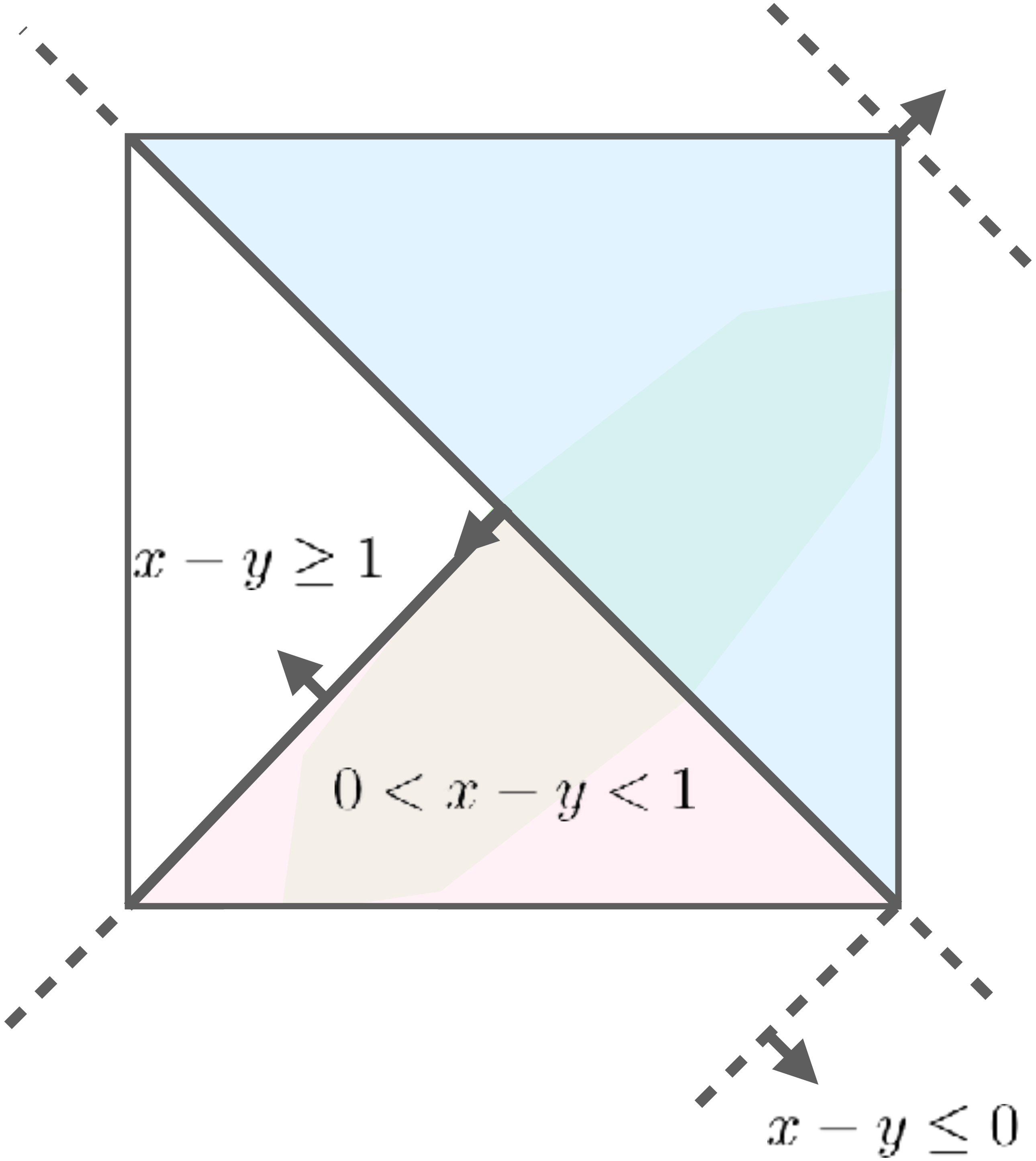
$1 < x + y < 2$ slab removed

Stabbing Planes

$\mathcal{F} = \{\text{unsatisfiable set of linear inequalities}\}$

Variables $x, y \in [0,1]$

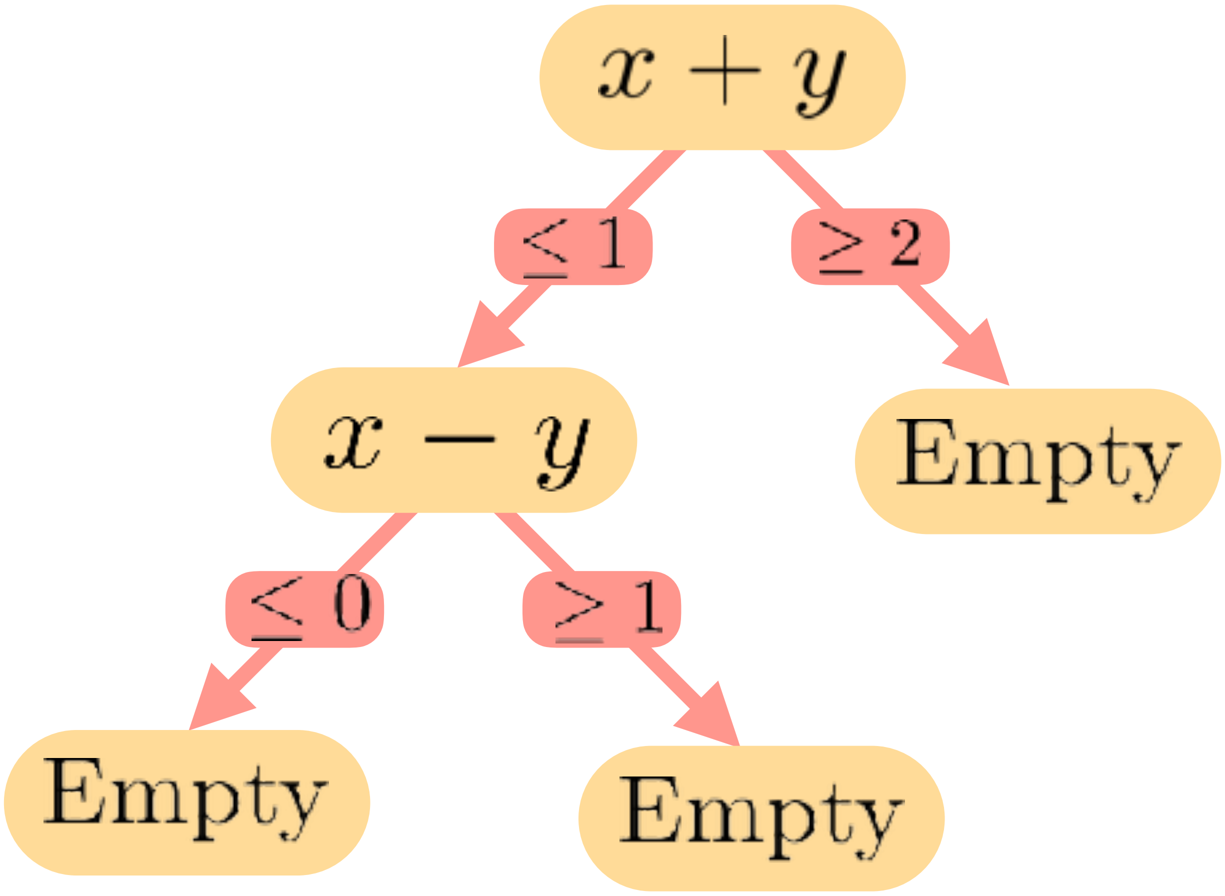


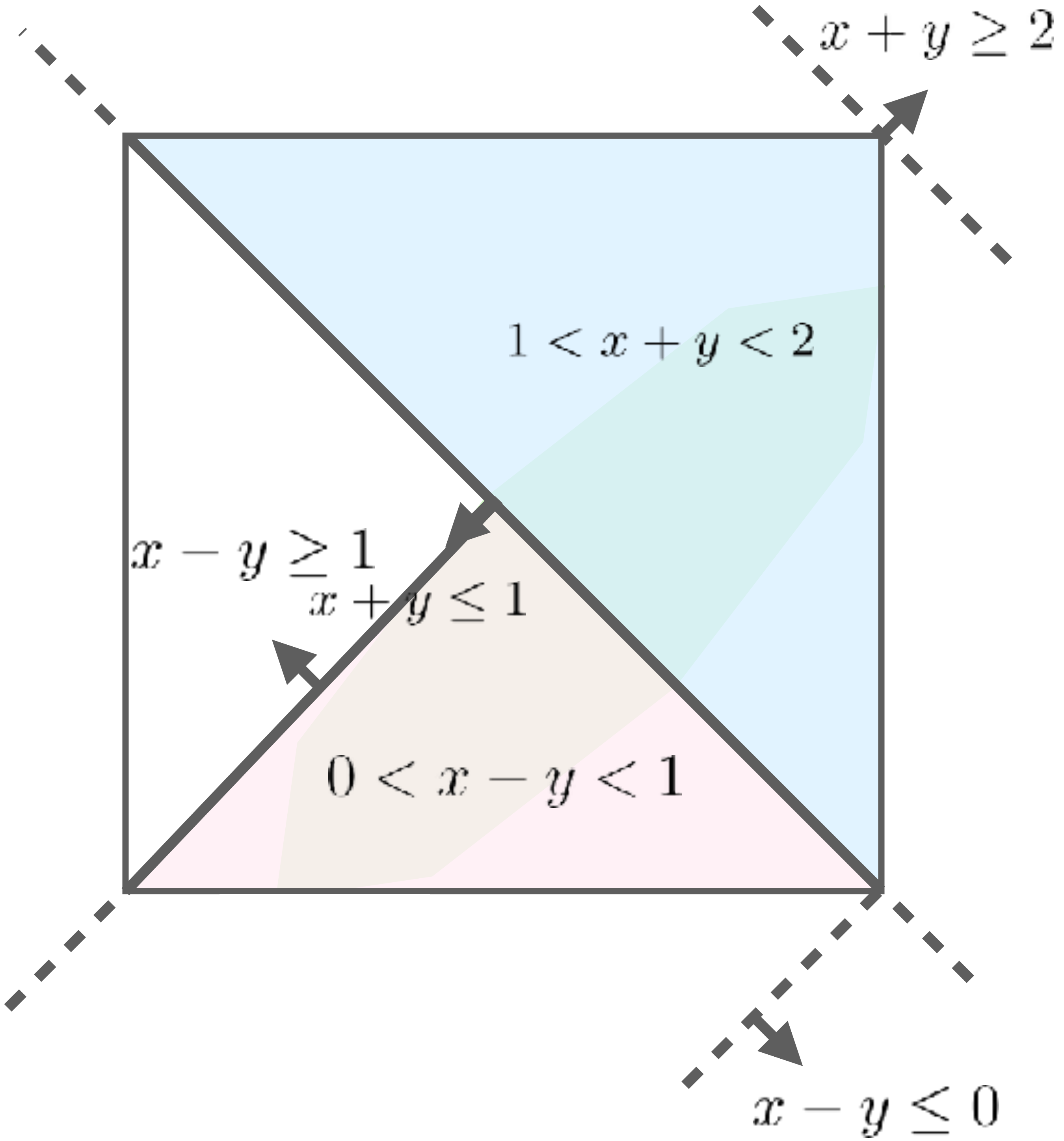


Stabbing Planes

$\mathcal{F} = \{\text{unsatisfiable set of linear inequalities}\}$

Variables $x, y \in [0, 1]$

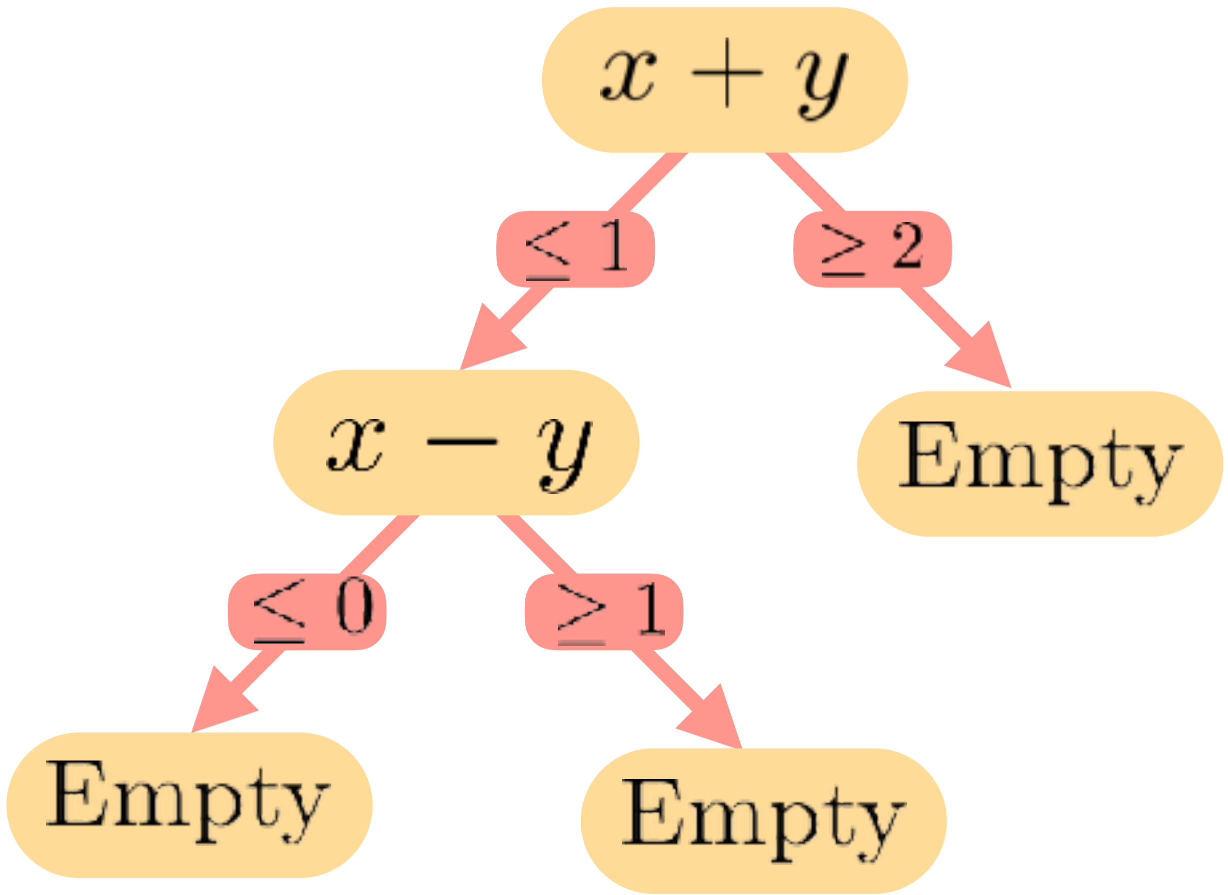


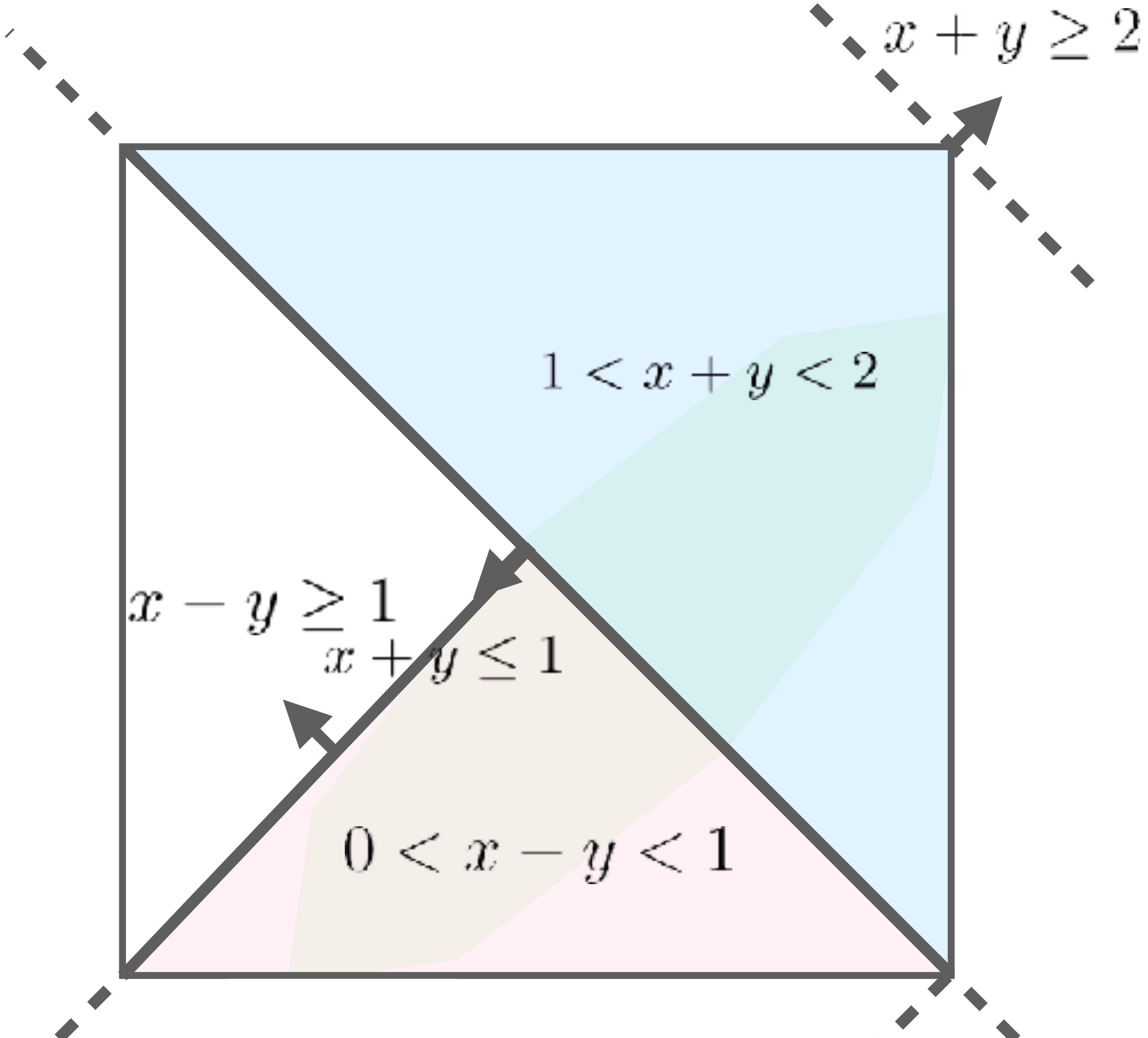


Stabbing Planes

$\mathcal{F} = \{\text{unsatisfiable set of linear inequalities}\}$

Variables $x, y \in [0, 1]$



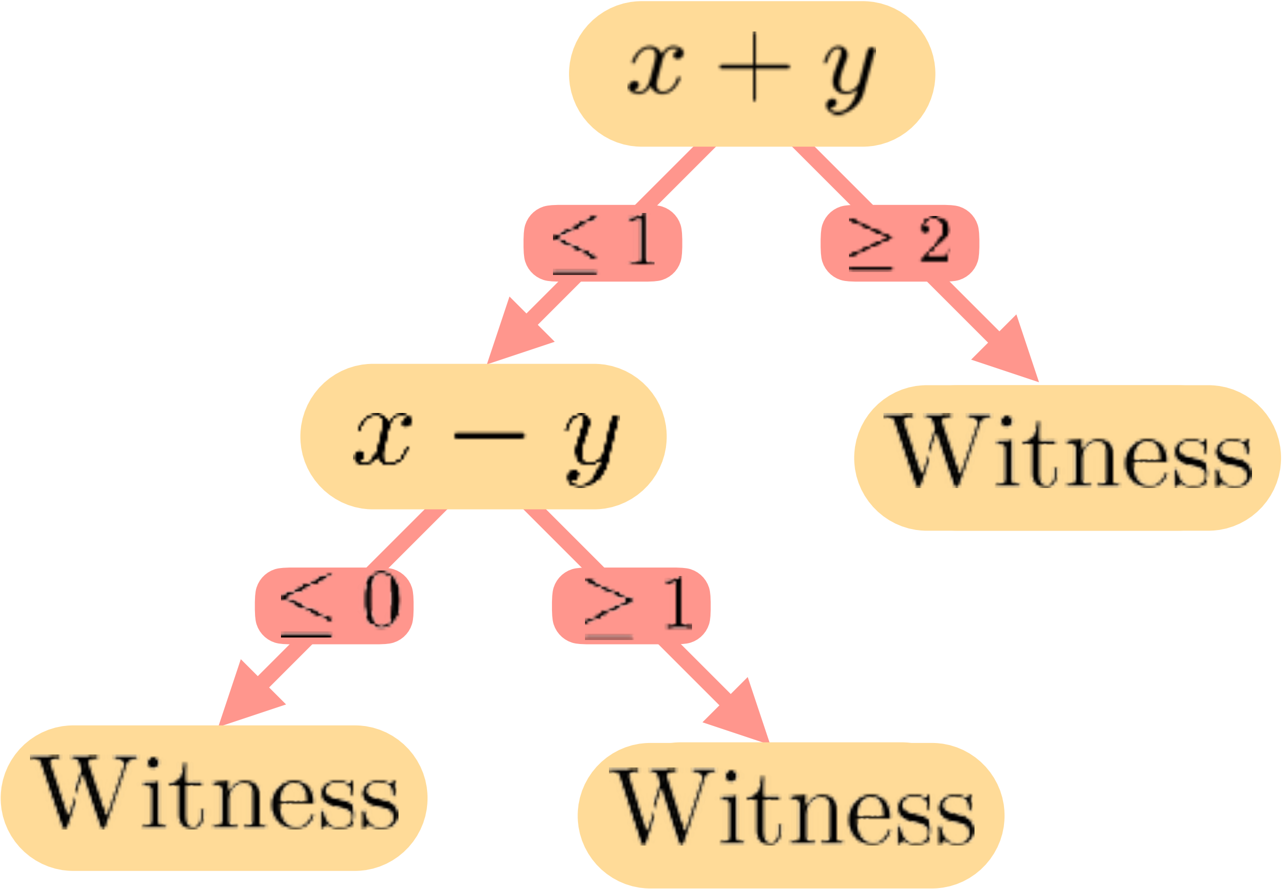


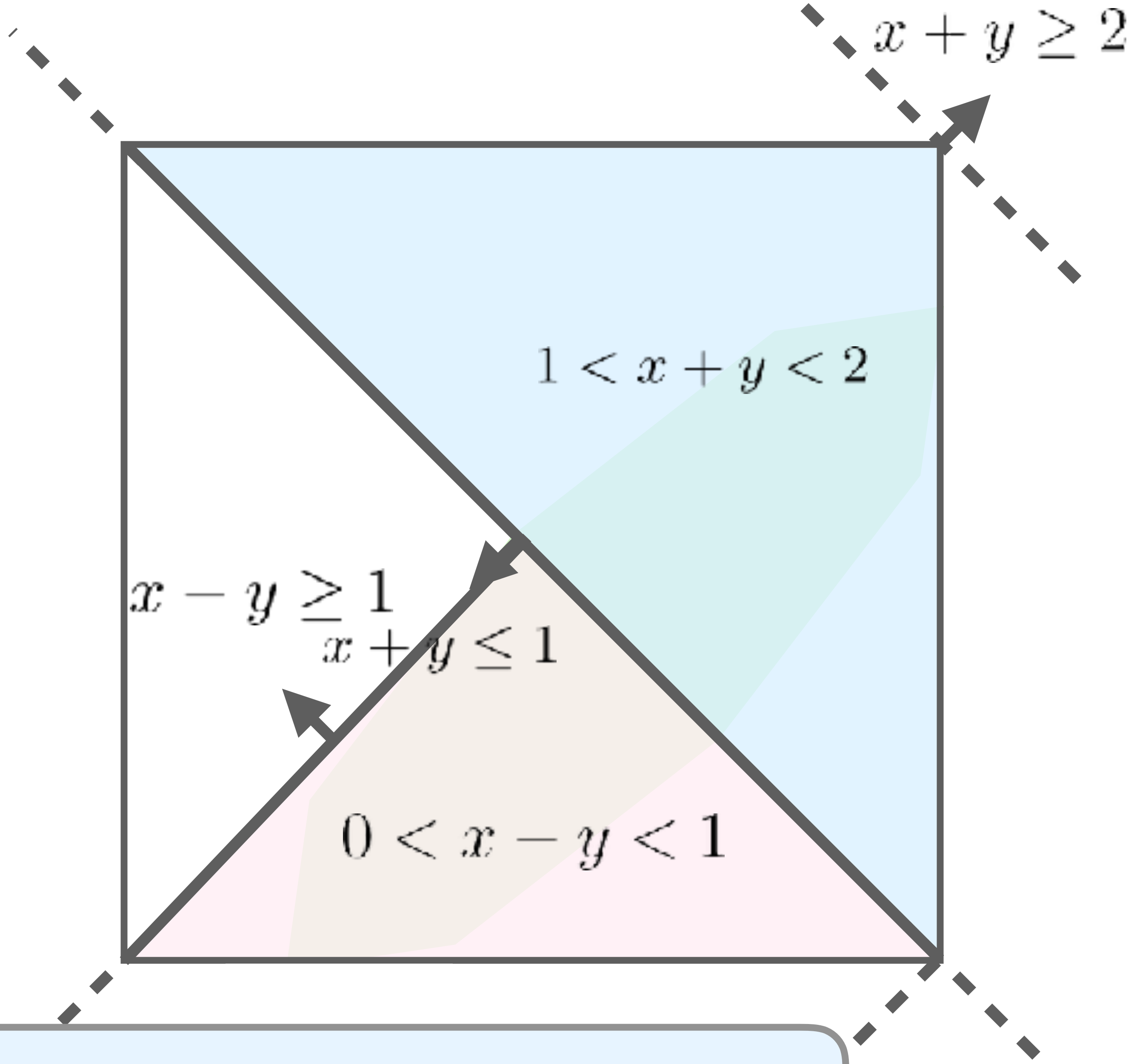
Farkas' Lemma: Polytope is empty iff there is a non-negative linear combination of its constraints equalling $0 \geq 1$

Stabbing Planes

$\mathcal{F} = \{\text{unsatisfiable set of linear inequalities}\}$

Variables $x, y \in [0, 1]$



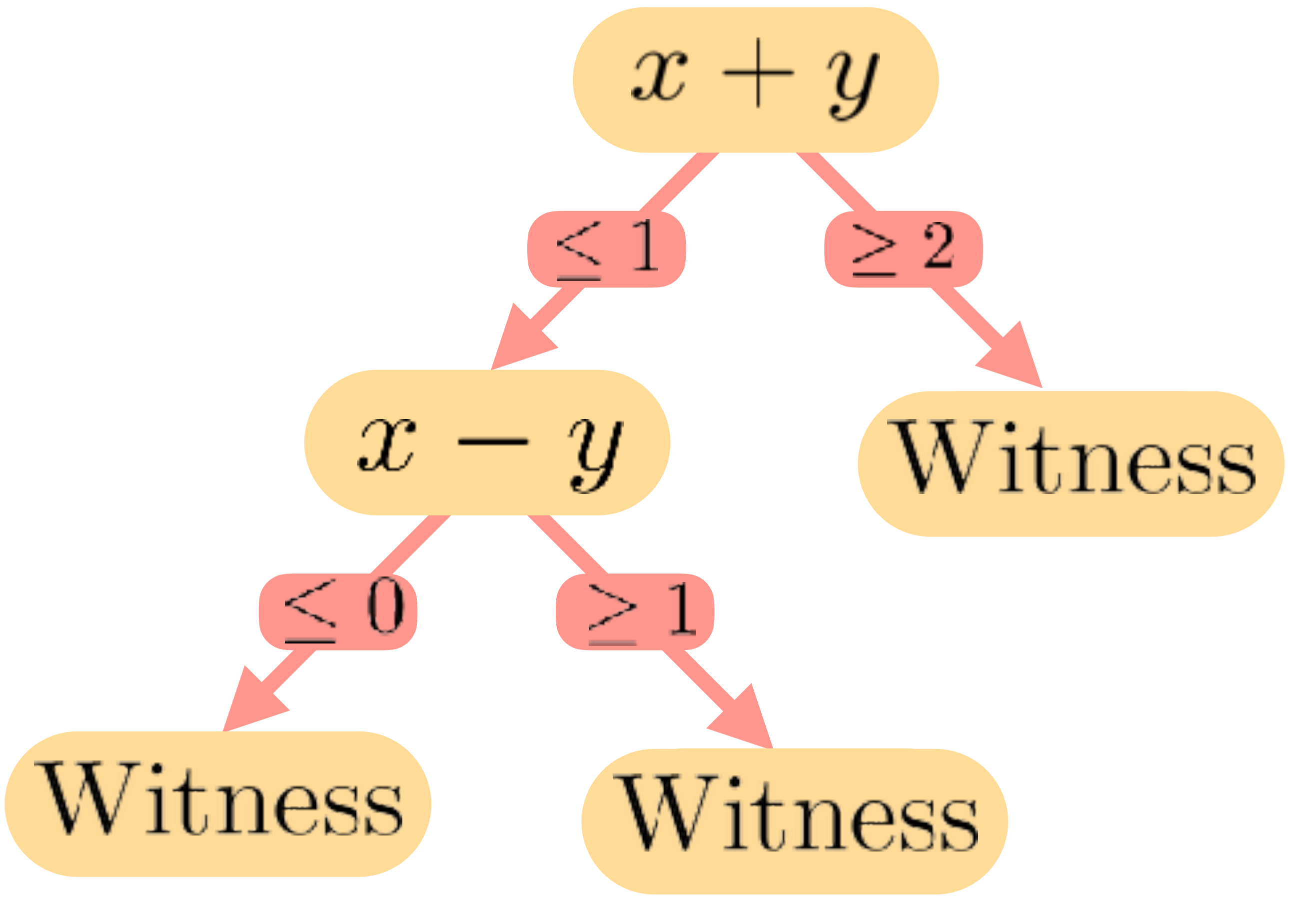


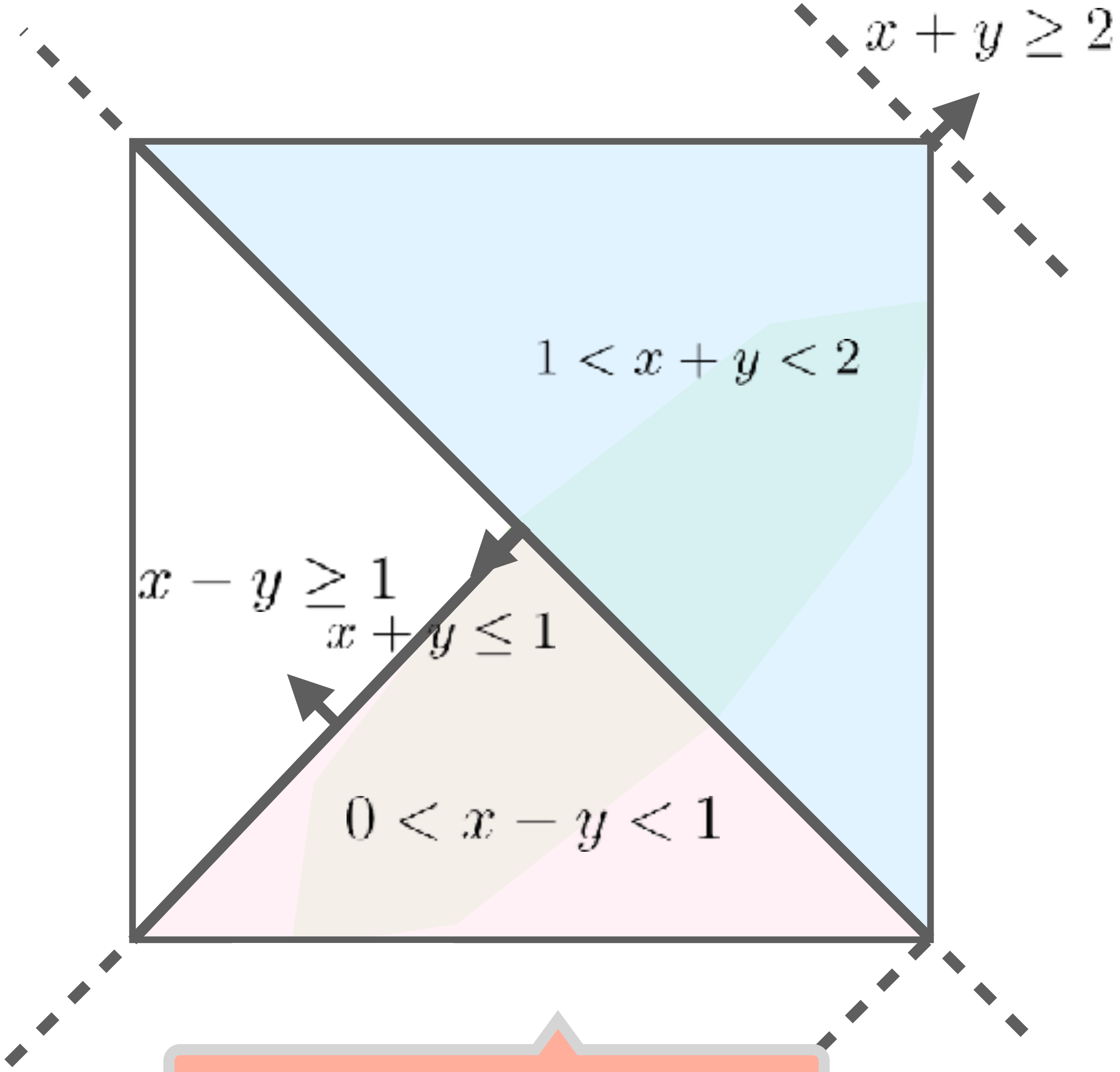
Witness: Non-negative linear combination of inequalities in $\mathcal{F}$ and constraints along path to this node equalling $0 \geq 1$

Stabbing Planes

$$\mathcal{F} = \{\text{unsatisfiable set of linear inequalities}\}$$

Variables $x, y \in [0, 1]$





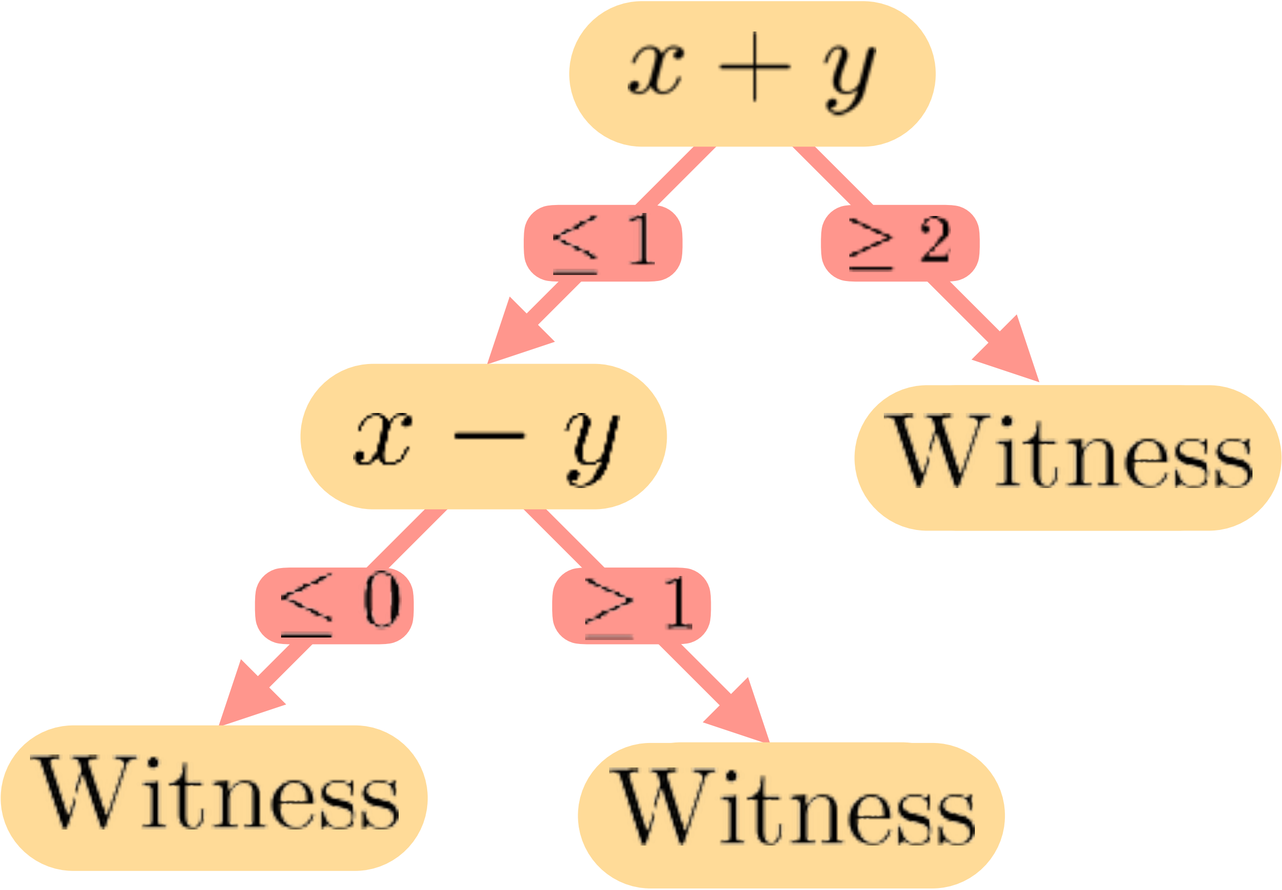
Algebraic proof

Geometric proof

Stabbing Planes

$\mathcal{F} = \{\text{unsatisfiable set of linear inequalities}\}$

Variables $x, y \in [0, 1]$



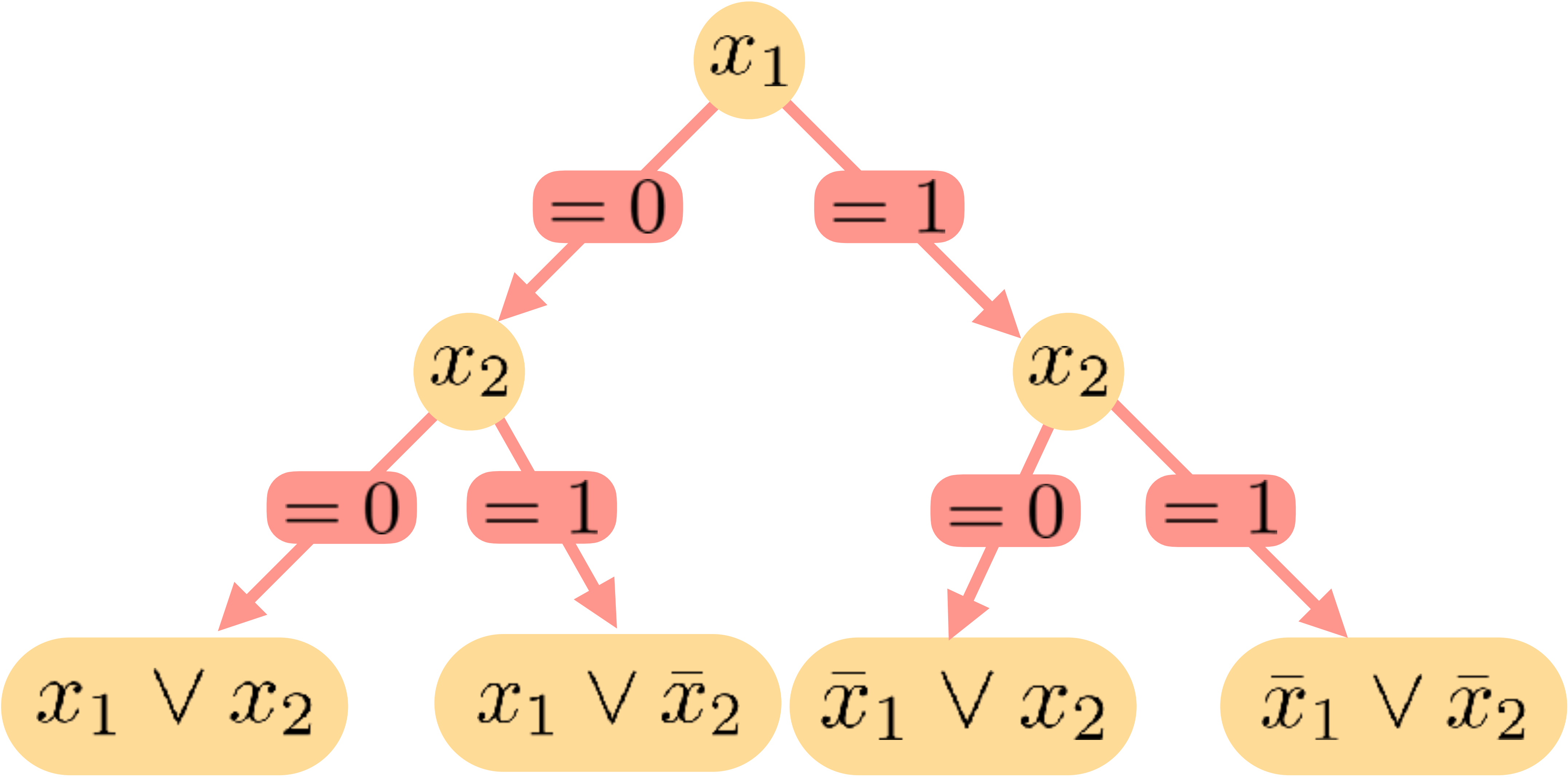
<u>Complexity measures</u>

Size: Number of nodes

Depth: Tree-depth

SP Generalizes DPLL

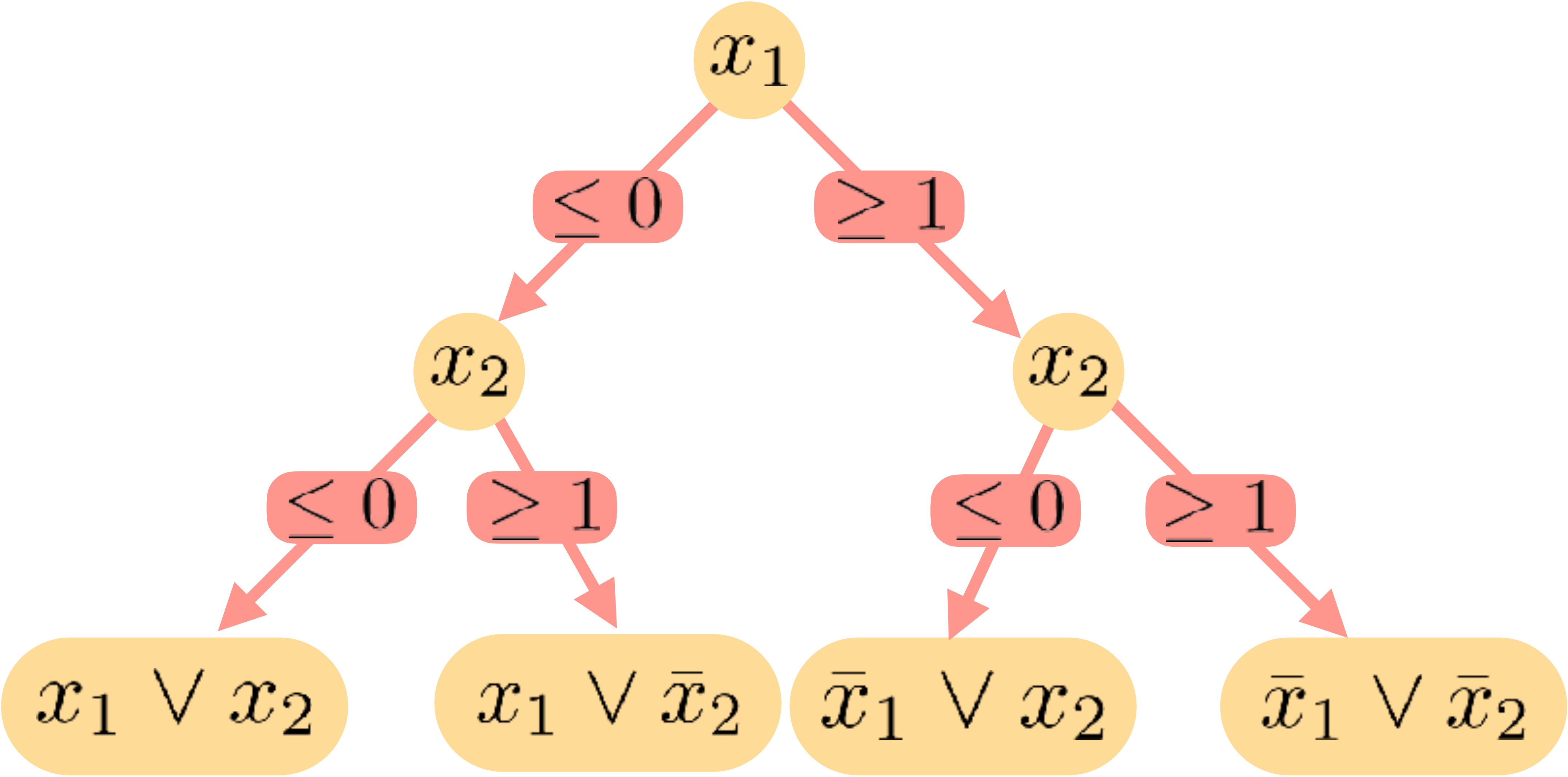
$$\mathcal{F} = (x_1 \vee x_2) \wedge (\bar{x}_1 \vee x_2) \wedge (x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee \bar{x}_2)$$



SP Generalizes DPLL

$$\mathcal{F} = \{x_1 + x_2 \geq 1, x_1 - x_2 \geq 0, x_2 - x_1 \geq 0, -x_1 - x_2 \geq -1\}$$

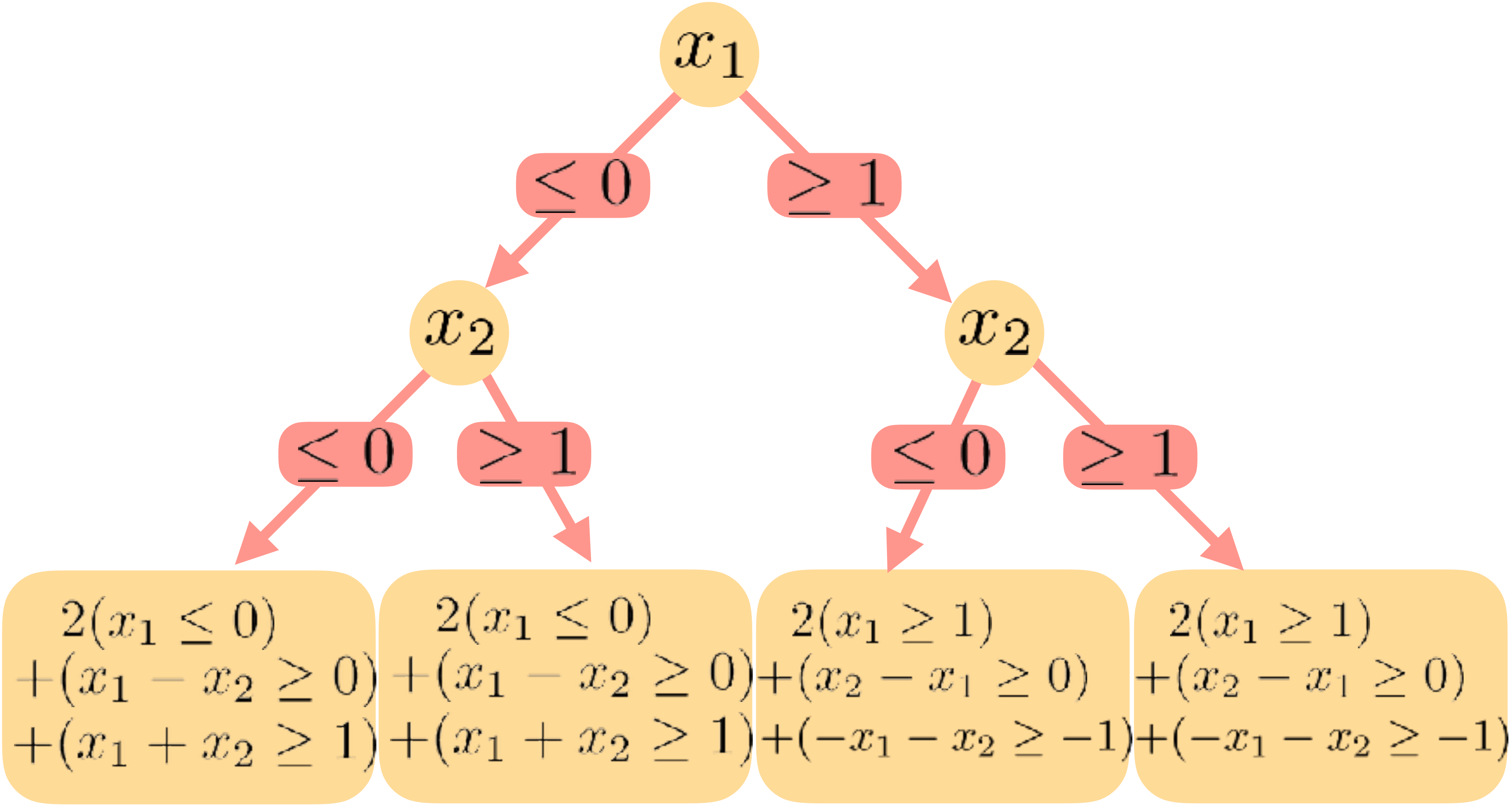
Variables $x_1, x_2 \in [0, 1]$



SP Generalizes DPLL

$$\mathcal{F} = \{x_1 + x_2 \geq 1, x_1 - x_2 \geq 0, x_2 - x_1 \geq 0, -x_1 - x_2 \geq -1\}$$

Variables $x_1, x_2 \in [0, 1]$



$$2(x_1 \leq 0) + (x_1 - x_2 \geq 0) + (x_1 + x_2 \geq 1)$$

$$2(x_1 \leq 0) + (x_1 - x_2 \geq 0) + (x_1 + x_2 \geq 1)$$

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

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

Stabbing Planes

Polynomially equivalent to a *tree-like* variant of the R(CP) proof system introduced in [Krajíček98]

- R(CP) - rule based
- Stabbing Planes - query based

Query-based viewpoint valuable for upper bounds.

Theorem: Quasi-polynomial size SP proof of any system of linear equations over a finite field

Proof Sketch

Input: Unsatisfiable system of $\mathrm{mod}\ 2$ linear equations

over $\{0, 1\}$ assignments

$$\begin{array}{c|ccccccc} & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \\ \hline C_1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ C_2 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ C_3 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ C_4 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ C_5 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ C_6 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ C_7 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \end{array} = \begin{array}{c} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{array} \mod 2$$

Proof Sketch

Input: Unsatisfiable system of $\mathrm{mod}\ 2$ linear equations
=>Exists subset which sum to 0 = 1 mod 2

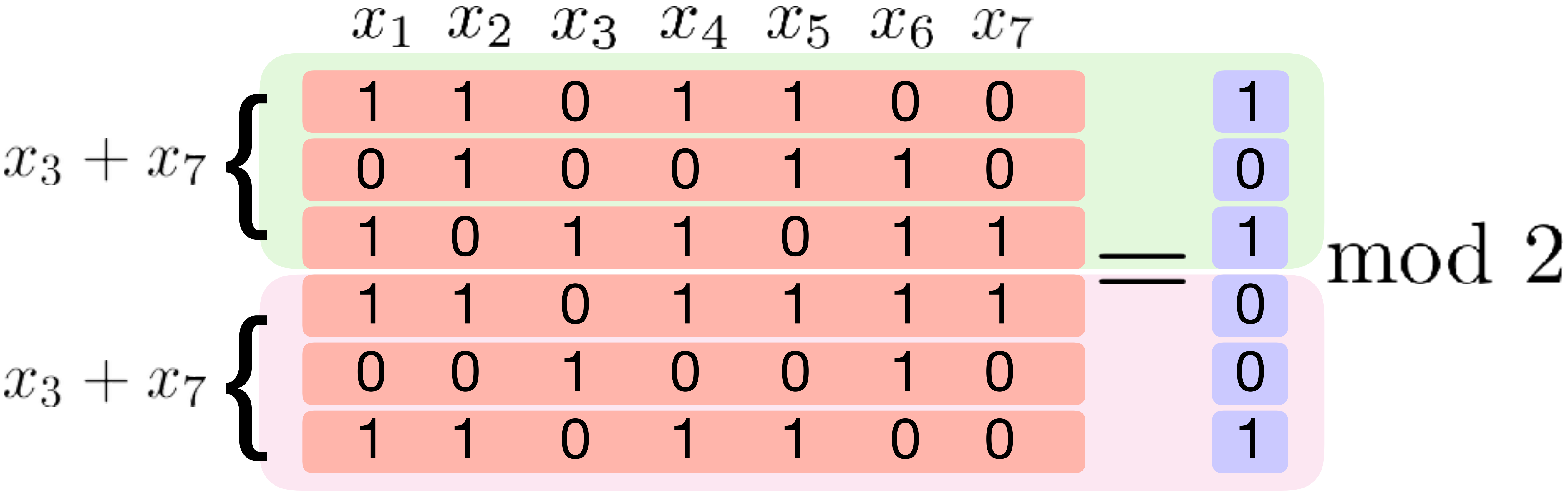
Sum to
0=1 mod2

$$\begin{array}{cccccccccc}
 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & & \\
C_1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & & 1 \\
C_2 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & & 0 \\
C_3 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & = & 1 \\
C_4 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & & 0 \\
C_5 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & & 0 \\
C_6 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & & 1 \\
C_7 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & & 1
\end{array} \quad \mathrm{mod}\ 2$$

Proof Sketch

Idea: Split set of constraints in half. For any $\{0, 1\}$ assignment, one of the halves has a falsified equation.

- The sum of the constraints tell us which one!



$$\begin{array}{cccccccccc}
 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & & \\
 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & & 1 \\
x_3 + x_7 \left\{ \right. & 0 & 1 & 0 & 0 & 1 & 1 & 0 & & 0 \\
 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & = & 1 \\
 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & & 0 \\
x_3 + x_7 \left\{ \right. & 0 & 0 & 1 & 0 & 0 & 1 & 0 & & 0 \\
 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & & 1
\end{array} \mod 2$$

Proof Sketch

Idea: Split set of constraints in half. For any assignment, one of the halves has a falsified equation.

- The sum of the constraints tell us which one!

$x_3 + x_7$ must be $0 \bmod 2$

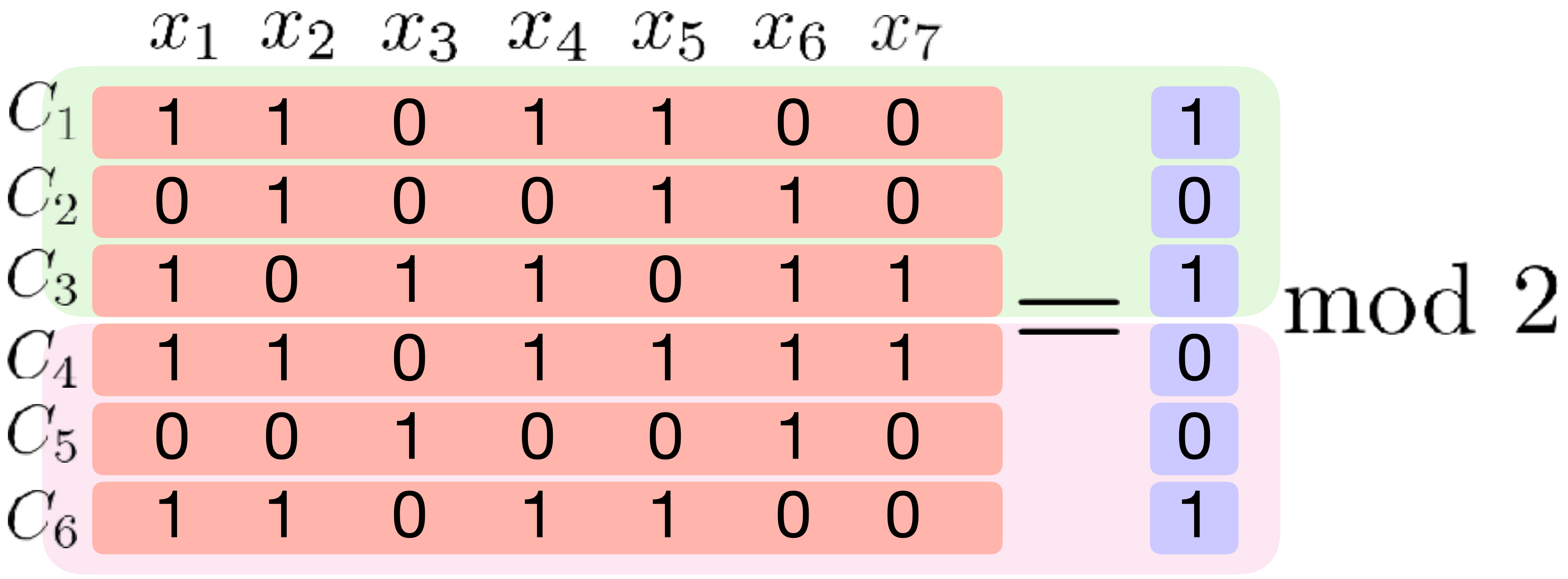
$$\begin{array}{c} \\ x_3+x_7\left\{\begin{array}{c}\\\\\end{array}\right. \\ x_3+x_7\left\{\begin{array}{c}\\\\\end{array}\right. \end{array}
\begin{array}{ccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 \end{array}
= \begin{array}{c} \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{array} \bmod 2$$

$x_3 + x_7$ must be $1 \bmod 2$

Proof Sketch

Recursive Step:

1. Partition constraints into two sets S_1, S_2



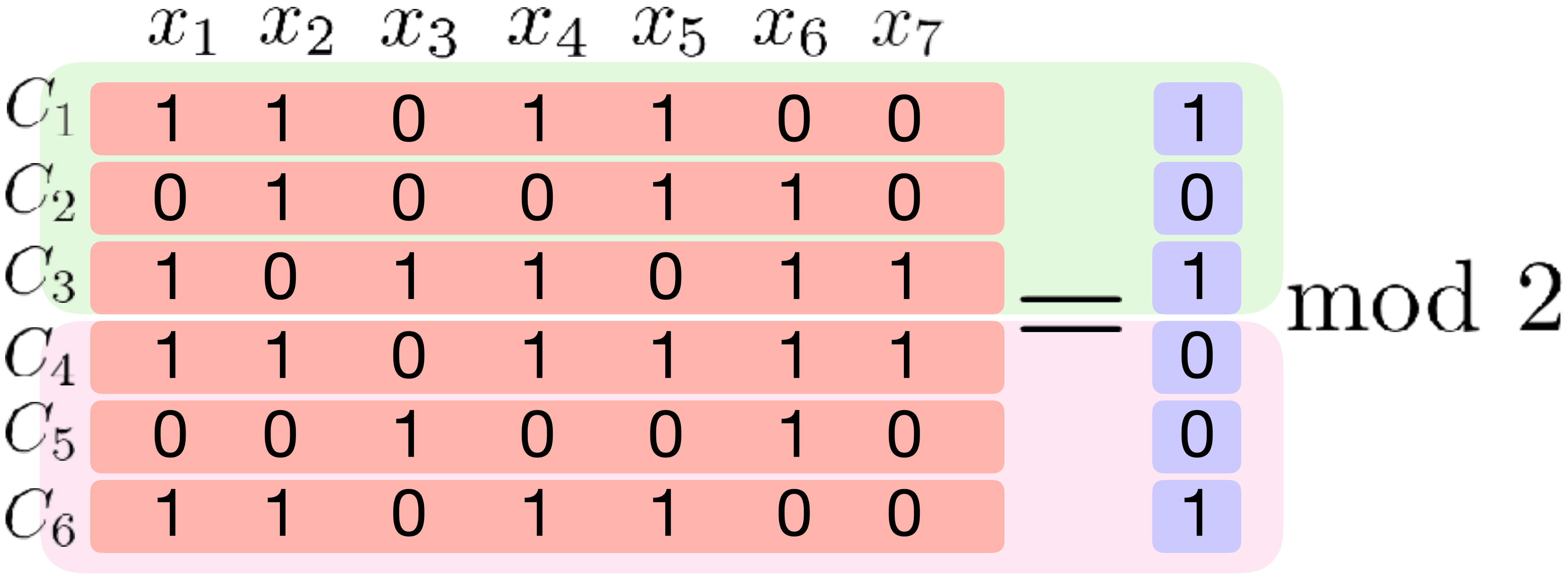
$$\begin{array}{cccccccc} & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \\ C_1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ C_2 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ C_3 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ C_4 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ C_5 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ C_6 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \end{array} = \begin{array}{c} \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{array} \mod 2$$

Proof Sketch

Recursive Step:

1. Partition constraints into two sets S_1, S_2
2. Determine value of sum of constraints $\bmod 2$ in both sets

$\sum_{C_i \in S_1} \geq k$ for $k = 1, \ldots, \max$

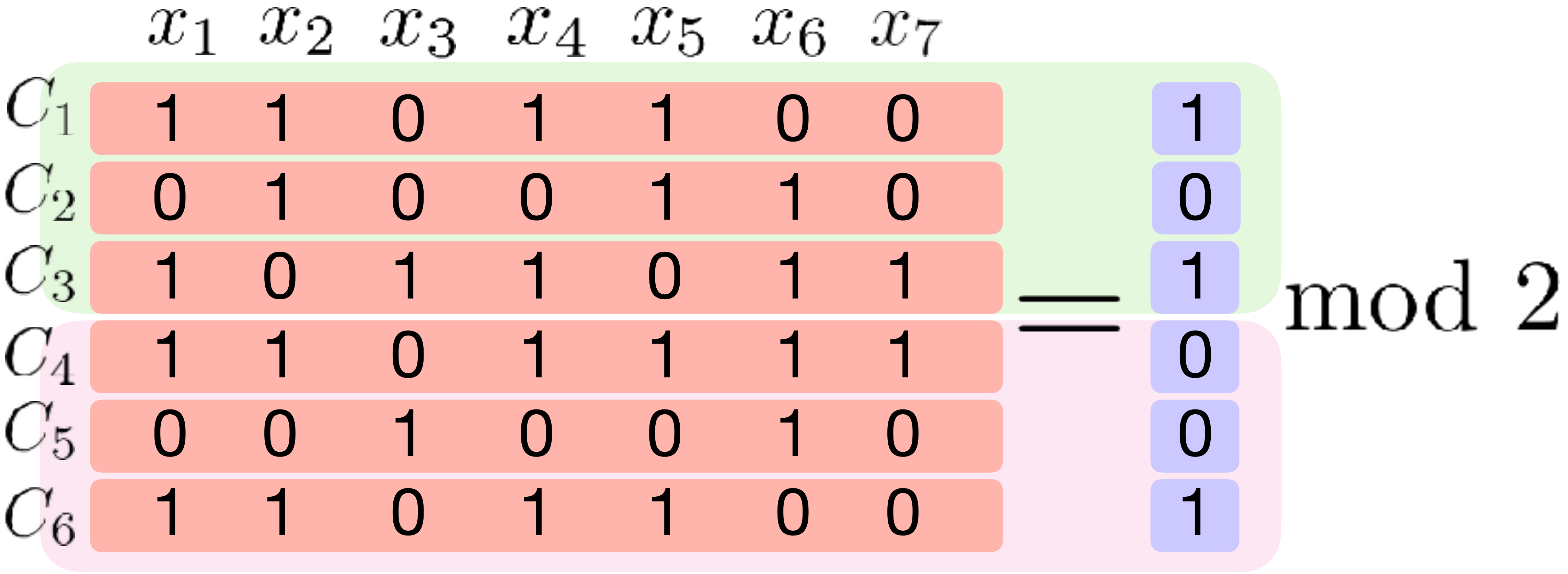


$$\begin{array}{c|ccccccc|c|c} & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & & \\ C_1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & & 1 \\ C_2 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & & 0 \\ C_3 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & = & 1 \\ C_4 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & & 0 \\ C_5 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & & 0 \\ C_6 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & & 1 \end{array} \bmod 2$$

Proof Sketch

Recursive Step:

1. Partition constraints into two sets S_1, S_2
2. Determine value of sum of constraints $\bmod 2$ in both sets
3. Recurse on set whose sum is not satisfied



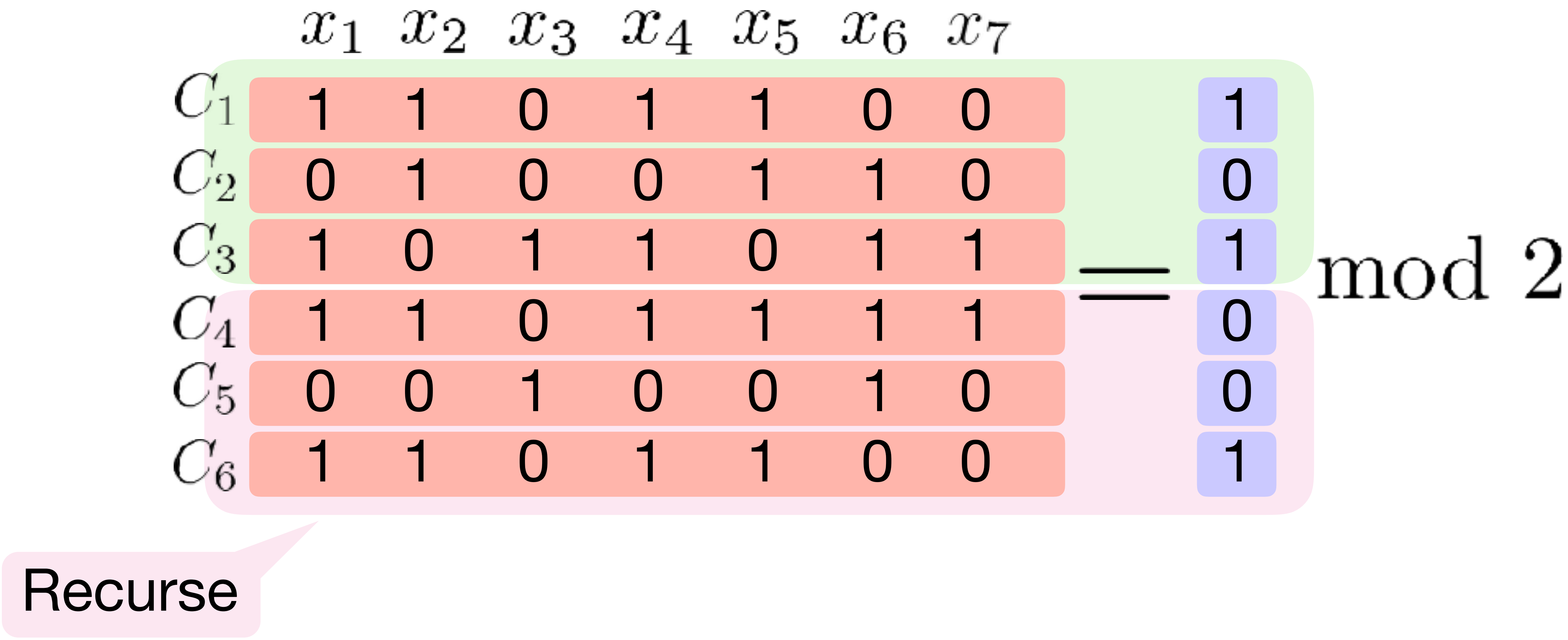
$$\begin{array}{cccccccc} & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \\ C_1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ C_2 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ C_3 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ C_4 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ C_5 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ C_6 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \end{array} = \begin{array}{c} \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{array} \bmod 2$$

Proof Sketch

Recursive Step:

1. Partition constraints into two sets S_1, S_2
2. Determine value of sum of constraints $\bmod 2$ in both sets
3. Recurse on set whose sum is not satisfied

Suppose: $x_3 + x_7 \equiv 0 \bmod 2$



$$\begin{array}{c|ccccccc|c|c}
 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & & \\
C_1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & & 1 \\
C_2 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & & 0 \\
C_3 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & = & 1 \\
C_4 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & & 0 \\
C_5 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & & 0 \\
C_6 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & & 1
\end{array} \bmod 2$$

Recurse

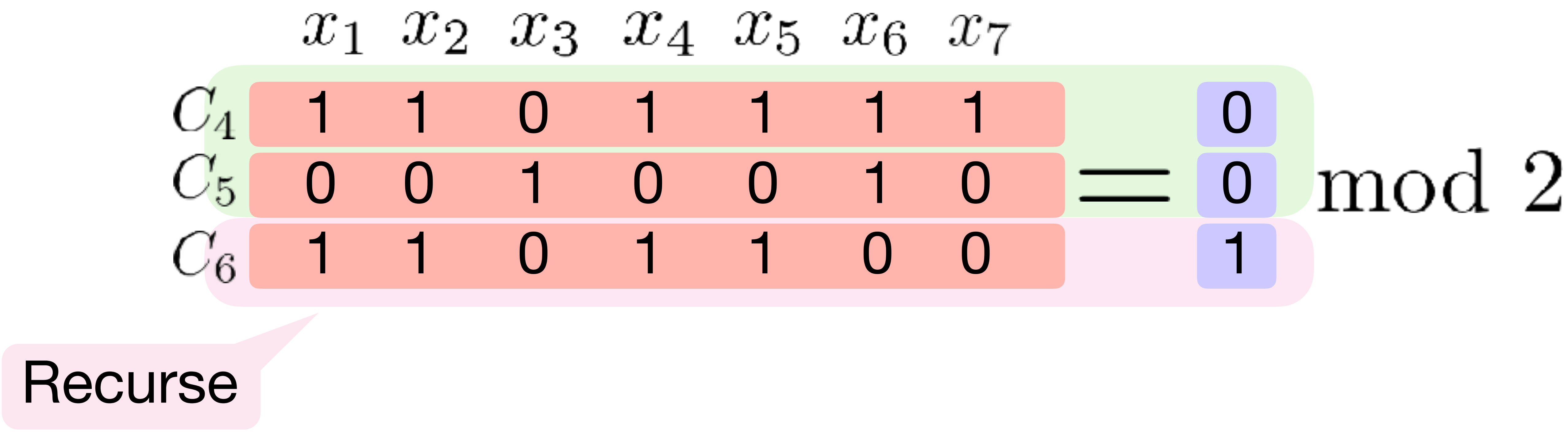
Proof Sketch

Recursive Step:

1. Partition constraints into two sets S_1, S_2
2. Determine value of sum of constraints $\bmod 2$ in both sets
3. Recurse on set whose sum is not satisfied

Suppose:

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_7 \equiv 0 \bmod 2$$
$$x_1 + x_2 + x_4 + x_5 \equiv 0 \bmod 2$$



$$\begin{array}{c|ccccccc|c|c} & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & & \\ C_4 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & & 0 \\ C_5 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & = & 0 & \bmod 2 \\ C_6 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & & 1 \end{array}$$

Recurse

Proof Sketch

Termination:

By recursive step we have derived $x_1 + x_2 + x_4 + x_5 \equiv 0 \bmod 2$
Using the constraint $x_1 + x_2 + x_4 + x_5 \equiv 1 \bmod 2$ we can derive $0 \geq 1$

$$\begin{array}{cccccccccc} & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & & \\ C_6 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & = & 1 \bmod 2 \end{array}$$

Proof Sketch

Termination:

By recursive step we have derived $x_1 + x_2 + x_4 + x_5 \equiv 0 \bmod 2$
Using the constraint $x_1 + x_2 + x_4 + x_5 \equiv 1 \bmod 2$ we can derive $0 \geq 1$

Analysis:

Each recursive step requires branches $O(n)$
$O(\log n)$ recursive steps - Reduce the set of constraints by half each time

Other Results

SP can solve systems of linear equations over a finite field

- quasi-polynomial size proofs of *Tseitin formulas* (conjectured to be hard for Cutting Planes)

Other Results

SP can solve systems of linear equations over a finite field

- quasi-polynomial size proofs of *Tseitin formulas* (conjectured to be hard for Cutting Planes)

SP can polynomially simulate Cutting Planes

- size s CP proof $\Longrightarrow$ size $O(s)$ SP proof
- Surprising because SP is tree-like, while CP is DAG-lik

Other Results

SP can solve systems of linear equations over a finite field

- quasi-polynomial size proofs of *Tseitin formulas* (conjectured to be hard for Cutting Planes)

SP can polynomially simulate Cutting Planes

- size s CP proof $\Longrightarrow$ size $O(s)$ SP proof
- Surprising, SP proofs are tree-like, while CP proofs are DAG-like

$\Omega(n/\log^2 n)$ depth lower bound

- reduction to real communication complexity
- same technique cannot give size lower bounds
  - real communication protocols can't be balanced,
  - SP proofs can't be balanced

Open Problem

Super-polynomial size lower bounds on SP?

Does SP proof size (#nodes) equal SP proof bit-size?

- [Muroga72] Any integer-linear inequality separating two subsets U,V subset $\{0,1\}^n$ can be represented by poly(n) bits

Separate Cutting Planes and Stabbing Planes

- Candidate: Tseitin formulas

SP-based search algorithms?